

OpenCable™ Application Platform Specification

OCAP Digital Video Recorder (DVR)

OC-SP-OCAP-DVR-I02-050524

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 2005 Cable Television Laboratories, Inc. All rights reserved.

Document Status Sheet

Document Control Number:	OC-SP-OCAP-DVR-I02-050524			
Document Series:	OpenCable™ Application Platform Specification			
Reference:	OCAP Digital Video Recorder (DVR)			
Revision History:	I01-040524, Issued Specification I02-050524, Issued Specification			
Date:	May 24, 2005			
Status Code:	Work in Process	Draft	Issued	Closed
Distribution Restrictions:	Author Only	CL/Member	CL/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	3
1.1	OCAP DVR PURPOSE	3
1.2	OCAP DVR REQUIREMENTS	3
1.3	OCAP DVR APPLICATION AREAS (INFORMATIVE)	3
1.3.1	<i>Personal Video Recorder (PVR)</i>	3
1.3.2	<i>Time Shift</i>	4
1.3.3	<i>Pushed Content</i>	4
2	REFERENCES.....	5
2.1	NORMATIVE REFERENCES	5
3	DEFINITIONS & ABBREVIATIONS	7
3.1	DEFINITIONS	7
3.2	ABBREVIATIONS	7
4	CONVENTIONS.....	9
4.1	PROFILE LANGUAGE.....	9
4.2	ORGANIZATION.....	9
5	GENERAL CONSIDERATIONS	11
5.1	INTRODUCTION	11
5.2	RELATIONSHIP WITH OCAP AND GEM SPECIFICATIONS	11
5.3	BASIC ARCHITECTURE (INFORMATIVE).....	11
6	RECORDING AND PLAYBACK PROCESS	15
6.1	DVB-GEM SPECIFICATION CORRESPONDENCE.....	15
6.2	OCAP 1.0 DVR SPECIFIC REQUIREMENTS	16
6.2.1	<i>Extensions to A088 MHP PVR/PDR Common Core</i>	16
7	RECORDING AND PLAYBACK APIS	25
7.1	DVB-GEM SPECIFICATION CORRESPONDENCE.....	25
7.2	OCAP 1.0 DVR SPECIFIC REQUIREMENTS	25
7.2.1	<i>Extensions to A088 MHP PVR/PDR Common Core</i>	25
8	SIGNALING.....	29
9	APPLICATION MODEL	31
10	SECURITY.....	33
10.1	DVB-GEM SPECIFICATION CORRESPONDENCE.....	33
10.2	OCAP 1.0 DVR SPECIFIC REQUIREMENTS	33
10.2.1	<i>Extensions to A088 MHP PVR/PDR Common Core</i>	33

11 MINIMUM PLATFORM CAPABILITIES 35

11.1 DVB-GEM SPECIFICATION CORRESPONDENCE 35

11.2 OCAP 1.0 DVR SPECIFIC REQUIREMENTS 35

11.2.1 Extensions to A088 MHP PVR/PDR Common Core 35

I.1 ECNS INCLUDED IN OC-SP-OCAP-DVR-I02-050524 167

List of Figures

FIGURE 5-1:	DVR ARCHITECTURE	12
FIGURE 6-1:	RECORDING REQUEST STATE DIAGRAM	18

This page intentionally left blank.

List of Tables

TABLE 2-1	OCAP DVR NORMATIVE REFERENCES	5
TABLE 7-1	CORRELATION BETWEEN OCAP 1.0 AND A088 MHP PVR/PDR COMMON CORE SPECIFICATION.....	25
TABLE 10-1	CORRELATION BETWEEN OCAP 1.0 AND A088 MHP PVR/PDR COMMON CORE SPECIFICATION.....	33
TABLE 10-2	SECURITY RESTRICTIONS FOR INDIVIDUAL RECORDING REQUESTS	33
TABLE 11-1	CORRELATION BETWEEN OCAP 1.0 AND A088 MHP PVR/PDR COMMON CORE SPECIFICATION.....	35
TABLE A-1	CORRELATION BETWEEN OCAP 1.0 AND A088 MHP PVR/PDR COMMON CORE SPECIFICATION.....	37
TABLE B-1	MAPPING FOR GEM REQUIRED RESPONSIBILITIES	39
TABLE C-1	CORRELATION BETWEEN OCAP 1.0 AND A088 MHP PVR/PDR COMMON CORE SPECIFICATION.....	41
TABLE I-1	ECNS INCORPORATED IN OC-SP-OCAP-DVR-I02-050524	167

This page intentionally left blank.

Summary

This document defines a minimal profile 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 [2]). The OCAP 1.0 and OCAP DVR Profile, 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 Profile is based on the OCAP 1.0 Profile, and includes that document in its entirety.

Purpose

The 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 Profile will be executed on OpenCable-compliant host devices. The OCAP DVR Profile 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 the 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 for a Digital Video Recorder (DVR) software environment for digital cable receivers with local storage, and is a modular extension to the OpenCable Application Platform 1.0 (OCAP 1.0 [2]). The OCAP 1.0 Profile, and the OCAP DVR Profile (this document), were developed by Cable Television Laboratories, Inc. (CableLabs), in conjunction with representatives from its member cable operating companies, as well as leading software and hardware firms.

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

1.1 OCAP DVR Purpose

The 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 will be executed on OpenCable-compliant host devices. The OCAP DVR 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 the 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 encounter, such that broadcast event descriptions might be maintained by applications, and that the platform supports DVR applications deployed by various service providers that have no a priori knowledge of each other, and may compete for resources.

1.3 OCAP DVR Application Areas (informative)

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

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 [2].

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 Profile.

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 Profile:

Table 2-1 OCAP DVR Normative References

OCAP 1.0 Index	References	Edition	Description	Note
[1]	OCAP Security	OC-SP-SEC-I05-040831	OpenCable System Security Specification	[1]
[2]	OCAP 1.0	OC-SP-OCAP1.0-I15-050415	OpenCable Application Platform Specification, OCAP 1.0 Profile	[1]
[3]	OCAP HOST2.0	OC-SP-HOST2.0-CFR-I05-050503	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)	
[7]	A088 MHP PVR/PDR Common Core Specification	2005 April	Digital Video Broadcasting (DVB); Digital Recording Extension to Globally Executable MHP (GEM) www.mhp.org/documents/mhp_a088.zip	

This page intentionally left blank.

3 Definitions & Abbreviations

3.1 Definitions

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

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

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.

Trans-rating A process of modifying MPEG compression to achieve greater compression.

3.2 Abbreviations

DVR Digital Video Recorder

PVR Personal Video Recorder

This page intentionally left blank.

4 Conventions

The following conventions are used in this manual:

- The Courier New 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.
- *Arial Italicized* font type is used to indicate section titles when referenced within the text.

4.1 Profile 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 Profile.

MUST NOT/SHALL NOT

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

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.

4.2 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 OCAP DVR Profile adds packages to the OCAP 1.0 Profile, as seen in Annex D, *OCAP DVR API (org.ocap.dvr)*, Annex E, *OCAP DVR Storage API (org.ocap.dvr.storage)*, Annex F, *OCAP Shared DVR API (org.ocap.shared.dvr)*, Annex G, *OCAP Shared DVR Navigation API (org.ocap.shared.dvr.navigation)*, and Annex H, *OCAP DVR Shared Media API (org.ocap.dvr.shared.media)*.

This page intentionally left blank.

5 General Considerations

5.1 Introduction

This Profile fully defines a DVR extension to OCAP 1.0 [2]. This Profile 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.

5.2 Relationship with OCAP and GEM Specifications

Implementers of this Profile 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].

With exceptions noted elsewhere in this document, all normative clauses of A088 MHP PVR/PDR Common Core Specification [7], Digital Video Broadcasting (DVB); Digital Recording Extension to Globally Executable MHP, SHALL apply.

5.3 Basic Architecture (Informative)

The figure below shows an overview of the architecture assumed by the present document. Several aspects are omitted for the sake of clarity.

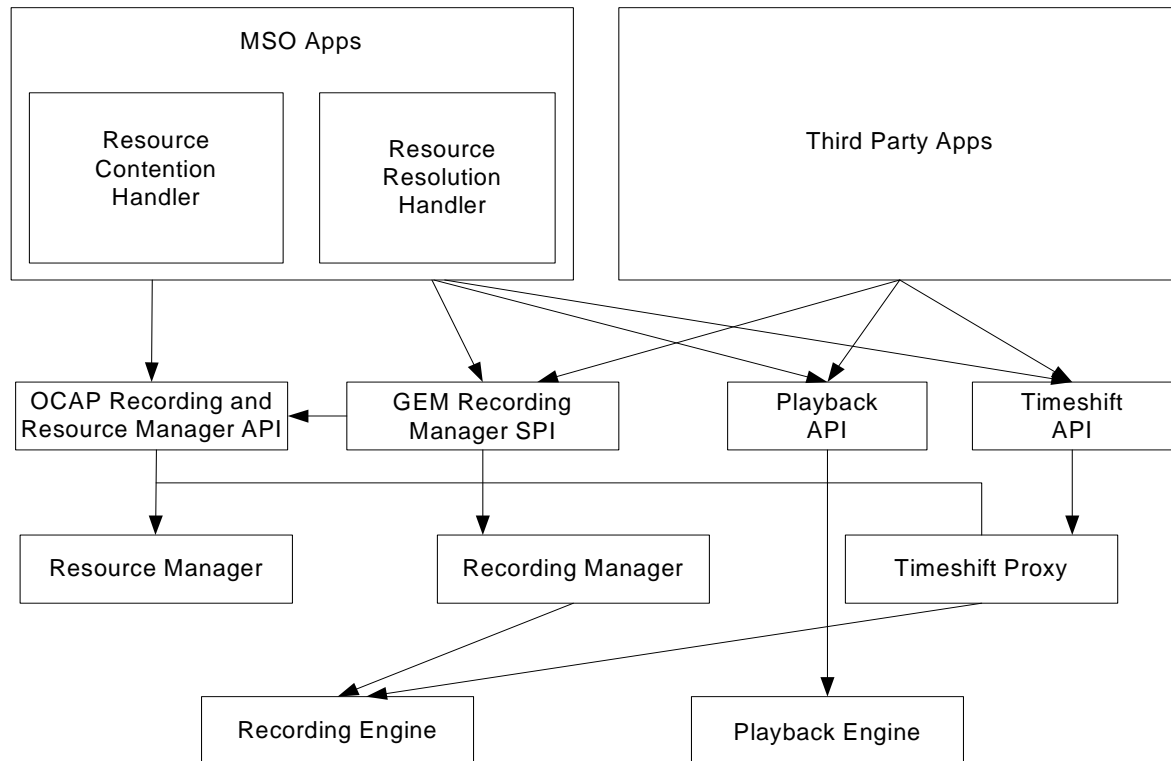


Figure 5-1: DVR Architecture

In this architecture, among other things, the Recording manager is responsible for:

- Managing pending recording requests.
- Interfacing with the resource manager to reserve resources required for pending and current recordings.
- Updating the conflict status of pending recording requests based on the resource availability.
- Interfacing with the recording engine to start and stop recordings at appropriate times.
- Updating the status of recording requests and creating `RecordedServices` as recordings are made.

Resource Manager in this architecture is an extension of the resource manager functionality required to implement the OCAP 1.0 specification. Among other things, the resource manager is responsible for:

- Detecting resource conflicts between pending recording requests
- Detecting resource conflicts between pending and inprogress recording requests
- Detecting resource conflicts between recording requests and other activities that require resource reservations such as time-shift buffering, presentation of broadcast services by services contexts, the `NetworkInterfaceController` `reserve` method, etc.
- Notifying the resource contention handler when the resource conflicts are detected
- Resolving conflicts based on the priority specified by MSO applications.

In this architecture, an MSO application may register to handle resource contentions. When the resource manager detects a resource contention, the contention handler is invoked to resolve the resource contention.

An MSO application may register to handle request resolution. Typically this would be an application that has access to EPG listing data. When the recording manager encounters a recording request that requires additional information to process, this request resolution handler is invoked. The request resolution handler may process series requests and schedule recording requests corresponding to individual episodes.

Applications may use the GEM recording manager API to:

- Request recordings to be made
- Manage the list of recording requests maintained by the recording manager
- Manage recorded services.

This page intentionally left blank.

6 Recording and Playback Process

This chapter describes the recording and playback process for scheduled recordings and time-shift recording.

6.1 DVB-GEM Specification Correspondence

Section 6 Recording and Playback Process (this section) corresponds to A088 MHP PVR/PDR Common Core Specification [7], Chapter 6 as follows:

Table 6-1 Correlation between OCAP 1.0 and A088 MHP PVR/PDR Common Core Specification

OCAP Compliance	A088 MHP PVR/PDR Common Core Specification Section	GEM Compliance
6 "Recording and Playback Process"	6 Recording and Playback Process	Extension
6.2.1.1 "Scheduled Recordings"	No corresponding section	Extension
6.2.1.1.1 "RecordingSpecs"	No corresponding section	Extension
6.2.1.1.2 "Managing recording requests"	6.1 Managing scheduled recording	Extension
6.2.1.1.3 "The recording process"	6.2 The recording process	Extension
6.2.1.1.3.1 "Identifying streams to be recorded"	No corresponding section	Extension
6.2.1.1.3.2 "Identifying and recording applications"	No corresponding section	Extension
6.2.1.1.4 "Managing completed recordings"	6.3 Managing completed recordings	
6.2.1.1.5 "Recording request state transitions (Informative)"	No corresponding section	Extension
6.2.1.1.6 "Resource Management for recording requests"	No corresponding section	Extension
6.2.1.1.7 "Request Resolution Process"	No corresponding section	Extension
6.2.1.1.8 "RecordedService"	No corresponding section	Extension
6.2.1.2 "Playback of recorded services"	6.4 Playback of scheduled recordings	Extension
6.2.1.2.1 "Process for playback"	6.4.1 Process of playback	Extension
No corresponding section	6.4.2 Event during playback	Complete compliance
6.2.1.3 "Time-Shift Buffer"	6.5 Time-shift	Extension
6.2.1.3.1 "Overview (Informative)"	No corresponding section	Extension
6.2.1.3.2 "Recording"	6.5.1 Recording	Extension
6.2.1.3.2.1 "Identifying the streams to be recorded"	No corresponding section	Extension
6.2.1.3.2.2 "Identifying and recording applications"	No corresponding section	Extension
6.2.1.3.3 "Playback"	6.5.2 Playback	Extension
6.2.1.3.4 "Resource Management"	No corresponding section	Extension
6.2.1.4 "Storage"	No corresponding section	Extension
6.2.1.4.1 "Storage Management"	No corresponding section	Extension
6.2.1.4.2 "Storage Initialization"	No corresponding section	Extension

6.2 OCAP 1.0 DVR Specific Requirements

6.2.1 Extensions to A088 MHP PVR/PDR Common Core

6.2.1.1 Scheduled Recordings

6.2.1.1.1 RecordingSpecs

Applications may use an extension of the `RecordingSpec` class to specify a recording request. Implementations SHALL support the following extensions of `RecordingSpec` passed in as an argument to the `RecordingManager.record` method:

- `org.ocap.shared.dvr.LocatorRecordingSpec`
- `org.ocap.shared.dvr.ServiceRecordingSpec`
- `org.ocap.shared.dvr.ServiceContextRecordingSpec`
- `org.ocap.dvr.PrivateRecordingSpec`

Implementations SHALL support `RecordingProperties` specified through the extension `OcapRecordingProperties`.

6.2.1.1.2 Managing Recording Requests

In addition to the activities defined in clause 6.1 of A088 MHP PVR/PDR Common Core Specification [7], the process of managing recording requests SHALL include the following:

- a) Detecting conflicts between multiple recording requests and between recording requests and resource reservations for other activities in the system. When conflicts are detected, the conflicts are resolved as specified in Section 6.2.1.1.6, *Resource Management for Recording Requests* for recording requests. After the conflict is resolved, status of the recording request SHALL be updated based on the conflict resolution. Recording requests that are not expected to be recorded due to resource conflicts SHOULD NOT be deleted by the implementation unless explicitly deleted by the application.
- b) Invoking the Request resolution handler when the record method is called with an instance of `PrivateRecordingSpec`. Details of how the request resolution process SHOULD be handled is defined in Section 6.2.1.1.7, *Request Resolution Process*.

6.2.1.1.3 The Recording Process

The recording process as defined in clause 6.2 of A088 MHP PVR/PDR Common Core Specification [7], SHALL be followed.

In addition to the activities defined in clause 6.3 of A088 MHP PVR/PDR Common Core Specification [7], the process for managing completed recordings SHALL include the following activity:

- If the implementation estimates that a recording request that is in progress may not complete successfully due to lack to storage space available, the implementation SHOULD update the status of the recording in progress to reflect that. The estimation need not be accurate and any proprietary algorithm may be used for this computation. Applications SHOULD use this only as an advisory notification and may use its own mechanisms to decide whether sufficient space is available to complete the recording.

If a `ServiceContextRecordingSpec` is used as the parameter to the record method, the implementation SHALL store and record the relevant portion already contained within the time-shift buffer when the `startTime` is in the past.

The time-shift buffer MAY retain only a portion of the time-shift buffer content after the start of a recording initiated using a `ServiceContextRecordingSpec`. Recordings initiated with the invocation of the `record` method with a `ServiceContextRecordingSpec` as the parameter shall not be terminated if the service context is destroyed. In this case the recording shall continue until the scheduled end time. The method `ServiceContextRecordingSpec.getServiceContext()` shall return `null` if the service context has been destroyed.

6.2.1.1.3.1 Identifying Streams to be Recorded

If the recording request was specified using a `ServiceRecordingSpec` or a `ServiceContextRecordingSpec`, the implementation SHALL:

- a) Record the audio and video streams that are present on the Service up to the limits in the recording capability of the OCAP-DVR device. NOTE: Minimum capabilities for recording streams are defined in Section 11.2.1.4, *Recording Multiple Streams from the Same Service*, of the present document.
- b) Record Closed Captioning information and Content Advisory information if these are included in the Service.
- c) Manage (increment where needed) CCI bits and store them along with the stream if CCI bits are present in the broadcast stream.
- d) Create a `TimeLine` accessible through the `TimeLineControl` API corresponding to each time-base associated with the Service as signalled through `DSMCC NPTReferenceDescriptors`.

Note: Recorded streams may be stored in a proprietary format in the OCAP DVR device.

6.2.1.1.3.2 Identifying and Recording Applications

If an application is signalled as to be recorded (the scheduled recording flag in the application recording descriptor is set to '1') and if the application does not rely on the use of dynamic data during its execution (`dynamic_flag` in the application recording descriptor is set to 0) then the application SHALL be recorded. Implementations MAY record all applications. Implementations MAY record dynamic data associated with the applications and make them available through data access APIs in a manner consistent with access of data from broadcast streams.

6.2.1.1.4 Managing Completed Recordings

In addition to the activities defined in clause 6.3 of A088 MHP PVR/PDR Common Core Specification [7], the process for managing completed recordings SHALL include the following activities:

- a) Deleting the `RecordedService` once the expiration period is past. The implementation SHALL NOT allow any application to access `RecordedServices` past the expiration period. If a playback is in progress when the recording's expiration period is reached the playback SHALL be terminated. The implementation SHALL delete the recording within 1 hour after the expiration of the recording.
- b) Maintaining the recording and all associated recording requests until the recording is explicitly deleted by the application or until the recording has expired. Associated recording requests for a recording in this context are the recording request that caused the recording, the parent recording request for that recording request and all ancestors of that recording request.
- c) Maintaining failed recording requests until the recording request is explicitly deleted by the application or until the expiration period of the recording request.
- d) Maintain leaf recording requests in the `DELETED_STATE` and any completely resolved parent recording requests corresponding to deleted recorded services for a period of 28 days after the time of deletion or until explicitly deleted by an application. Implementation shall delete these recordings requests 28 days after the deletion of the recorded service.

6.2.1.1.5 Recording Request State Transitions (Informative)

The following diagram illustrates possible transitions between various states of the recording request:

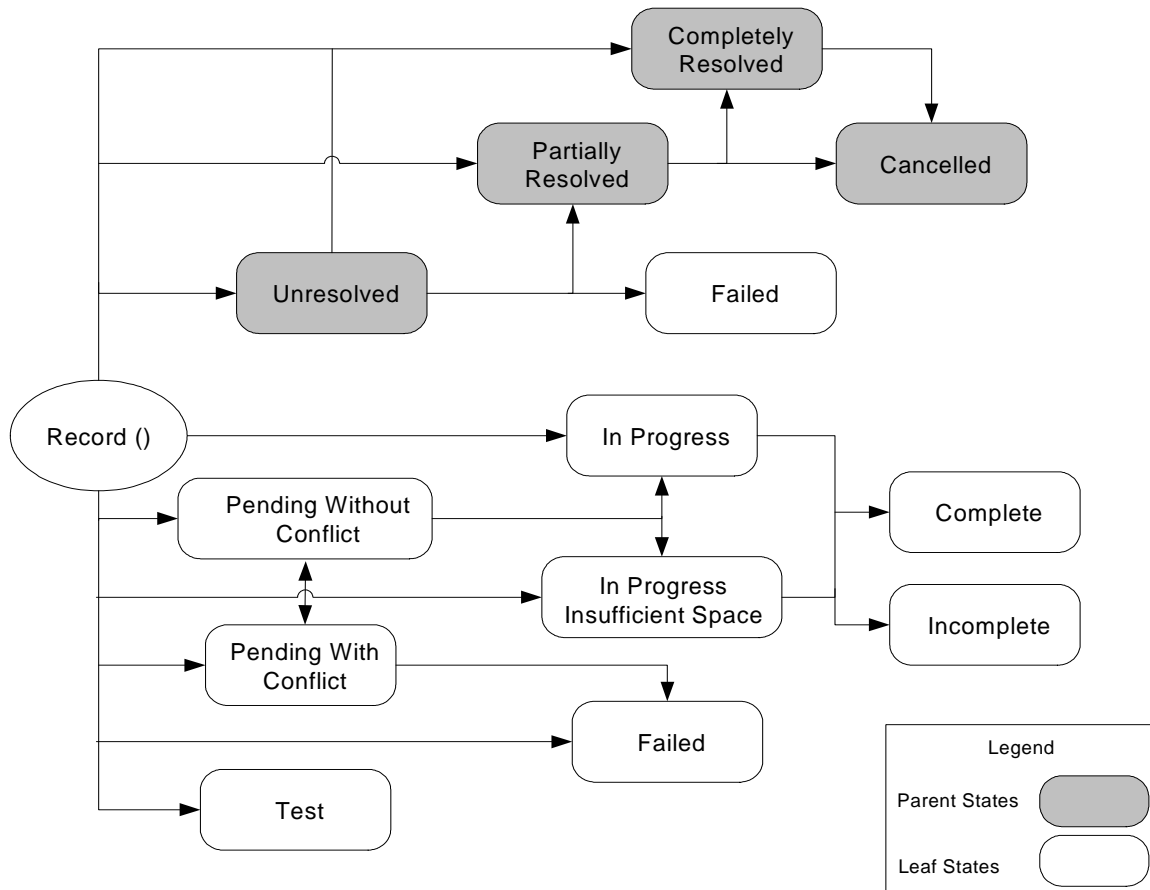


Figure 6-1: Recording Request State Diagram

6.2.1.1.6 Resource Management for Recording Requests

The implementation SHALL perform the following resource management activities for recording requests:

- Create an instance of `RecordingResourceUsage` corresponding to each recording request in one of the pending states. The method `getResource(...)` may return `null` for `RecordingResourceUsages` corresponding to pending recording request.
- When new recording requests are inserted or when existing recording requests are modified, detect conflicts by checking if resources are sufficient to complete all recording requests in pending without conflict state and in-progress states. If conflicts are detected, resolve the conflict as specified in Section 19.2.1.1 of the OCAP 1.0 [2]. Update the states of pending recording requests based on the conflict resolution. If the resolution of conflict resulted in a recording request in in-progress state losing resources needed to continue recording, the recording SHALL be terminated and the state for the recording request SHALL be changed to incomplete state.

- c) Before starting a recording, create an instance of `RecordingResourceUsage` corresponding to the recording request and reserve the resources required for the recording. The method `getResource(...)` for this recording resource usage SHALL NOT return null for any resource names returned by the method `getResourceNames()`.

6.2.1.1.7 Request Resolution Process

When recording requests are specified by applications using `PrivateRecordingSpec`, the implementation SHALL invoke the request resolution handler to resolve the request. The request resolution process includes the following:

- a) When the `RecordingManager.record()` method is called with a `PrivateRecordingSpec` as an argument, the implementation SHALL create a `ParentRecordingRequest` in UNRESOLVED state. The implementation SHALL invoke any registered request resolution handler with the newly created recording request as the parameter.
- b) When the `OcapRecordingManager.resolve(...)` method is called with a `PrivateRecordingSpec` as an argument, the implementation SHALL create a `ParentRecordingRequest` in UNRESOLVED state and make that a child of the recording request that was passed as an argument to the `resolve(...)` method. The implementation SHALL invoke any registered request resolution handler with the newly created recording request as an argument.
- c) When the `OcapRecordingManager.resolve(...)` method is called with a `RecordingSpec` other than the `PrivateRecordingSpec` as an argument, the implementation SHALL create a `LeafRecordingRequest` and make that a child of the recording request that was passed as an argument to the `resolve(...)` method.

6.2.1.1.8 RecordedService

The implementation SHALL create a `RecordedService` when a recording request enters one of the ‘in progress’ states. A `RecordedService` SHALL NOT be listed in `ServiceLists` returned by the `SIManager` class in the JavaTV service package. The `getLocator()` method of `RecordedService` SHALL return an OCAP Locator which is different from the originating service Locator. The `getName()` method for a recorded service SHALL return a unique name starting with the string “RecordedService”. The method `getServiceType` SHALL return `RECORDED_SERVICE`. The method `hasMultipleInstance` SHALL always return `false`. The implementation SHALL support the `retrieveDetails` method. All methods in objects implementing `ServiceDetails` for `RecordedServices` SHOULD function the same as objects implementing the `ServiceDetail` for broadcast services.

6.2.1.2 Playback of Recorded Services

6.2.1.2.1 Process for Playback

The process for playback SHALL be as defined in clause 6.4 of A088 MHP PVR/PDR Common Core Specification [7] and additionally constrained as follows:

- a) When playing content which is currently being recorded, if the end of the content is reached and recording stops, the playback must continue without interruption at this point, regardless of any (implementation dependent) process to copy the newly recorded content from any temporary buffer to a more permanent location on the storage device.
- b) When a recorded service is selected on a service context, the playback SHALL begin from the media time set using the method `RecordedService.setMediaTime(...)`.

- c) If the playback location is the same as the recording point (playing back the live point), the implementation SHALL display the broadcast stream rather than the stream coming off the storage medium.

The following rules apply to players presenting recorded content, both associated with a service context, presenting a recorded service, and those directly created from a `MediaLocator`:

- a) 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.
- b) 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.
- c) Audio SHALL NOT be presented for any playback rate other than 1.0
- d) If recording is ongoing, and if the playback (at a rate more than 1.0) hits the end of the recorded content, the implementation SHALL set the playback rate to 1.0 and send an `EndOfContentEvent` to any registered controller listeners. If the recording is not ongoing, the implementation SHALL set the playback rate to 0.0 and send an `EndOfContentEvent` to any registered controller listeners.
- e) 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 `BeginningOfContentEvent` to any registered controller listeners.

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

6.2.1.3 Time-Shift Buffer

6.2.1.3.1 Overview (Informative)

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. 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. A time-shift buffer may be used by applications to initiate circular buffering of broadcast services and to present buffered services on a service context. When a time-shift buffer is not attached to a service context, an application may select a broadcast service on the time-shift buffer. The time-shift buffer would start buffering the service without presenting the service. To present the service being buffered the time-shift buffer should be attached to a service context. When a time-shift buffer is attached to a service context and a service is selected on the service context, the selected service is simultaneously decoded and displayed and stored in the time-shift buffer. 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. Once the time-shift buffer is attached to a service context, invoking the `select` method on the attached service context may flush the contents of the time-shift buffer and buffering will be initiated for the newly selected service. If the newly selected service is the same as the service that is being buffered, the circular buffer is not flushed. Buffering will continue even when the service context is detached.

6.2.1.3.2 Recording

The time-shift recording process as defined in clause 6.5 of A088 MHP PVR/PDR Common Core Specification [7] SHALL be followed. In addition to the activities defined in clause 6.5.1 of A088 MHP PVR/PDR Common Core Specification [7], the process for the time-shift recording SHALL include the following activity:

- 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 was broadcast since the boot or flush event.

6.2.1.3.2.1 Identifying the Streams to be Recorded

The implementation SHALL:

- a) Record the audio and video streams that are present on the Service up to the limits in the recording capability of the OCAP-DVR device.

Note: Minimum capabilities for recording streams are defined in Section 11.2.1.4, *Recording Multiple Streams from the Same Service*, of the present document.

- b) Record Closed Captioning information and Content Advisory information if these are included in the Service.
- c) Manage (increment where needed) CCI bits and store them along with the stream if CCI bits are present in the broadcast stream.

6.2.1.3.2.2 Identifying and Recording Applications

All applications bound to the broadcast service with `time_shift` flag in the application recording descriptor set to '1' SHALL be recorded. Implementations MAY record dynamic data associated with the applications and make them available through data access APIs in manner consistent with access of data from broadcast streams.

6.2.1.3.3 Playback

The time-shift playback process as defined in clause 6.5.2 of A088 MHP PVR/PDR Common Core Specification [7] SHALL be followed.

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:

- a) 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 `BeginningOfContentEvent` to any registered controller listeners. This SHOULD occur only when the playback is at a rate less than 1.0. The time-shift buffer implementation SHOULD make sure that the location corresponding to the current media time is never invalidated.
- b) If the playback location is the same as the record point (playing back the live point), the implementation SHALL display the broadcast stream rather than the recorded stream coming of the storage medium.

6.2.1.3.4 Resource Management

Resource management for the time-shift buffer SHALL be done according to the following rules:

- a) When a broadcast service is selected on an unattached time-shift buffer, the implementation SHALL create an instance of a `TimeShiftBufferResourceUsage` and attempt to reserve the resources needed for the buffering of the broadcast service. If there is a conflict, the conflict SHALL be resolved as specified in section 19.2.1.1 of the OCAP 1.0 [2].
- b) If the resources needed to continue buffering have been taken away, the implementation SHALL terminate buffering but the presentation SHALL continue on any attached service context till the playback hits the end of the buffered content. A `RecordingTerminatedEvent` is sent to any registered listeners on an attached service context.

- c) When a time-shift buffer that was buffering is attached to a service context, the resources used by the time-shift buffer SHALL be logically transferred to the service context, for example, the resource SHOULD be represented using a `ServiceContextResourceUsage` instead of a `TimeShiftBufferResourceUsage`. Similarly, when a time-shift buffer that is buffering is detached from a `ServiceContext`, the presentation on the service context SHALL be terminated and the resources SHALL be logically transferred to the time-shift buffer.
- d) When a service context is attached to a time-shift buffer, the resources used for the buffering and the presentation SHALL be represented using a `ServiceContextResourceUsage`.

6.2.1.4 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 Profile 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.

6.2.1.4.1 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 an application.

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.

6.2.1.4.2 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., re-partitioning). 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 manner.

The `MediaStorageOption` provides an overloaded `initialize()` method that allows a 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`.

This page intentionally left blank.

7 Recording and Playback APIs

This section describes recording and playback APIs particular to the OCAP DVR platform.

7.1 DVB-GEM Specification Correspondence

Section 7 Recording and Playback API (this section) corresponds to A088 MHP PVR/PDR Common Core Specification [7], Chapter 7 as follows:

Table 7-1 Correlation between OCAP 1.0 and A088 MHP PVR/PDR Common Core Specification

OCAP Compliance	A088 MHP PVR/PDR Common Core Specification Section	GEM Compliance
7.2.1.1 "Recording API"	7 Recording and playback APIs	Extension
No corresponding section	7.1 Recording and recording management	Complete compliance
No corresponding section	7.2 Playback	Complete compliance
7.2.1.1 "Recording API"	No corresponding section	Extension
7.2.1.2 "OCAP DVR API"	No corresponding section	Extension
No corresponding section	7.3 Other APIs	Complete compliance
7.2.1.3 "Permissions"	7.4 Recording and recording management	Extension
No corresponding section	7.4.1 Unsigned applications	Complete compliance
7.2.1.3.1 "Signed applications"	7.4.2 Signed applications	Extension
7.2.1.3.2 "Monitor Application Permission"	No corresponding section	Extension

7.2 OCAP 1.0 DVR Specific Requirements

7.2.1 Extensions to A088 MHP PVR/PDR Common Core

7.2.1.1 Recording API

The OCAP DVR platform extends the A088 MHP PVR/PDR Common Core Specification [7] specification and adds support for the following packages:

`org.ocap.dvr`

`org.ocap.dvr.storage`

7.2.1.2 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 D, *OCAP DVR API (org.ocap.dvr)*, Annex E, *OCAP DVR Storage API (org.ocap.dvr.storage)*, Annex F, *OCAP Shared DVR API (org.ocap.shared.dvr)*, Annex G, *OCAP Shared DVR Navigation API (org.ocap.shared.dvr.navigation)*, and Annex H, *OCAP DVR Shared Media API (org.ocap.dvr.shared.media)*. 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 packages, classes, and interfaces required by this section. These are enhancements to the APIs defined in OCAP 1.0 [2], as well as the following packages, classes, and interfaces.

GEM PVR API:

```
org.ocap.shared.dvr.LeafRecordingRequest
org.ocap.shared.dvr.ParentRecordingRequest
org.ocap.shared.dvr.RecordedService
org.ocap.shared.dvr.RecordingChangedListener
org.ocap.shared.dvr.RecordingRequest
org.ocap.shared.dvr.LocatorRecordingSpec
org.ocap.shared.dvr.RecordingChangedEvent
org.ocap.shared.dvr.RecordingManager
org.ocap.shared.dvr.RecordingPermission
org.ocap.shared.dvr.RecordingProperties
org.ocap.shared.dvr.RecordingSpec
org.ocap.shared.dvr.RecordingTerminatedEvent
org.ocap.shared.dvr.ServiceContextRecordingSpec
org.ocap.shared.dvr.ServiceRecordingSpec
org.ocap.shared.dvr.AccessDeniedException
org.ocap.shared.dvr.NoMoreDataEntriesException
org.ocap.shared.dvr.RecordingFailedException
org.ocap.shared.dvr.navigation.RecordingList
org.ocap.shared.dvr.navigation.RecordingListIterator
org.ocap.shared.dvr.navigation.AppIDFilter
org.ocap.shared.dvr.navigation.OrgIDFilter
org.ocap.shared.dvr.navigation.RecordingListComparator
org.ocap.shared.dvr.navigation.RecordingListFilter
org.ocap.shared.dvr.navigation.RecordingStateFilter

org.ocap.shared.media.MediaTimeFactoryControl
```

org.ocap.shared.media.TimeLine
org.ocap.shared.media.TimeLineControl
org.ocap.shared.media.TimeShiftControl
org.ocap.shared.media.BeginningOfContentEvent
org.ocap.shared.media.EndOfContentEvent
org.ocap.shared.media.EnteringLiveModeEvent
org.ocap.shared.media.LeavingLiveModeEvent
org.ocap.shared.media.TimeLineInvalidException
org.ocap.shared.media.TimeOutOfRangeException

OCAP DVR Extensions to GEM

org.ocap.dvr.storage.AllocateTimeShiftBufferOption
org.ocap.dvr.storage.MediaStorageOption
org.ocap.dvr.storage.MediaStorageVolume
org.ocap.dvr.storage.SpaceAllocationHandler
org.ocap.dvr.storage.TimeShiftBufferListener
org.ocap.dvr.storage.TimeShiftBufferOption
org.ocap.dvr.storage.TimeShiftBufferEvent
org.ocap.dvr.storage.TimeShiftBufferSelectedEvent
org.ocap.dvr.storage.TimeShiftBufferSelectionFailedEvent
org.ocap.dvr.storage.TimeShiftBufferTerminatedEvent

org.ocap.dvr.OcapRecordedService
org.ocap.dvr.OcapRecordingRequest
org.ocap.dvr.RecordingAlertListener
org.ocap.dvr.RecordingResourceUsage
org.ocap.dvr.RequestResolutionHandler
org.ocap.dvr.OcapRecordingManager
org.ocap.dvr.OcapRecordingProperties
org.ocap.dvr.PrivateRecordingSpec
org.ocap.dvr.RecordingAlertEvent
org.ocap.dvr.TimeShiftBufferResourceUsage

7.2.1.3 Permissions

7.2.1.3.1 Signed Applications

No additional `RecordingPermission` is assigned to signed applications. `RecordingPermissions` with names “create”, “modify”, “delete” or “cancel” SHALL NOT be granted to a signed application unless the signed application has `MonitorApplicationPermission(“recording”)` or `MonitorApplicationPermission(“handler.recording”)`.

7.2.1.3.2 Monitor Application Permission

An application with `MonitorApplicationPermission(“recording”)` is assigned the following set of permissions defined in `org.ocap.shared.dvr.RecordingPermission`:

- a) `RecordingPermission(“create”, “own”)` - schedule a `RecordingRequest`.
- b) `RecordingPermission(“read”, “own”)` - obtain the list of `RecordingRequests`.
- c) `RecordingPermission(“modify”, “own”)` - modify properties or application specific data for a `RecordingRequest`.
- d) `RecordingPermission(“delete”, “own”)` - delete a `RecordingRequest` including recorded content.
- e) `RecordingPermission(“cancel”, “own”)` - cancel a pending `RecordingRequest`.

An application with `MonitorApplicationPermission(“handler.recording”)` is assigned the following set of permissions defined in `org.ocap.shared.dvr.RecordingPermission`:

- `RecordingPermission(“*”, “**”)` - create, read, modify, delete or cancel any `RecordingRequest` or `RecordedService` regardless of any restrictions specified through the extended file access permission associated with the `RecordingRequest`.

8 Signaling

This chapter is in complete compliance with A088 MHP PVR/PDR Common Core Specification [7], Chapter 8.

This page intentionally left blank.

9 Application Model

This chapter is in complete compliance with A088 MHP PVR/PDR Common Core Specification [7], Chapter 9.

This page intentionally left blank.

10 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].

10.1 DVB-GEM Specification Correspondence

Section 10 Security (this section) corresponds to A088 MHP PVR/PDR Common Core Specification [7], Chapter 10 as follows:

Table 10-1 Correlation between OCAP 1.0 and A088 MHP PVR/PDR Common Core Specification

OCAP Compliance	A088 MHP PVR/PDR Common Core Specification Section	GEM Compliance
10 Security	10 Security	Complete compliance
Annex 10.2.1.1, Access scope of recordings	No corresponding section	Extension
Annex 10.2.1.2, Content Protection	No corresponding section	Extension
Annex 10.2.1.3, Minimum Security Constraints	No corresponding section	Extension

10.2 OCAP 1.0 DVR Specific Requirements

10.2.1 Extensions to A088 MHP PVR/PDR Common Core

10.2.1.1 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).

The following table describes the security restrictions for individual recording requests:

Table 10-2 Security restrictions for individual recording requests

Method	Policy
<code>RecordingManager.addRecordingChangedListener(RecordingListListener)</code>	No additional permissions required
<code>RecordingManager.getEntries()</code>	Only entries for which the application has read extended file access permission will be listed.
<code>RecordingManager.getEntries(RecordingListFilter)</code>	Only entries for which the application has read extended file access permission will be listed.
<code>RecordingRequest.addAppData(int, byte[])</code>	Permitted only for entries for which the application has write extended file access permission.

Table 10-2 Security restrictions for individual recording requests

Method	Policy
<code>RecordingRequest.removeAppData(int)</code>	Permitted only for entries for which the application has write extended file access permission.
<code>RecordingRequest.setRecordingProperties(RecordingSpec)</code>	Permitted only for entries for which the application has write extended file access permission.
<code>RecordingRequest.cancel()</code>	Permitted only for entries for which the application has write extended file access permission.
<code>RecordingRequest.stop()</code>	Permitted only for entries for which the application has write extended file access permission.
<code>RecordingRequest.delete()</code>	Permitted only for entries for which the application has write extended file access permission.
<code>RecordingManager.record()</code>	No additional permissions required
<code>LeafRecordingRequest.getService()</code>	No additional permissions required

10.2.1.2 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 manner that unites 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 manner that unites 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.

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 Profile.

10.2.1.3 Minimum Security Constraints

Minimum "level of security" requirements are asserted by this Profile 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-128 as defined by FIPS-197 [6]

This Profile 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.

11 Mimimum Platform Capabilities

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

11.1 DVB-GEM Specification Correspondence

Section 11, Minimum Platform Capabilities (this section) does not correspond to any A088 MHP PVR/PDR Common Core Specification [7] chapter, as depicted below:

Table 11-1 Correlation between OCAP 1.0 and A088 MHP PVR/PDR Common Core Specification

OCAP Compliance	A088 MHP PVR/PDR Common Core Specification Section	GEM Compliance
11 Minimum Platform Capabilities	No corresponding section	Extension
11.2.1.1 "Bit Rate"	No corresponding section	Extension
11.2.1.2 "Storage Devices"	No corresponding section	Extension
11.2.1.3 "Time-shift buffers"	No corresponding section	Extension
11.2.1.5 "OpenCable Set-top Terminal Core Requirements"	No corresponding section	Extension

11.2 OCAP 1.0 DVR Specific Requirements

11.2.1 Extensions to A088 MHP PVR/PDR Common Core

11.2.1.1 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 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, which 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.

11.2.1.2 Storage Devices

Implementations SHALL enable simultaneous recording and playback to/from 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.

11.2.1.3 Time-shift Buffers

Implementations SHALL support at least one 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 timeshift buffers to various service contexts.

11.2.1.4 Recording Multiple Streams from the Same Service

Implementations SHALL support simultaneous recording of at least one video stream and at least two audio streams that are broadcast as a part of a broadcast service. If a broadcast service contains more than one video stream, implementations SHALL record at least the first video stream in the PMT elementary stream loop. If the broadcast service contains more than two audio streams, the implementation shall record the audio stream matching the user preferred language, if available. If no match is found, the implementation SHALL record the first audio stream in the PMT elementary stream loop, as the preferred audio. If more than one audio stream is available in the broadcast service, the implementation SHALL arbitrarily pick a second audio stream of a different language from the first to store with the recorded service. It is implementation dependent regarding whether additional audio or video streams available in the service being recorded are actually stored with the recorded service.

11.2.1.5 OpenCable Set-top Terminal Core Requirements

The resources as specified for all OCAP profiles in the OCAP HOST2.0 [3] are REQUIRED.

This is a place holder for reference to DVR CFR.

Annex A Application Recording Description (Normative)

This annex defines requirements for application recording.

A.1 DVB-GEM Specification Correspondence

This Annex corresponds to A088 MHP PVR/PDR Common Core Specification [7], Annex A, as follows:

Table A-1 Correlation between OCAP 1.0 and A088 MHP PVR/PDR Common Core Specification

OCAP Compliance	A088 MHP PVR/PDR Common Core Specification Section	GEM Compliance
Annex A (normative) Application Recording Description	Annex A (normative) Application recording description	Complete compliance

This page intentionally left blank.

Annex B Responsibilities of this Specification (Informative)

This Annex complies with A088 MHP PVR/PDR Common Core Specification [7], Annex B (informative), Responsibilities of GEM Recording Specifications. The responsibilities called out in the referenced Annex are defined in the following sections of this specification.

Table B-1 Mapping for GEM Required Responsibilities

	GEM Requirement	OCAP DVR Spec Section
1	Which types of streams are to be considered as "recordable streams". Stream types corresponding to clauses 7.2.1 ("Audio") and 7.2.2 ("Video") of GEM in the GEM terminal specification on which the GEM recording specification is based, must be considered as "recordable streams".	6.2.1.1.3.1 "Identifying streams to be recorded"
2	"Mechanisms for resolving conflicts between requested recordings (e.g., use of the tuner).	6.2.1.1.6 "Resource Management for recording requests"
3	"Minimum capabilities for the number of streams (or number of streams of each type) that a GEM recording terminal must be able to record.	11.2.1.4 "Recording multiple streams from the same service"
4	"The definition of which applications are recordable in both scheduled and time-shift recording (which need not be the same).	6.2.1.1.3.2 "Identifying and recording applications" and 6.2.1.3.2.2 "Identifying and recording applications"
5	"Requirements on a GEM recording terminal to monitor for dynamic data (in the DSMCC object carousel or GEM functional equivalent), during scheduled and time-shift recording (which need not be the same).	6.2.1.1.3.2 "Identifying and recording applications" and 6.2.1.3.2.2 "Identifying and recording applications"
6	Requirements on a GEM recording terminal to monitor for GEM triggers or DSMCC stream events during scheduled and time-shift recording (which need not be the same).	6.2.1.1.3.2 "Identifying and recording applications" and 6.2.1.3.2.2 "Identifying and recording applications"
7	Requirements on a GEM recording terminal to monitor for dynamic application signaling during scheduled and time-shift recording (which need not be the same).	6.2.1.1.3.2 "Identifying and recording applications" and 6.2.1.3.2.2 "Identifying and recording applications"
8	"Requirements on reconstructing the dynamic behavior of recorded applications during playback of scheduled and time-shift recordings (which need not be the same).	6.2.1.1.3.2 "Identifying and recording applications" and 6.2.1.3.2.2 "Identifying and recording applications".
9	"How accurately the expiration period should be enforced by implementations.	6.2.1.1.4 "Managing completed recordings"
10	"The definition of at least one protocol for transmitted time lines.	6.2.1.1.3.1 "Identifying streams to be recorded"
11	The conditions when a JMF player or service context has a time-shift buffer attached	6.2.1.3 "Time-Shift Buffer"
12	A mechanism to associate security attributes with individual recording requests. A mapping from that mechanism to the language in each of the methods in the following table relating to "RecordingRequest specific security attributes".	10.2.1.1 "Access scope of recordings" Table 11, "Security restrictions for individual recording requests," on page 33

This page intentionally left blank.

Annex C External References; Errata, Clarifications, and Exemptions (Normative)

This annex defines clarifications for JMF usage.

C.1 DVB-GEM Specification Correspondence

This Annex corresponds to A088 MHP PVR/PDR Common Core Specification [7], Annex C, as follows:

Table C-1 Correlation between OCAP 1.0 and A088 MHP PVR/PDR Common Core Specification

OCAP Compliance	A088 MHP PVR/PDR Common Core Specification Section	GEM Compliance
Annex C (normative) External references; errata, clarifications, and exemptions	Annex C (normative) External references; errata, clarifications, and exemptions	Complete compliance

This page intentionally left blank.

Annex D OCAP DVR API (org.ocap.dvr)

This section presents the `org.ocap.dvr` API.

Package org.ocap.dvr

Class Summary	
Interfaces	
<code>OcapRecordedService</code>	This interface represents a <code>RecordedService</code> in OCAP.
<code>OcapRecordingRequest</code>	This interface represents a <code>LeafRecordingRequest</code> in OCAP.
<code>RecordingAlertListener</code>	Listener for Recording Alerts.
<code>RecordingResourceUsage</code>	This interface represents a grouping of resources specific to an recording function performed by an application.
<code>RequestResolutionHandler</code>	This interface will be implemented by the application that registers the <code>RequestResolutionHandler</code> .
<code>TimeShiftBufferResourceUsage</code>	This interface represents a grouping of resources specific to a time-shift buffering performed by an application.
Classes	
<code>OcapRecordingManager</code>	<code>RecordingManager</code> represents the entity that performs recordings.
<code>OcapRecordingProperties</code>	Encapsulates the details about how a recording is to be made.
<code>PrivateRecordingSpec</code>	Specifies a recording request that can be resolved only by an application defined request resolution handler.
<code>RecordingAlertEvent</code>	Event notifying that a scheduled recording is about to occur.

org.ocap.dvr

OcapRecordedService

Declaration

```
public interface OcapRecordedService
```

Description

This interface represents a RecordedService in OCAP. The object returned when an applications calls the getService method on a RecordingRequest will be an instance of this interface.

Member Summary

Methods

long	getRecordedBitRate()	Get the bit-rate used for encoding and storage of this recorded service.
long	getRecordedSize()	Gets the size of the recording in bytes.
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

getRecordedBitRate()

```
public long getRecordedBitRate()
```

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

Returns: Bit-rate in bytes per second.

getRecordedSize()

```
public long getRecordedSize()
```

Gets the size of the recording in bytes.

Returns: Space occupied by the recording in bytes.

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 OcapRecordingManager

Declaration

```
public abstract class OcapRecordingManager extends RecordingManager
```

```
java.lang.Object
|
|--RecordingManager
|
|--org.ocap.dvr.OcapRecordingManager
```

Description

RecordingManager represents the entity that performs recordings. An instance of this class is returned when an application calls the RecordingManager.getInstance() class in OCAP platforms.

Member Summary

Constructors

```
OcapRecordingManager ()
```

Methods

```
abstract 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.

abstract 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.

abstract ResourceUsage[] getPrioritizedResourceUsages (RecordingRequest recording)
    Get the prioritized list of overlapping ResourceUsages corresponding to a
    particular recording request.

abstract void removeRecordingAlertListener (RecordingAlertListener ral)
    Removes a register event listener for receiving recording events.

abstract RecordingRequest resolve (RecordingRequest request, RecordingSpec spec, int
    resolutionState)
    Schedule a child recording request corresponding to an unresolved or partially
    resolved recording request.

abstract void setPrioritization (ResourceUsage[] resourceUsageList)
    Sets the relative priorities for a set of ResourceUsages.

abstract void setRequestResolutionHandler (RequestResolutionHandler rrh)
    Set the RequestResolutionHandler that will be invoked when any application
    calls the RecordingManager.record method.

abstract void setSpaceAllocationHandler (SpaceAllocationHandler sah)
    Set the SpaceAllocationHandler that will be invoked when any application
    attempts to allocate space in any MediaStorageVolume.
```

Constructors

OcapRecordingManager()

```
public OcapRecordingManager()
```

Methods

addRecordingAlertListener(RecordingAlertListener)

```
public abstract 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.

Parameters:

ral - The listener to be registered.

addRecordingAlertListener(RecordingAlertListener, long)

```
public abstract 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.

Parameters:

ral - The listener to be registered.

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

getPrioritizedResourceUsages(RecordingRequest)

```
public abstract ResourceUsage[] getPrioritizedResourceUsages(RecordingRequest
    recording)
```

Get the prioritized list of overlapping ResourceUsages corresponding to a particular recording request. The list of resource usages may include RecordingResourceUsages and other types of ResourceUsages. The ResourceUsage corresponding to the specified recording request is also included in the prioritized list. The prioritized list is sorted in descending order of prioritization. The prioritization for resource usages are based on the order specified by the ResourceContentionHandler or based on the order specified by a previous call to the setPrioritization() call. If a ResourceContentionHandler is not registered or the setResourcePriorities() was not called previously, the prioritization order will be based on application priorities of the applications returned by the getAppID() call on the ResourceUsages.

Parameters:

recording - the RecordingRequest for which overlapping resource usages are sought.

Returns: the list of ResourceUsages overlapping with the specified RecordingRequest, including the ResourceUsage corresponding to the specified RecordingRequest, sorted in descending order of prioritization, null if the RecordingRequest is not in one of the pending or in-progress states.

Throws:

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

removeRecordingAlertListener(RecordingAlertListener)

```
public abstract 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.

resolve(RecordingRequest, RecordingSpec, int)

```
public abstract RecordingRequest resolve(RecordingRequest request, RecordingSpec spec, int resolutionState)
```

Schedule a child recording request corresponding to an unresolved or partially resolved recording request. This method is called either by the `RequestResolutionHandler` or by an application that has enough information to provide request resolutions. The implementation shall generate a recording request corresponding to each successful invocation of this method and make that recording request a child of the `RecordingRequest` passed in as the first parameter. If the implementation has enough information to resolve the newly created recording request, the implementation should resolve the recording request.

Implementation should set the state of the recording request "request" to "resolutionState" before the return of this call.

Parameters:

`request` - the `RecordingRequest` for which the resolution is provided.

`spec` - the `RecordingSpec` for the child recording request.

`resolutionState` - the state of the `RecordingRequest` after the return of this method. The possible values for this parameter are the states defined in `ParentRecordingRequest`.

Returns: the newly scheduled recording request.

Throws:

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

`java.lang.IllegalArgumentException` - if the `resolutionState` is not a state defined in `ParentRecordingRequest`, or if the request is not in unresolved or partially resolved state.

setPrioritization(ResourceUsage[])

```
public abstract void setPrioritization(ResourceUsage[] resourceUsageList)
```

Sets the relative priorities for a set of `ResourceUsages`. This method may be used by an application with `MonitorAppPermission("handler.recording")` to set the relative priorities for a set of overlapping resource usages. The implementation should use the specified prioritization scheme to resolve conflicts (resource conflicts as well as conflicts for `RecordingRequests`) between these overlapping resource usages. This call may change the relative priorities specified by the contention handler or a previous

call to this method. Changing the relative priorities for the resource usages may result in one or more recording requests changing states.

Parameters:

`resourceUsageList` - a list of `ResourceUsages` sorted in descending order of prioritization

Throws:

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

setRequestResolutionHandler(RequestResolutionHandler)

```
public abstract void  
    setRequestResolutionHandler(org.ocap.dvr.RequestResolutionHandler rrh)
```

Set the `RequestResolutionHandler` that will be invoked when any application calls the `RecordingManager.record` method. 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:

`rrh` - the request resolution handler.

Throws:

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

setSpaceAllocationHandler(SpaceAllocationHandler)

```
public abstract void setSpaceAllocationHandler(SpaceAllocationHandler sah)
```

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

Parameters:

`sah` - the space reservation handler.

Throws:

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

org.ocap.dvr OcapRecordingProperties

Declaration

```
public class OcapRecordingProperties extends RecordingProperties
```

```
java.lang.Object
|
|--RecordingProperties
|   |
|   |--org.ocap.dvr.OcapRecordingProperties
```

Description

Encapsulates the details about how a recording is to be made.

Member Summary

Fields

static byte	HIGH_BIT_RATE	Indicates an implementation specific value for high bit-rate.
static byte	LOW_BIT_RATE	Indicates an implementation specific value for low bit-rate.
static byte	MEDIUM_BIT_RATE	Indicates an implementation specific value for medium bit-rate.
static byte	RECORD_IF_NO_CONFLICTS	Record only if there are no conflicts.
static byte	RECORD_WITH_CONFLICTS	Record even when resource conflicts exist.
static byte	TEST_RECORDING	Schedule only test recording requests corresponding to this spec.

Constructors

```
OcapRecordingProperties(byte bitRate, long
expirationPeriod, byte priorityFlag,
ExtendedFileAccessPermissions access,
java.lang.String organization, MediaStorageVolume
destination)
    Constructor
```

Methods

ExtendedFileAccessPermissions	getAccessPermissions()	Return the file access permission to use for the recording
byte	getBitRate()	Return the bitRate to use for the recording
MediaStorageVolume	getDestination()	Return the volume that represents the storage location of the recording
java.lang.String	getOrganization()	Return the name of the organization that this recording will be tied to

Member Summary
<pre>byte getPriorityFlag()</pre> <p>Return whether or not the recording should be made if there are resource conflicts</p>

Fields

HIGH_BIT_RATE

```
public static final byte HIGH_BIT_RATE
```

Indicates an implementation specific value for high bit-rate.

LOW_BIT_RATE

```
public static final byte LOW_BIT_RATE
```

Indicates an implementation specific value for low bit-rate.

MEDIUM_BIT_RATE

```
public static final byte MEDIUM_BIT_RATE
```

Indicates an implementation specific value for medium 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 constructor for instances of this class, the recording request in PENDING_NO_CONFLICT_STATE is scheduled only if there are no conflicts. If there are conflicts the record method will create and schedule a recording request in PENDING_WITH_CONFLICT_STATE.

RECORD_WITH_CONFLICTS

```
public static final byte RECORD_WITH_CONFLICTS
```

Record even when resource conflicts exist. This value could be used as the priorityFlag parameter value to the constructor for instances of this class. When this is used as a priority value, if the request conflicts with another RecordingRequest, resources are resolved using recording resource conflict resolution rules.

TEST_RECORDING

```
public static final byte TEST_RECORDING
```

Schedule only test recording requests corresponding to this spec. This value could be used as the priorityFlag parameter value to the constructor for instances of this class. When an OcapRecordingProperties with this value used as a priority value is used to schedule a recording request, any leaf recording requests scheduled will be in the TEST_STATE. If a test recording request is unresolved, partially resolved or completely resolved, the states would be UNRESOLVED_STATE, PARTIALLY_RESOLVED_STATE and COMPLETELY_RESOLVED_STATE respectively. Test recording requests maybe used by applications to detect potential conflicts before scheduling a regular recording. Scheduling a test recording request will not affect the states of any other recording requests.

No events will be generated corresponding to a test recording request. Test recording requests will not change state to any other state.

Constructors

OcapRecordingProperties(byte, long, byte, ExtendedFileAccessPermissions, String, MediaStorageVolume)

```
public OcapRecordingProperties(byte bitRate, long expirationPeriod,
    byte priorityFlag, ExtendedFileAccessPermissions access,
    java.lang.String organization, MediaStorageVolume destination)
    throws IllegalArgumentException
```

Constructor

Parameters:

`bitRate` - 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 trans-rating. When trans-rating is supported, HIGH_BIT_RATE indicates no trans-rating, and MEDIUM_BIT_RATE to LOW_BIT_RATE indicates increasing compression with a potential decrease in video quality.

`expirationPeriod` - The period in seconds after the initiation of recording when leaf recording requests with this recording property are deemed as expired. The implementation will delete off recorded services associated with any expired recording requests without application intervention and transition those recording requests to deleted state.

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

`access` - File access permission for the recording request.

`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. When an instance of this class is used with a ServiceRecordingSpec a LocatorRecordingSpec, or a ServiceContextRecordingSpec where the specified service context is not attached to a time-shift buffer, with the value of this parameter set to null, the implementation shall use the default recording volume (see org.ocap.storage.MediaStorageOption) in one of the storage devices connected. If the value is null when used with a ServiceContextRecordingSpec, when the service context specified in the ServiceContextRecordingSpec is attached to a time-shift buffer, the default recording volume from the storage device where the time-shift buffer is located shall be used. When an instance of this class is used with a ServiceContextRecordingSpec, the record(..) method will throw an IllegalArgumentException if the destination is not in same storage device where an attached time-shift buffer is located.

Throws:

`java.lang.IllegalArgumentException` - if `bitRate` 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, TEST_RECORDING or RECORD_WITH_CONFLICTS, or if `organization` is not found in the application's certificate file.

Methods

getAccessPermissions()

```
public ExtendedFileAccessPermissions getAccessPermissions()
```

Return the file access permission to use for the recording

Returns: the file access permission passed into the constructor

getBitRate()

```
public byte getBitRate()
```

Return the bitRate to use for the recording

Returns: the bitRate as passed into the constructor

getDestination()

```
public MediaStorageVolume getDestination()
```

Return the volume that represents the storage location of the recording

Returns: the volume passed into the constructor

getOrganization()

```
public java.lang.String getOrganization()
```

Return the name of the organization that this recording will be tied to

Returns: the organization passed into the constructor

getPriorityFlag()

```
public byte getPriorityFlag()
```

Return whether or not the recording should be made if there are resource conflicts

Returns: the priority flag passed into the constructor

org.ocap.dvr

OcapRecordingRequest

Declaration

```
public interface OcapRecordingRequest
```

Description

This interface represents a LeafRecordingRequest in OCAP.

When the implementation detects a schedule conflict, it either resolves the conflict using the Application priority of the conflicting recordings, 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 recordings to be pending without conflict and some to be pending with conflict.

Member Summary

Fields

```
static int TEST_STATE
    This recording request is a test recording request.
```

Methods

```
RecordingList getOverlappingEntries()
    Gets any other RecordingRequest that overlaps with the duration of this
    recording request.
long getSpaceRequired()
    Gets the estimated space, in bytes, required for the recording.
```

Fields

TEST_STATE

```
public static final int TEST_STATE
```

This recording request is a test recording request. Actual recording is not initiated for recording requests in this state. RecordingRequests in this state do not transition to other states. No events are generated when a recording request is added or deleted in this state. A recording in this state is a leaf recording request. Recordings in this states are the leaf recording requests corresponding to invocation of the `RecordingManager.record(..)` method with the priority value in `OcapRecordingProperties` set to `TEST_RECORDING`.

Methods

getOverlappingEntries()

```
public RecordingList getOverlappingEntries()
```

Gets any other RecordingRequest that overlaps with the duration of this recording request. This method will return null unless the recording request is in the PENDING_WITH_CONFLICTS_STATE, PENDING_NO_CONFLICTS_STATE, IN_PROGRESS_INSUFFICIENT_SPACE_STATE or IN_PROGRESS_STATE. The returned list will contain only overlapping recording requests for which the application has read access permission. The 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.

Returns: a RecordingList

getSpaceRequired()

```
public long getSpaceRequired()
```

Gets the estimated space, in bytes, required for the recording.

Returns: Space required for the recording in bytes. This method returns zero if the recordings is in failed state.

org.ocap.dvr PrivateRecordingSpec

Declaration

```
public class PrivateRecordingSpec extends RecordingSpec
```

```
java.lang.Object
|
+--RecordingSpec
|
+--org.ocap.dvr.PrivateRecordingSpec
```

Description

Specifies a recording request that can be resolved only by an application defined request resolution handler.

Member Summary	
Constructors	
	<pre>PrivateRecordingSpec(java.io.Serializable requestData, RecordingProperties properties) Constructor</pre>
Methods	
	<pre>java.io.Serializable getPrivateData() Returns the private data stored in this recording spec</pre>

Constructors

PrivateRecordingSpec(Serializable, RecordingProperties)

```
public PrivateRecordingSpec(java.io.Serializable requestData,
RecordingProperties properties)
```

Constructor

Parameters:

`requestData` - private data the format of which is known only to the application.

`properties` - the definition of how the recording is to be done

Methods

getPrivateData()

```
public java.io.Serializable getPrivateData()
```

Returns the private data stored in this recording spec

Returns: the private data passed into the constructor

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 LeafRecordingRequests in pending states.

Member Summary	
Constructors	RecordingAlertEvent(RecordingRequest source) Constructs the event.
Methods	RecordingRequest getRecordingRequest() Returns the RecordingRequest that caused the event.

Constructors

RecordingAlertEvent(RecordingRequest)

```
public RecordingAlertEvent(RecordingRequest source)
```

Constructs the event.

Parameters:

source - The RecordingRequest that caused the event.

Methods

getRecordingRequest()

```
public RecordingRequest getRecordingRequest()
```

Returns the RecordingRequest that caused the event.

Returns: The RecordingRequest 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

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

<code>RecordingRequest</code>	<code>getRecordingRequest()</code>	Gets the <code>RecordingRequest</code> associated with the set of resources contained in the usage and initiated by the application returned by the base <code>ResourceUsage.getAppID</code> method.
-------------------------------	------------------------------------	--

Methods

`getRecordingRequest()`

```
public RecordingRequest getRecordingRequest()
```

Gets the `RecordingRequest` 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 request associated with the resource usage.

org.ocap.dvr RequestResolutionHandler

Declaration

```
public interface RequestResolutionHandler
```

Description

This interface will be implemented by the application that registers the RequestResolutionHandler. The RequestResolutionHandler will be invoked whenever a new unresolved recording request is added to the RecordingManager database. The RecordingResolutionHandler may call the resolve(..) method of the OcapRecordingManager multiple times to schedule one or more recording requests corresponding to the recording request.

Member Summary

Methods

```
void requestResolution(RecordingRequest request)
```

This method would be invoked by the implementation when an unresolved recording request is scheduled in response to an application calling the record(..) method of the RecordingManager.

Methods

requestResolution(RecordingRequest)

```
public void requestResolution(RecordingRequest request)
```

This method would be invoked by the implementation when an unresolved recording request is scheduled in response to an application calling the record(..) method of the RecordingManager.

org.ocap.dvr TimeShiftBufferResourceUsage

Declaration

```
public interface TimeShiftBufferResourceUsage
```

Description

This interface represents a grouping of resources specific to a time-shift buffering performed by an application.

Member Summary

Methods

<pre>TimeShiftBufferOption getTimeShiftBufferOption()</pre>	<p>Gets the <code>TimeShiftBufferOption</code> associated with the set of resources contained in the usage where the last service selection was initiated by the application returned by the base <code>ResourceUsage.getAppID</code> method.</p>
---	---

Methods

`getTimeShiftBufferOption()`

```
public TimeShiftBufferOption getTimeShiftBufferOption()
```

Gets the `TimeShiftBufferOption` associated with the set of resources contained in the usage where the last service selection was initiated by the application returned by the base `ResourceUsage.getAppID` method.

Returns: The time-shift buffer associated with the resource usage.

This page intentionally left blank.

Annex E OCAP DVR Storage API (org.ocap.dvr.storage)

This section presents the `org.ocap.dvr.storage` API.

Package

org.ocap.dvr.storage

Class Summary	
Interfaces	
<code>AllocateTimeShiftBufferOption</code>	This interface represents an option that can be contained within a <code>StorageProxy</code> and can be used to allocate a time-shift buffer.
<code>MediaStorageOption</code>	This interface represents an option object provided by a <code>org.ocap.storage.StorageProxy</code> that supports media volumes (<code>MediaStorageVolume</code>) that are used by the DVR recording and playback APIs for storing media content.
<code>MediaStorageVolume</code>	This interface represents a media volume on a storage device and is contained within a <code>org.ocap.storage.StorageProxy</code> .
<code>SpaceAllocationHandler</code>	A class implementing this interface decides whether requests to allocate storage space should be allowed or not.
<code>TimeShiftBufferListener</code>	The <code>TimeShiftBufferListener</code> interface is implemented by applications wishing to receive events related to <code>TimeShiftBuffers</code> .
<code>TimeShiftBufferOption</code>	This interface represents a time-shift buffer that can be used to buffer and apply trick modes to a live multi-media stream.
Classes	
<code>TimeShiftBufferEvent</code>	The parent class for <code>TimeShiftBuffer</code> events.
<code>TimeShiftBufferSelectedEvent</code>	<code>TimeShiftBufferSelectedEvent</code> event is generated to indicate that the broadcast service passed as input is being recorded.
<code>TimeShiftBufferSelectionFailedEvent</code>	<code>TimeShiftBufferSelectionFailedEvent</code> event is generated when a <code>TimeShiftBuffer</code> selection method fails.
<code>TimeShiftBufferTerminatedEvent</code>	A <code>TimeShiftBufferTerminatedEvent</code> is generated when the recording of a service terminates.

org.ocap.dvr.storage

AllocateTimeShiftBufferOption

Declaration

```
public interface AllocateTimeShiftBufferOption
```

Description

This interface represents an option that can be contained within a StorageProxy and can be used to allocate a time-shift buffer. This option will only be contained within StorageProxy objects that also contain a MediaStorageOption 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.dvr.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 `org.ocap.storage.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 Storage manager.

`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 `MediaStorageOption` `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.dvr.storage

MediaStorageOption

Declaration

```
public interface MediaStorageOption
```

Description

This interface represents an option object provided by a `org.ocap.storage.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>org.ocap.storage.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.

Member Summary

```

void initialize(long mediafsSize)
    Initializes the storage device so that there are at least mediafsSize bytes
    available for MEDIAFS use.
boolean simultaneousPlayAndRecord()
    Indicates if the storage device supports simultaneous play and record.

```

Methods

allocateMediaVolume(String, ExtendedFileAccessPermissions)

```

public org.ocap.dvr.storage.MediaStorageVolume allocateMediaVolume(java.lang.String
    name, 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.dvr.storage.MediaStorageVolume getDefaultRecordingVolume()

```

Gets the default volume that the implementation setup as the default recording volume for the containing `org.ocap.storage.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 mediafsSize passed is greater than the sum of what is returned by getTotalGeneralStorageCapacity() and getTotalMediaStorageCapacity().

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.dvr.storage MediaStorageVolume

Declaration

```
public interface MediaStorageVolume
```

Description

This interface represents a media volume on a storage device and is contained within a `org.ocap.storage.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

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

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 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.

Throws:

`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 this 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 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 allocated space has been used, this method returns the same value as the `getAllocatedSpace` method. When this method is called on a `MediaStorageVolume` without an explicit allocation, as is the case when `allocate` has not been called or was called with a value of 0, then the value returned is the space available on the associated `StorageProxy`'s `MEDIAFS` that has not been explicitly allocated to another `MediaStorageVolume` or `TimeShiftBufferOption`.

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.dvr.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.dvr.storage TimeShiftBufferEvent

Declaration

```
public class TimeShiftBufferEvent extends java.util.EventObject
    java.lang.Object
    |
    |-- java.util.EventObject
    |   |-- org.ocap.dvr.storage.TimeShiftBufferEvent
```

All Implemented Interfaces: java.io.Serializable

Direct Known Subclasses: TimeShiftBufferSelectedEvent,
TimeShiftBufferSelectionFailedEvent, TimeShiftBufferTerminatedEvent

Description

The parent class for TimeShiftBuffer events.

Member Summary	
Constructors	<pre>TimeShiftBufferEvent (TimeShiftBufferOption source) Constructor for this event.</pre>
Methods	<pre>TimeShiftBufferOption getTimeShiftBuffer () Reports the TimeShiftBuffer that generated the event.</pre>

Constructors

TimeShiftBufferEvent(TimeShiftBufferOption)

```
public TimeShiftBufferEvent (org.ocap.dvr.storage.TimeShiftBufferOption source)
    Constructor for this event.
```

Parameters:

source - The TimeShiftBuffer that generated this event.

Methods

getTimeShiftBuffer()

```
public org.ocap.dvr.storage.TimeShiftBufferOption getTimeShiftBuffer ()
    Reports the TimeShiftBuffer that generated the event.
```

Returns: a TimeShiftBufferOption

org.ocap.dvr.storage TimeShiftBufferListener

Declaration

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

All Superinterfaces: java.util.EventListener

Description

The TimeShiftBufferListener interface is implemented by applications wishing to receive events related to TimeShiftBuffers.

Member Summary

Methods

```
void receiveTimeShiftBufferEvent (TimeShiftBufferEvent e)  
    Notifies the TimeShiftBufferListener of an event generated by a  
    TimeShiftBufferOption.
```

Methods

receiveTimeShiftBufferEvent(TimeShiftBufferEvent)

```
public void receiveTimeShiftBufferEvent (org.ocap.dvr.storage.TimeShiftBufferEvent e)
```

Notifies the TimeShiftBufferListener of an event generated by a TimeShiftBufferOption.

Parameters:

e - The generated event.

org.ocap.dvr.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.

A time-shift buffer could either be in the attached state or the detached state.

When a time-shift buffer is not attached to any service context, the time-shift buffer is in the detached state. If an application calls the select method on a time-shift buffer in detached state, the time-shift buffer would start buffering the selected broadcast service without presenting the selected service.

If the time-shift buffer is attached to a service context, the time-shift buffer is in the attached state. When in the attached state, time-shift buffer would buffer the broadcast service that is selected on the attached service context. While in the attached state, the select method or the stop method will throw `IllegalStateException`.

If the service context to which the time-shift buffer is attached is destroyed the time-shift buffer returns to the detached state. Any circular recording will continue without interruption.

Member Summary

Methods

<code>void</code>	<code>addListener (TimeShiftBufferListener listener)</code>	Adds a listener to receive events related to this time-shift buffer.
<code>void</code>	<code>attach (javax.tv.service.selection.ServiceContext serviceContext)</code>	Attaches the time-shift buffer to a <code>javax.tv.service.selection.ServiceContext</code> .
<code>void</code>	<code>detach ()</code>	Detaches the time-shift buffer from a <code>ServiceContext</code> .
<code>long</code>	<code>getMinimumBufferSize ()</code>	Gets the implementation specific minimum time-shift buffer size for this storage device.
<code>long</code>	<code>getName ()</code>	Gets the name passed in as the argument to the method <code>AllocateTimeShiftBufferOption.allocateMediaBuffer(..)</code> that created this time shift buffer.
<code>javax.tv.service.Service</code>	<code>getService ()</code>	Reports the Service being recorded in this time-shift buffer.
<code>javax.tv.service.selection.ServiceContext</code>	<code>getServiceContext ()</code>	Gets the <code>ServiceContext</code> the time-shift buffer is currently attached to.
<code>StorageProxy</code>	<code>getStorageProxy ()</code>	Gets a <code>StorageProxy</code> associated with this timeshift buffer.
<code>long</code>	<code>getTotalCapacity ()</code>	Gets the total capacity of the time-shift buffer.

Member Summary

<pre> boolean isAttached() void removeListener(TimeShiftBufferListener listener) long resize(long newSize) void select(javax.tv.service.Service service) void stop() </pre>	<pre> Check if this time-shift buffer is attached to a ServiceContext. Removes a listener from receiving events related to this time-shift buffer. Resizes the time-shift buffer. Selects a service to be recorded within this circular time shift buffer. Causes the time-shift buffer to stop recording the selected service and enter the not recording state. </pre>
--	--

Methods

addListener(TimeShiftBufferListener)

```
public void addListener(org.ocap.dvr.storage.TimeShiftBufferListener listener)
```

Adds a listener to receive events related to this time-shift buffer. If the specified listener is already added, no action is performed.

Parameters:

`listener` - The TimeShiftBufferListener to add.

Throws:

`java.lang.IllegalStateException` - - If the time-shift buffer has been destroyed.

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.

If the time shift buffer is currently recording, the recording continues without interruptions.

Parameters:

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

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, if the time-shift buffer is already attached to a ServiceContext different from the parameter, or if another time-shift buffer is already attached to the ServiceContext.

detach()

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

Detaches the time-shift buffer from a ServiceContext. If not attached this method returns silently. If the time shift buffer is currently recording, the recording continues without interruptions.

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.

getName()

```
public long getName()
```

Gets the name passed in as the argument to the method `AllocateTimeShiftBufferOption.allocateMediaBuffer(..)` that created this time shift buffer.

Returns: the name of the time-shift buffer.

getService()

```
public javax.tv.service.Service getService()
```

Reports the Service being recorded in this time-shift buffer. If the time-shift buffer is currently recording a service the Service returned will be a representation of the Service indicated in the last successful `select()` method call on the time shift buffer or any attached service context. If the time-shift buffer is not recording a service then null is returned.

Returns: The service being recorded; null if the time-shift buffer is not currently recording.

Throws:

`java.lang.IllegalStateException` - - If the time-shift buffer has been destroyed.

getServiceContext()

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

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

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

getStorageProxy()

```
public StorageProxy getStorageProxy()
```

Gets a StorageProxy associated with this timeshift buffer.

Returns: the associated storage proxy.

getTotalCapacity()

```
public long getTotalCapacity()
```

Gets the total capacity of the time-shift buffer.

Returns: Total capacity in kilo-bytes.

isAttached()

```
public boolean isAttached()
```

Check if this time-shift buffer is attached to a ServiceContext.

Returns: true if a ServiceContext is attached and false otherwise.

removeListener(TimeShiftBufferListener)

```
public void removeListener(org.ocap.dvr.storage.TimeShiftBufferListener listener)
```

Removes a listener from receiving events related to this time-shift buffer. If the specified listener is not added, no action is performed.

Parameters:

`listener` - The TimeShiftBufferListener to remove.

Throws:

`java.lang.IllegalStateException` - - If the time-shift buffer has been destroyed.

resize(long)

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

Resizes the time-shift buffer. 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, or if the time-shift buffer is actively buffering a service.

select(Service)

```
public void select(javax.tv.service.Service service)
```

Selects a service to be recorded within this circular time shift buffer. Selecting a service on a time-shift buffer initiates the circular recording of the service on the time shift buffer. Presentation of the service is not initiated as a result of this call.

Presentation of what is being recorded in the time-shift buffer is done through a service context attached to it. Attaching or detaching a ServiceContext does not affect recording of the Service. Each invocation of this method will initiate the recording of the newly selected service.

This method can be called only on a time-shift buffer that is not attached to a service context. For a time-shift buffer that is attached to a service context, the select call on the service context will initiate recording of the service newly selected on the service context.

After a time-shift buffer in recording state is attached to a service context, if an application selects the service being recorded on the time-shift buffer on the service context, the circular time shift buffer will not be flushed and the presentation of the selected service will be initiated on the service context from the beginning of the service on the buffer.

The time-shift buffer will attempt to acquire all necessary resource required for the starting the circular recording of the selected service.

If a call to this method completes successfully a *TimeShiftBufferSelectedEvent* event is generated. If the selection operation fails a *TimeShiftBufferSelectionFailedEvent* event is generated. Once selected if the recording by a time-shift buffer is terminated (due to loss of resources or any other reason) a *TimeShiftBufferTerminatedEvent* event is generated.

Parameters:

`service` - The service to be selected.

Throws:

`java.lang.IllegalStateException` - - If the time shift buffer has been destroyed or if the time shift buffer is already attached.

`java.lang.IllegalArgumentException` - - If the service to be selected does not refer to a broadcast channel.

stop()

```
public void stop()
```

Causes the time-shift buffer to stop recording the selected service and enter the not recording state. Resources used for recording will be released, and a *TimeShiftBufferTerminatedEvent* will be posted. This operation completes asynchronously.

No action is performed if the time-shift buffer is already in the not recording state.

Throws:

`java.lang.IllegalStateException` - - If the time shift buffer has been destroyed or if the time shift buffer is already attached.

org.ocap.dvr.storage TimeShiftBufferSelectedEvent

Declaration

```
public class TimeShiftBufferSelectedEvent extends TimeShiftBufferEvent
```

```
java.lang.Object
|
+--java.util.EventObject
    |
    +--org.ocap.dvr.storage.TimeShiftBufferEvent
        |
        +--org.ocap.dvr.storage.TimeShiftBufferSelectedEvent
```

All Implemented Interfaces: java.io.Serializable

Description

TimeShiftBufferSelectedEvent event is generated to indicate that the broadcast service passed as input is being recorded. This event is generated only if the associated TimeShiftBufferOption is not attached.

Member Summary

Constructors

```
TimeShiftBufferSelectedEvent (TimeShiftBufferOption
source)
    Constructor for this event.
```

Constructors

TimeShiftBufferSelectedEvent(TimeShiftBufferOption)

```
public TimeShiftBufferSelectedEvent (org.ocap.dvr.storage.TimeShiftBufferOption
source)
```

Constructor for this event.

Parameters:

source - The TimeShiftBufferOption that generated this event.

org.ocap.dvr.storage TimeShiftBufferSelectionFailedEvent

Declaration

```
public class TimeShiftBufferSelectionFailedEvent extends TimeShiftBufferEvent
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--org.ocap.dvr.storage.TimeShiftBufferEvent
|
+--org.ocap.dvr.storage.TimeShiftBufferSelectionFailedEvent
```

All Implemented Interfaces: java.io.Serializable

Description

TimeShiftBufferSelectionFailedEvent event is generated when a TimeShiftBuffer selection method fails. This event is not generated when the selection method fails with an exception.

Member Summary	
Fields	
static int	CA_REFUSAL Reason code: Selection failed due to the CA system refusing to permit it.
static int	CONTENT_NOT_FOUND Reason code: Selection failed because the requested content could not be found in the network.
static int	INSUFFICIENT_RESOURCES Reason code: Selection failed due to a lack of resources required to present this service.
static int	INTERRUPTED Reason code: Selection has been interrupted by another selection request.
static int	TUNING_FAILURE Reason code: Selection failed due to problems with tuning.
Constructors	
	TimeShiftBufferSelectionFailedEvent (TimeShiftBufferOption source, int reason) Constructor for this event.
Methods	
int	getReason() Returns the reason why selection failed.

Fields

CA_REFUSAL

```
public static final int CA_REFUSAL
```

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

CONTENT_NOT_FOUND

```
public static final int CONTENT_NOT_FOUND
```

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

INSUFFICIENT_RESOURCES

```
public static final int INSUFFICIENT_RESOURCES
```

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

INTERRUPTED

```
public static final int INTERRUPTED
```

Reason code: Selection has been interrupted by another selection request.

TUNING_FAILURE

```
public static final int TUNING_FAILURE
```

Reason code: Selection failed due to problems with tuning.

Constructors

TimeShiftBufferSelectionFailedEvent(TimeShiftBufferOption, int)

```
public  
    TimeShiftBufferSelectionFailedEvent(org.ocap.dvr.storage.TimeShiftBuffer  
    Option source, int reason)
```

Constructor for this event.

Parameters:

source - The TimeShiftBufferOption that generated this event.

reason - The reason why the selection failed.

Methods

getReason()

```
public int getReason()
```

Returns the reason why selection failed.

Returns: The reason why selection failed.

org.ocap.dvr.storage TimeShiftBufferTerminatedEvent

Declaration

```
public class TimeShiftBufferTerminatedEvent extends TimeShiftBufferEvent
```

```
java.lang.Object
|
+--java.util.EventObject
|
+--org.ocap.dvr.storage.TimeShiftBufferEvent
|
+--org.ocap.dvr.storage.TimeShiftBufferTerminatedEvent
```

All Implemented Interfaces: java.io.Serializable

Description

A `TimeShiftBufferTerminatedEvent` is generated when the recording of a service terminates. This includes both normal termination (such as when an application calls the `stop()` method) and abnormal termination (as may occur when there is some change in the environment). Examples of abnormal termination include:

- a tuning operation has made the service unavailable
- the removal of fundamental resources required to present the service has made the service unavailable
- the withdrawal of CA authorization, has blocked access to the service

A `TimeShiftBufferTerminatedEvent` is also generated following a `TimeShiftBufferSelectionFailedEvent`, if the time-shift buffer was previously in the *recording* state. A `TimeShiftBufferTerminatedEvent` is only generated when no components of the requested service can be recorded.

Once this event has been generated, a `TimeShiftBufferOption` will be in the *not recording* state until a call to a `select()` method succeeds. When this event is generated, all resources used for the presentation have been released.

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	SERVICE_VANISHED	Reason code: The service vanished from the network.
static int	TUNED_AWAY	Reason code: Tuning made the service unavailable.
static int	USER_STOP	Reason code: The user requested that the presentation be stopped.

Member Summary	
Constructors	<pre>TimeShiftBufferTerminatedEvent (TimeShiftBufferOption source, int reason) Constructor for this event.</pre>
Methods	<pre>int getReason() Returns the reason why the recording was terminated.</pre>

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.

SERVICE_VANISHED

```
public static final int SERVICE_VANISHED
```

Reason code: The service vanished from the network.

TUNED_AWAY

```
public static final int TUNED_AWAY
```

Reason code: Tuning made the service unavailable.

USER_STOP

```
public static final int USER_STOP
```

Reason code: The user requested that the presentation be stopped.

Constructors

TimeShiftBufferTerminatedEvent(TimeShiftBufferOption, int)

```
public TimeShiftBufferTerminatedEvent(org.ocap.dvr.storage.TimeShiftBufferOption
source, int reason)
```

Constructor for this event.

Parameters:

source - The TimeShiftBufferOption that generated this event.

reason - The reason why the recording was terminated.

Methods

getReason()

```
public int getReason()
```

Returns the reason why the recording was terminated.

Returns: The reason why the recording was terminated.

Annex F OCAP Shared DVR API (org.ocap.shared.dvr)

This section presents the `org.ocap.shared.dvr` API.

Package org.ocap.shared.dvr

Description

The shared DVR API for scheduling and managing recording requests.

Class Summary	
Interfaces	
<code>LeafRecordingRequest</code>	This interface represents information corresponding to a leaf level recording request.
<code>ParentRecordingRequest</code>	This interface represents information corresponding to a parent recording request.
<code>RecordedService</code>	This interface represents the recorded portion of a service that is being recorded or was recorded for a period of time.
<code>RecordingChangedListener</code>	Listener to receive changes in the recording list maintained by the <code>RecordingManager</code> .
<code>RecordingRequest</code>	This interface represents information corresponding to a recording request.
Classes	
<code>DeletionDetails</code>	This class contains details about the deletion of a recorded service.
<code>LocatorRecordingSpec</code>	Specifies a recording request in terms of one or more <code>Locators</code> .
<code>RecordedServiceType</code>	This class represents the service type value for a <code>RecordedService</code> .
<code>RecordingChangedEvent</code>	Event used to notify listeners of changes in the list of recording requests maintained by the <code>RecordingManager</code> .
<code>RecordingManager</code>	<code>RecordingManager</code> represents the entity that performs recordings.
<code>RecordingPermission</code>	Controls access to recording features by an application.
<code>RecordingProperties</code>	Base class for specifying properties defining how a recording is to be made.
<code>RecordingSpec</code>	Base class for specifying what to record and how to record it.
<code>RecordingTerminatedEvent</code>	An Event Notifying that recording has terminated for the <code>ServiceContext</code> .
<code>ServiceContextRecordingSpec</code>	Specifies a recording request in terms of what is being presented on a <code>ServiceContext</code> .
<code>ServiceRecordingSpec</code>	Specifies a recording request in terms of a <code>Service</code> .
Exceptions	

Class Summary

<code>AccessDeniedException</code>	Exception thrown when an application is blocked from operating on a <code>RecordingRequest</code> by security attributes associated with that <code>RecordingRequest</code> .
<code>NoMoreDataEntriesException</code>	No more data entries allowed for this recording request.
<code>RecordingFailedException</code>	This exception is returned when applications call the <code>getFailedException()</code> for a failed recording request or an incomplete recording request.

org.ocap.shared.dvr AccessDeniedException

Declaration

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

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

All Implemented Interfaces: java.io.Serializable

Description

Exception thrown when an application is blocked from operating on a RecordingRequest by security attributes associated with that RecordingRequest.

Member Summary

Constructors

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

Constructors

AccessDeniedException()

```
public AccessDeniedException()
```

Constructs a AccessDeniedException with no detail message

org.ocap.shared.dvr DeletionDetails

Declaration

```
public class DeletionDetails
    java.lang.Object
    |
    +--org.ocap.shared.dvr.DeletionDetails
```

Description

This class contains details about the deletion of a recorded service.

Member Summary	
Fields	
static int	EXPIRED Reason code: The recorded service was deleted by the implementation because the recording request has expired.
static int	USER_DELETED Reason code: The recorded service was explicitly deleted by the application.
Methods	
java.util.Date	getDeletionTime() Gets the time when the recorded service was deleted.
int	getReason() Reports the reason for why the recorded service was deleted.

Fields

EXPIRED

```
public static final int EXPIRED
```

Reason code: The recorded service was deleted by the implementation because the recording request has expired.

USER_DELETED

```
public static final int USER_DELETED
```

Reason code: The recorded service was explicitly deleted by the application.

Methods

getDeletionTime()

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

Gets the time when the recorded service was deleted.

Returns: the deletion time.

getReason()

```
public int getReason()
```

Reports the reason for why the recorded service was deleted.

Returns: the reason code for which the recorded service was deleted.

org.ocap.shared.dvr LeafRecordingRequest

Declaration

```
public interface LeafRecordingRequest extends RecordingRequest
```

All Superinterfaces: RecordingRequest

Description

This interface represents information corresponding to a leaf level recording request. The recording request represented by this interface corresponds to a recording request that has been completely resolved to a single recording.

A leaf recording request may be pending (i.e. waiting for the start-time to occur), in-progress, completed, incomplete, or failed.

While in pending state, a recording request may be in conflict for resources with other recordings. Any such conflicts must be resolved before the scheduled start time of the recording, if not, the pending recording request is expected to result in a failed recording.

Member Summary

Fields

```
static int COMPLETED_STATE
    Recording for this recording request has completed successfully.
static int DELETED_STATE
    The recorded service corresponding to this recording request has been deleted.
static int FAILED_STATE
    The recording request has failed.
static int IN_PROGRESS_INSUFFICIENT_SPACE_STATE
    Recording has been initiated for this recording request and is ongoing, but the
    implementation has detected that storage space may not be sufficient to
    complete the recording.
static int IN_PROGRESS_STATE
    Recording has been initiated for this recording request and is ongoing.
static int INCOMPLETE_STATE
    Recording for this recording request was initiated but failed in the
    IN_PROGRESS_STATE before the COMPLETED_STATE could be reached.
static int PENDING_NO_CONFLICT_STATE
    The recording request is Pending.
static int PENDING_WITH_CONFLICT_STATE
    The recording request may not be initiated due to resource conflicts.
```

Methods

```
void cancel()
    Cancels a pending recording request.
DeletionDetails getDeletionDetails()
    Gets detailed information about the deletion of the recorded service
    corresponding to this recording request.
```

Member Summary

<pre> java.lang.Exception getFailedException() Gets the exception that caused the recording request to enter the FAILED_STATE, or INCOMPLETE_STATE. RecordedService getService() Returns the RecordedService corresponding to the recording request. void stop() Stops the recording for an in-progress recording request regardless of how much of the duration has been recorded. </pre>

Fields

COMPLETED_STATE

```
public static final int COMPLETED_STATE
```

Recording for this recording request has completed successfully.

DELETED_STATE

```
public static final int DELETED_STATE
```

The recorded service corresponding to this recording request has been deleted.

FAILED_STATE

```
public static final int FAILED_STATE
```

The recording request has failed.

IN_PROGRESS_INSUFFICIENT_SPACE_STATE

```
public static final int IN_PROGRESS_INSUFFICIENT_SPACE_STATE
```

Recording has been initiated for this recording request 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 request and other recording requests that are in progress. The recording request is not expected to complete successfully. A recording request may enter this state from IN_PROGRESS_STATE or PENDING_NO_CONFLICT_STATE. Implementation may start a recording request 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 request in IN_PROGRESS_STATE and then transition to FAILED_STATE when space runs out. It is also possible for a recording request to start in IN_PROGRESS_INSUFFICIENT_SPACE_STATE and later move to IN_PROGRESS_STATE or COMPLETED_STATE. This could occur if other recording requests were deleted after the start of recording or if implementation computes a better estimate of the space needed as the recording progress.

IN_PROGRESS_STATE

```
public static final int IN_PROGRESS_STATE
```

Recording has been initiated for this recording request and is ongoing. Recording is expected to complete successfully.

INCOMPLETE_STATE

```
public static final int INCOMPLETE_STATE
```

Recording for this recording request was initiated but failed in the IN_PROGRESS_STATE before the COMPLETED_STATE could be reached. The RecordingRequest 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 request is Pending. recording for this request is expected to complete successfully.

PENDING_WITH_CONFLICT_STATE

```
public static final int PENDING_WITH_CONFLICT_STATE
```

The recording request may not be initiated due to resource conflicts. The implementation has detected a resource conflict for the scheduled time of this recording request and the current resolution of the conflict does not allow this recording request to be initiated successfully.

Methods

cancel()

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

Cancels a pending recording request. The recording request will be deleted from the database after the successful invocation of this method. Cancelling a recording request may resolve one or more conflicts. In this case some pending recordings with conflicts would be changed to pending without conflicts.

Throws:

`AccessDeniedException` - if the calling application is not permitted to perform this operation by RecordingRequest specific security attributes.

`java.lang.SecurityException` - if the calling application does not have `RecordingPermission("cancel",...)` or `RecordingPermission("*",...)`

`java.lang.IllegalStateException` - if the state of the recording is not in `PENDING_STATE_NO_CONFLICT_STATE` or `PENDING_WITH_CONFLICT_STATE`.

getDeletionDetails()

```
public org.ocap.shared.dvr.DeletionDetails getDeletionDetails()
    throws IllegalStateException
```

Gets detailed information about the deletion of the recorded service corresponding to this recording request.

Returns: The deletion details for this recording request.

Throws:

`java.lang.IllegalStateException` - if the recording request is not in the `DELETED_STATE`.

getFailedException()

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

Gets the exception that caused the recording request 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 request is not in the `FAILED_STATE` or `INCOMPLETE_STATE`.

getService()

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

Returns the `RecordedService` corresponding to the recording request.

Returns: The recorded service associated with the recording request.

Throws:

`java.lang.IllegalStateException` - if the recording request is not in `INCOMPLETE_STATE`, `IN_PROGRESS_STATE`, `IN_PROGRESS_INSUFFICIENT_SPACE_STATE` or `COMPLETED_STATE`.

`AccessDeniedException` - if the calling application is not permitted to perform this operation by `RecordingRequest` specific security attributes.

stop()

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

Stops the recording for an in-progress recording request regardless of how much of the duration has been recorded. Moves the recording to the `INCOMPLETE_STATE`.

Throws:

`AccessDeniedException` - if the calling application is not permitted to perform this operation by `RecordingRequest` specific security attributes.

`java.lang.SecurityException` - if the calling application does not have `RecordingPermission("cancel",..)` or `RecordingPermission("*",..)`

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

org.ocap.shared.dvr LocatorRecordingSpec

Declaration

```
public class LocatorRecordingSpec extends RecordingSpec

java.lang.Object
|
+--org.ocap.shared.dvr.RecordingSpec
|
+--org.ocap.shared.dvr.LocatorRecordingSpec
```

Description

Specifies a recording request in terms of one or more Locators.

If multiple locators are contained within the source, all of them **MUST** be part of the same service.

When instances of this class are passed to `RecordingManager.record(..)`, no additional failure modes shall apply.

When an instance of this recording spec is passed in as a parameter to the `RecordingRequest.reschedule(..)` method, an `IllegalArgumentException` is thrown if the source is different from the source specified in the current recording spec for the recording request and if the recording request is in progress state.

Member Summary

Constructors

```
LocatorRecordingSpec(javax.tv.locator.Locator source,
java.util.Date startTime, long duration,
RecordingProperties properties)
    Constructor
```

Methods

```
long getDuration()
    Returns the duration passed as an argument to the constructor.

javax.tv.locator getSource()
    .Locator[] Returns the source of the recording

java.util.Date getStartTime()
    Returns the start time passed as an argument to the constructor.
```

Constructors

LocatorRecordingSpec(Locator[], Date, long, RecordingProperties)

```
public LocatorRecordingSpec(javax.tv.locator.Locator[] source,
java.util.Date startTime, long duration,
org.ocap.shared.dvr.RecordingProperties properties)
throws InvalidServiceComponentException
```

Constructor

Parameters:

`source` - Source of streams to be recorded. Implementations shall make a copy of this array before the constructor returns.

`startTime` - Start time of the recording. If the start time is in the past when the record method is called with this `LocatorRecordingSpec` as an argument, the current time is used instead and the duration reduced by the same amount. If the end time as computed as the start time + the duration is in the past when the record method is called, the record method will throw an `IllegalArgumentException`.

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

`properties` - the definition of how the recording is to be done

Throws:

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

`java.lang.IllegalArgumentException` - if duration is negative.

Methods

getDuration()

```
public long getDuration()
```

Returns the duration passed as an argument to the constructor.

Returns: the duration passed into the constructor

getSource()

```
public javax.tv.locator.Locator[] getSource()
```

Returns the source of the recording

Returns: the source passed into the constructor

getStartTime()

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

Returns the start time passed as an argument to the constructor.

Returns: the start time passed into the constructor

org.ocap.shared.dvr NoMoreDataEntriesException

Declaration

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

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

All Implemented Interfaces: java.io.Serializable

Description

No more data entries allowed for this recording request.

Member Summary

Constructors

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

Constructors

NoMoreDataEntriesException()

```
public NoMoreDataEntriesException()
```

Constructs a NoMoreDataEntriesException with no detail message

org.ocap.shared.dvr ParentRecordingRequest

Declaration

```
public interface ParentRecordingRequest extends RecordingRequest
```

All Superinterfaces: RecordingRequest

Description

This interface represents information corresponding to a parent recording request. The recording request represented by this interface may have one or more child recording requests.

A parent recording request may be in the unresolved state, partially resolved state, completely resolved state, or the cancelled state.

A recording request would be in the unresolved state if the implementation does not have enough information to process this recording request. A recording request in this state may transition to partially resolved state, completely resolved state, or failed state.

A recording request would be in the partially resolved state if the implementation has enough information to schedule some, but not all, child recording requests corresponding to this recording request. This would be the case of a recording request for a series where some episodes for the series are scheduled. A recording request is in completely resolved state if all its child recordings are known and scheduled.

Member Summary	
Fields	
static int	CANCELLED_STATE A recording request is in cancelled state, if an application has successfully called the cancel method for this recording request, but not all child recording request have been deleted.
static int	COMPLETELY_RESOLVED_STATE All child recordings corresponding to this recording request have been scheduled.
static int	PARTIALLY_RESOLVED_STATE Some recordings corresponding to this recording request have been scheduled.
static int	UNRESOLVED_STATE The implementation does not have enough information to process this recording request.
Methods	
void	cancel() Cancels the parent recording request.
org.ocap.shared.dvr.navigat ion.RecordingList	getKnownChildren() Gets all the child Recordings corresponding to this parent RecordingRequest.

Fields

CANCELLED_STATE

```
public static final int CANCELLED_STATE
```

A recording request is in cancelled state, if an application has successfully called the cancel method for this recording request, but not all child recording request have been deleted. A recording request in this state shall be deleted by the implementation once all child recording requests have been deleted.

COMPLETELY_RESOLVED_STATE

```
public static final int COMPLETELY_RESOLVED_STATE
```

All child recordings corresponding to this recording request have been scheduled. A recording request is in completely resolved state, if the implementation has enough information to schedule all recordings corresponding to the recording request. A recording request in completely resolved state would have one or more child recordings.

PARTIALLY_RESOLVED_STATE

```
public static final int PARTIALLY_RESOLVED_STATE
```

Some recordings corresponding to this recording request have been scheduled. A recording request is in partially resolved state, if the implementation has enough information to schedule some, but not all recordings corresponding to the recording request. A recording request in partially resolved state may have zero, one or more child recordings.

UNRESOLVED_STATE

```
public static final int UNRESOLVED_STATE
```

The implementation does not have enough information to process this recording request.

Methods

cancel()

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

Cancels the parent recording request. The implementation shall also cancel all the child recording requests through calling their cancel() method. No more child recordings will be scheduled for this recording request or for any of its child recordings. The recording request will be deleted from the database once all child recording requests has been deleted. Canceling a parent recording request does not delete any child recordings that cannot be cancelled (i.e. if a child recording request is not in a pending state). At the successful completion of this method the recording request would be deleted from the database or changed state to CANCELLED_STATE.

Throws:

`java.lang.SecurityException` - if the calling application does not have `RecordingPermission("cancel",...)` or `RecordingPermission("*",...)`

`AccessDeniedException` - if the calling application is not permitted to perform this operation by `RecordingRequest` specific security attributes.

`java.lang.IllegalStateException` - if the state of the recording request is not in `UNRESOLVED_STATE`, `PARTIALLY_RESOLVED_STATE` or `COMPLETELY_RESOLVED_STATE`.

getKnownChildren()

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

Gets all the child Recordings corresponding to this parent RecordingRequest. For a recording request in completely resolved state this method returns all children that are still maintained in the recording manager database (i.e. children removed from the database by calling the `delete()` or `cancel()` method will not be included in the list of child recordings). For a recording request in partially resolved state this method only returns currently known children for series.

Returns: The list of child Recordings corresponding to this Recording; null if there are no child recording requests in the RecordingManager database.

Throws:

`java.lang.IllegalStateException` - if the recording request is not in `PARTIALLY_RESOLVED_STATE` or `COMPLETELY_RESOLVED_STATE`.

org.ocap.shared.dvr RecordedService

Declaration

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

All Superinterfaces: javax.tv.service.Service

Description

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

A RecordedService has a media time attribute that determines where playback begins when the recorded service is selected on a ServiceContext. This media time is persistent and is applicable for all future service selections by all applications until the media time is changed with a call to setMediaTime.

Note the following subinterface-specific behavior for methods defined by the javax.tv.service.Service superinterface:

- The hasMultipleInstances() method shall always return false.
- The getServiceType() method shall always return RecordedServiceType.RECORDED_SERVICE.
- The getLocator() method shall return a locator corresponding to the recorded service that is different from the locator of the originating service. This locator when passed in to the SIManager.getService(..) should return this RecordedService.
- The getName() method shall return a human readable string.
- RecordedServices shall not be included in service lists returned by the method SIManager.filterServices(..).

Member Summary

Methods

	void	delete()	Deletes the recorded service.
javax.media.Time		getFirstMediaTime()	Gets the JMF media time at the start of the recorded service.
javax.media .MediaLocator		getMediaLocator()	Returns the MediaLocator corresponding to the RecordedService.
javax.media.Time		getMediaTime()	Gets the JMF media time that was set using the method setMediaTime(..)
	long	getRecordedDuration()	Gets the actual duration of the content recorded.
RecordingRequest		getRecordingRequest()	Gets the RecordingRequest corresponding to the RecordedService.
java.util.Date		getRecordingStartTime()	Gets the time when the recording was initiated.

Member Summary

```
void setMediaTime(javax.media.Time mediaTime)
```

Set the JMF media time for the location from where the playback will begin when this recorded service is selected on a ServiceContext.

Methods

delete()

```
public void delete()
    throws AccessDeniedException
```

Deletes the recorded service. The method removes the recorded service and all recorded elementary streams (e.g., files and directory entries) associated with the RecordedService. The corresponding recording request will transition to DELETED_STATE.

If the recording request is in the IN_PROGRESS state the implementation will stop the recording before deleting the recorded service. If the RecordedService was being presented when it was deleted, a `javax.tv.service.selection.PresentationTerminatedEvent` will be sent with reason SERVICE_VANISHED.

Throws:

`AccessDeniedException` - if the calling application is not permitted to perform this operation by RecordingRequest specific security attributes.

`java.lang.SecurityException` - if the calling application does not have `RecordingPermission("delete",..)` or `RecordingPermission("*",..)`

getFirstMediaTime()

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

Gets the JMF media time at the start of the recorded service.

Returns: a media time

getMediaLocator()

```
public javax.media.MediaLocator getMediaLocator()
```

Returns the MediaLocator corresponding to the RecordedService.

Returns: RecordedService MediaLocator.

getMediaTime()

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

Gets the JMF media time that was set using the method `setMediaTime(..)`

Returns: the value of JMF media time that was set using the method `setMediaTime(..)`, if that method was called on this RecordedService before. If the method `setMediaTime` was not called before, this method Should return the JMF media time corresponding to the beginning of the RecordedService.

getRecordedDuration()

```
public long getRecordedDuration()
```

Gets the actual duration of the content recorded. For recordings in progress this will return the duration of the completed part of the recording.

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

getRecordingRequest()

```
public org.ocap.shared.dvr.RecordingRequest getRecordingRequest ()
```

Gets the `RecordingRequest` corresponding to the `RecordedService`.

Returns: The `RecordingRequest` for the service.

getRecordingStartTime()

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

Gets the time when the recording was initiated.

Returns: the time when the recording was initiated by the implementation.

setMediaTime(Time)

```
public void setMediaTime(javax.media.Time mediaTime)  
    throws AccessDeniedException
```

Set the JMF media time for the location from where the playback will begin when this recorded service is selected on a `ServiceContext`.

If an instance of `Time` corresponding to value of 0 nanoseconds, or a negative value is set, or if this method was not called for this recorded service, the playback will begin at the start of the recorded content. If the instance of `Time` set corresponds to positive infinity or a value more than the duration of the recorded content, the playback will begin at the live point if recording is in progress for the recorded service. If the recording is not in progress, the playback will immediately hit the end-of-media. Note: The media time set will be applicable for all future service selections by all applications.

Parameters:

`mediaTime` - the media time corresponding to the location from where the playback will begin when this service is selected.

Throws:

`AccessDeniedException` - if the calling application is not permitted to perform this operation by `RecordingRequest` specific security attributes corresponding to the `RecordingRequest` associated with this recorded service.

`java.lang.SecurityException` - if the calling application does not have `RecordingPermission("modify",...)` or `RecordingPermission("*",...)`

org.ocap.shared.dvr RecordedServiceType

Declaration

```
public class RecordedServiceType extends javax.tv.service.ServiceType
```

```
java.lang.Object
|
+--javax.tv.service.ServiceType
|
+--org.ocap.shared.dvr.RecordedServiceType
```

Description

This class represents the service type value for a RecordedService.

Member Summary	
Fields	
static	RECORDED_SERVICE_TYPE
javax.tv.service	ServiceType for a Recorded service.
.ServiceType	
Constructors	
protected	RecordedServiceType(java.lang.String name)
	Provides an instance of RecordedServiceType.

Fields

RECORDED_SERVICE_TYPE

```
public static final javax.tv.service.ServiceType RECORDED_SERVICE_TYPE
ServiceType for a Recorded service.
```

Constructors

RecordedServiceType(String)

```
protected RecordedServiceType(java.lang.String name)
Provides an instance of RecordedServiceType.
```

Parameters:

name - The string name of this type (i.e. "RECORDED_SERVICE").

org.ocap.shared.dvr RecordingChangedEvent

Declaration

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

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

All Implemented Interfaces: java.io.Serializable

Description

Event used to notify listeners of changes in the list of recording requests maintained by the RecordingManager.

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

Fields

ENTRY_ADDED

```
public static final int ENTRY_ADDED
```

A new RecordingRequest was added.

ENTRY_DELETED

```
public static final int ENTRY_DELETED
```

A RecordingRequest was deleted.

ENTRY_STATE_CHANGED

```
public static final int ENTRY_STATE_CHANGED
```

The state of a RecordingRequest changed

Constructors

RecordingChangedEvent(RecordingRequest, int, int)

```
public RecordingChangedEvent(org.ocap.shared.dvr.RecordingRequest source,  
                             int newState, int oldState)
```

Constructs the event.

Parameters:

source - The RecordingRequest that caused the event.

newState - the state the RecordingRequest is now in.

oldState - the state the RecordingRequest was in before the state change.

Methods

getChange()

```
public int getChange()
```

Returns the change to the RecordingRequest.

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

getOldState()

```
public int getOldState()
```

Returns the old state for the RecordingRequest.

Returns: The old state.

getRecordingRequest()

```
public org.ocap.shared.dvr.RecordingRequest getRecordingRequest()
```

Returns the RecordingRequest that caused the event.

Returns: The RecordingRequest that caused the event.

getState()

```
public int getState()
```

Returns the new state for the RecordingRequest.

Returns: The new state.

org.ocap.shared.dvr RecordingChangeListener

Declaration

```
public interface RecordingChangeListener 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 recordingChanged(RecordingChangedEvent e)  
    Notifies the RecordingChangeListener of an event generated by the  
    RecordingManager.
```

Methods

recordingChanged(RecordingChangedEvent)

```
public void recordingChanged(org.ocap.shared.dvr.RecordingChangedEvent e)
```

Notifies the RecordingChangeListener of an event generated by the RecordingManager. Events are generated when there are changes in the list of recording requests maintained by the recording manager.

Parameters:

e - The generated event.

org.ocap.shared.dvr RecordingFailedException

Declaration

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

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

All Implemented Interfaces: java.io.Serializable

Description

This exception is returned when applications call the `getFailedException()` for a failed recording request or an incomplete recording request.

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	RESOLUTION_ERROR Reason code: The recording request failed due to errors in request resolution.
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.

Member Summary	
static int	TUNING_FAILURE Reason code: Recording failed due to problems with tuning.
static int	USER_STOP Reason code: The application terminated the recording using LeafRecordingRequest.stop method or by calling the stop on the service context (if the recording spec is an instance of ServiceContextRecordingSpec).
Constructors	
	RecordingFailedException() Constructs a RecordingFailedException with no detail message
Methods	
int	getReason() Reports the reason for which the recording request 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.

RESOLUTION_ERROR

```
public static final int RESOLUTION_ERROR
```

Reason code: The recording request failed due to errors in request resolution.

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 spec for the recording request in an instance of ServiceContextRecordingSpec.

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 LeafRecordingRequest.stop method or by calling the stop on the service context (if the recording spec is an instance of ServiceContextRecordingSpec).

Constructors

RecordingFailedException()

```
public RecordingFailedException()
```

Constructs a RecordingFailedException with no detail message

Methods

getReason()

```
public int getReason()
```

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

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

org.ocap.shared.dvr RecordingManager

Declaration

```
public abstract class RecordingManager
    java.lang.Object
    |
    +--org.ocap.shared.dvr.RecordingManager
```

Description

RecordingManager represents the entity that performs recordings.

Member Summary	
Constructors	
protected	RecordingManager () Constructor for instances of this class.
Methods	
abstract void	addRecordingChangedListener (RecordingChangedListener rcl) Adds an event listener for changes in status of recording requests.
org.ocap.shared.dvr.navigation.RecordingList	abstract getEntries () Gets the list of entries maintained by the RecordingManager.
org.ocap.shared.dvr.navigation.RecordingList	abstract getEntries (org.ocap.shared.dvr.navigation.RecordingListFilter filter) Gets the list of recording requests matching the specified filter.
RecordingManager	static getInstance () Gets the singleton instance of RecordingManager.
RecordingRequest	abstract record (RecordingSpec source) Records the stream or streams according to the source parameter.
abstract void	removeRecordingChangedListener (RecordingChangedListener rcl) Removes a registered event listener for changes in status of recording requests.

Constructors

RecordingManager()

```
protected RecordingManager ()
```

Constructor for instances of this class. This constructor is provided for the use of implementations and specifications which extend this specification. Applications shall not define sub-classes of this class. Implementations are not required to behave correctly if any such application defined sub-classes are used.

Methods

addRecordingChangedListener(RecordingChangedListener)

```
public abstract void
    addRecordingChangedListener(org.ocap.shared.dvr.RecordingChangedListener
        rcl)
```

Adds an event listener for changes in status of recording requests. For applications with RecordingPermission("read", "own"), the listener parameter will only be informed of changes that affect RecordingRequests of which the calling application has visibility as defined by any RecordingRequest specific security attributes. For applications with RecordingPermission("read", "*"), the listener parameter will be informed of all changes.

Parameters:

rcl - The listener to be registered.

Throws:

java.lang.SecurityException - if the calling application does not have RecordingPermission("read",...) or RecordingPermission("*",...)

getEntries()

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

Gets the list of entries maintained by the RecordingManager. This list includes both parent and leaf recording requests. For applications with RecordingPermission("read", "own"), only RecordingRequests of which the calling application has visibility as defined by any RecordingRequest specific security attributes will be returned. For applications with RecordingPermission("read", "*"), all RecordingRequests will be returned.

Returns: an instance of RecordingList

Throws:

java.lang.SecurityException - if the calling application does not have RecordingPermission("read",...) or RecordingPermission("*",...)

getEntries(RecordingListFilter)

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

Gets the list of recording requests matching the specified filter. For applications with RecordingPermission("read", "own"), only RecordingRequests of which the calling application has visibility as defined by any RecordingRequest specific security attributes will be returned. For applications with RecordingPermission("read", "*"), all RecordingRequests matching the specified filter will be returned.

Parameters:

filter - the filter to use on the total set of recording requests

Returns: an instance of RecordingList

Throws:

java.lang.SecurityException - if the calling application does not have RecordingPermission("read",...) or RecordingPermission("*",...)

getInstance()

```
public static org.ocap.shared.dvr.RecordingManager getInstance()
```

Gets the singleton instance of RecordingManager.

Returns: an instance of RecordingManager

record(RecordingSpec)

```
public abstract org.ocap.shared.dvr.RecordingRequest  
    record(org.ocap.shared.dvr.RecordingSpec source)  
    throws IllegalArgumentException, AccessDeniedException
```

Records the stream or streams according to the source parameter. The concrete sub-class of RecordingSpec may define additional semantics to be applied when instances of that sub-class are used.

Parameters:

source - specification of stream or streams to be recorded and how they are to be recorded.

Returns: an instance of RecordingRequest that represents the added recording.

Throws:

java.lang.IllegalArgumentException - if the source is an application defined class or as defined in the concrete sub-class of RecordingSpec for instances of that class

AccessDeniedException - if the calling application is not permitted to perform this operation by RecordingRequest specific security attributes.

java.lang.SecurityException - if the calling application does not have RecordingPermission("create",..) or RecordingPermission("*",..)

removeRecordingChangedListener(RecordingChangedListener)

```
public abstract void  
    removeRecordingChangedListener(org.ocap.shared.dvr.RecordingChangedListe  
ner rcl)
```

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

Parameters:

rcl - the listener to be removed.

org.ocap.shared.dvr RecordingPermission

Declaration

```
public final class RecordingPermission extends java.security.Permission
```

```
java.lang.Object
|
+--java.security.Permission
|
+--org.ocap.shared.dvr.RecordingPermission
```

All Implemented Interfaces: java.security.Guard, java.io.Serializable

Description

Controls access to recording features by an application. The name can be one of the values shown in the following list;

- “create” - schedule a RecordingRequest
- “read” - obtain the list of RecordingRequests
- “modify” - modify properties or application specific data for a RecordingRequest
- “delete” - delete a RecordingRequest including recorded content
- “cancel” - cancel a pending RecordingRequest
- “*” - all of the above

The action can be “own” and “*”. The action “own” is intended for use by normal applications. The action “*” is intended for use only by specially privileged applications and permits the operation defined by the name to be applied to all RecordingRequests regardless of any per-application restrictions associated with the RecordingRequest.

Granting of this permission shall include granting access to any storage devices required for the operations specified in the name parameter. No additional low permissions (e.g. FilePermission) are subsequently needed.

Member Summary

Constructors

```
RecordingPermission(java.lang.String name,
java.lang.String action)
Creates a new RecordingPermission with the specified name and action.
```

Methods

```
boolean equals(java.lang.Object obj)
Checks two RecordingPermission objects for equality
java.lang.String getActions()
Returns the actions as passed into the constructor.
int hashCode()
Returns the hash code value for this object.
```

Member Summary

```
boolean implies(java.security.Permission p)
    Check if this RecordingPermission "implies" the specified Permission.
```

Constructors

RecordingPermission(String, String)

```
public RecordingPermission(java.lang.String name, java.lang.String action)
```

Creates a new RecordingPermission with the specified name and action.

Parameters:

name - "create", "read", "modify", "delete", "cancel" or "*"

action - "own" or "*"

Methods

equals(Object)

```
public boolean equals(java.lang.Object obj)
```

Checks two RecordingPermission objects for equality

Overrides: equals in class Permission

Parameters:

obj - the object to test for equality with this object.

Returns: true if obj is a RecordingPermission with the same name and action as this RecordingPermission object

getActions()

```
public java.lang.String getActions()
```

Returns the actions as passed into the constructor.

Overrides: getActions in class Permission

Returns: the actions as a String

hashCode()

```
public int hashCode()
```

Returns the hash code value for this object.

Overrides: hashCode in class Permission

Returns: a hash code value for this object.

implies(Permission)

```
public boolean implies(java.security.Permission p)
```

Check if this RecordingPermission "implies" the specified Permission.

Overrides: `implies` in class `Permission`

Parameters:

`p` - the permission to check against

Returns: `true` if the specified permission is implied by this object, `false` if not.

org.ocap.shared.dvr RecordingProperties

Declaration

```
public abstract class RecordingProperties
    java.lang.Object
    |
    +--org.ocap.shared.dvr.RecordingProperties
```

Description

Base class for specifying properties defining how a recording is to be made.

Member Summary	
Constructors	RecordingProperties(long expirationPeriod) Constructor
Methods	long getExpirationPeriod() Return the value of the expiration period.

Constructors

RecordingProperties(long)

```
public RecordingProperties(long expirationPeriod)
```

Constructor

Parameters:

expirationPeriod - The period in seconds after the initiation of recording when leaf recording requests with this recording property are deemed as expired.

Methods

getExpirationPeriod()

```
public long getExpirationPeriod()
```

Return the value of the expiration period.

Returns: The expiration period as passed into the constructor.

org.ocap.shared.dvr RecordingRequest

Declaration

```
public interface RecordingRequest
```

All Known Subinterfaces: LeafRecordingRequest, ParentRecordingRequest

Description

This interface represents information corresponding to a recording request. The recording request represented by this interface may correspond to a single recording request or a series of other recording requests. Recording requests are hierarchical in nature. Implementations may resolve a recording request to a single recording request or to a series of other recording requests each of which may get further resolved into a single recording request or a series of recording requests. For Example, a recording request for “Sex and the City” may resolve to multiple recording requests, each for a particular season of the show. Each of these recording requests may get further resolved into multiple recording requests, each for a single episode. The implementation creates a recording request in response to the RecordingManager.record(RecordingSpec) method. The implementation also creates recording requests when a recording request is further resolved.

A recording request may either be a parent recording request or a leaf recording request. States for a recording request are defined in ParentRecordingRequest and LeafRecordingRequest. A recording request may be in any of the states corresponding to a leaf recording request or a parent recording request.

Member Summary

Methods

	void	addAppData(java.lang.String key, java.io.Serializable data)	Add application specific private data.
	void	delete()	Deletes the recording request from the database.
java.io.Serializable		getAppData(java.lang.String key)	Get application data corresponding to specified key.
org.dvb.application.AppID		getAppID()	Gets the application identifier of the application that owns this recording request.
java.lang.String[]		getKeys()	Get all Application specific data associated with this recording request.
RecordingRequest		getParent()	Gets the parent recording request corresponding to this recording request.
RecordingSpec		getRecordingSpec()	Returns the RecordingSpec corresponding to the recording request.
RecordingRequest		getRoot()	Gets the root recording request corresponding to this recording request.
	int	getState()	Returns the state of the recording request.

Member Summary

<pre> boolean isRoot () void removeAppData (java.lang.String key) void reschedule (RecordingSpec newRecordingSpec) void setRecordingProperties (RecordingProperties properties) </pre>	<p>Checks whether the recording request was a root recording request generated when the application called the <code>RecordingManager.record(..)</code> method.</p> <p>Remove Application specific private data corresponding to the specified key.</p> <p>Modify the details of a recording request.</p> <p>Modify the <code>RecordingProperties</code> corresponding to the <code>RecordingSpec</code> for this recording request.</p>
--	--

Methods

addAppData(String, Serializable)

```

public void addAppData (java.lang.String key, java.io.Serializable data)
    throws NoMoreDataEntriesException, AccessDeniedException

```

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

Parameters:

key - the ID under which the data is to be added

data - the data to be added

Throws:

`java.lang.SecurityException` - if the calling application does not have `RecordingPermission("modify",...)` or `RecordingPermission("*",...)`

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

`NoMoreDataEntriesException` - if the recording request is unable to store any more Application data. The implementation shall support at least 16 data entries per recording request.

`AccessDeniedException` - if the calling application is not permitted to perform this operation by `RecordingRequest` specific security attributes.

delete()

```

public void delete ()
    throws AccessDeniedException

```

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

If the recording request is in the `IN_PROGRESS` state the implementation will stop the recording before deleting the recording request. If a `RecordedService` was being presented when it was deleted, a `javax.tv.service.selection.PresentationTerminatedEvent` will be sent with reason `SERVICE_VANISHED`.

Throws:

`AccessDeniedException` - if the calling application is not permitted to perform this operation by `RecordingRequest` specific security attributes.

`java.lang.SecurityException` - if the calling application does not have `RecordingPermission("delete",..)` or `RecordingPermission("*",..)`

getAppData(String)

```
public java.io.Serializable getAppData(java.lang.String key)
```

Get application data corresponding to specified key.

Parameters:

key - the key under which any data is to be returned

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

getAppID()

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

Gets the application identifier of the application that owns this recording request. The owner of a root recording request is the application that called the `RecordingManager.record(..)` method. The owner of a non-root recording request is the owner of the root for the recording request.

Returns: Application identifier of the owning application.

getKeys()

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

Get all Application specific data associated with this recording request.

Returns: All keys corresponding to the `RecordingRequest`; Null if there if no application data.

getParent()

```
public org.ocap.shared.dvr.RecordingRequest getParent()
```

Gets the parent recording request corresponding to this recording request.

Returns: the parent recording request for this recording request, null if the application does not have read access permission for the parent recording request or if this recording request is the root recording request.

getRecordingSpec()

```
public org.ocap.shared.dvr.RecordingSpec getRecordingSpec()
```

Returns the `RecordingSpec` corresponding to the recording request. This will be either the source as specified in the call to the `record(..)` method which caused this recording request to be created or the `RecordingSpec` generated by the system during the resolution of the original application specified `RecordingSpec`. Any modification to the `RecordingSpec` due to any later calls to the `SetRecordingProperties` methods on this instance will be reflected on the returned `RecordingSpec`.

When the implementation generates a recording request while resolving another recording request, a new instance of the `RecordingSpec` is created with an identical copy of the `RecordingProperties` of the parent recording request.

Returns: a RecordingSpec containing information about this recording request.

getRoot()

```
public org.ocap.shared.dvr.RecordingRequest getRoot()
```

Gets the root recording request corresponding to this recording request. A root recording request is the recording request that was returned when the application called the RecordingManager.record(..) method.

If the current recording request is a root recording request, the current recording request is returned.

Returns: the root recording request for this recording request, null if the application does not have read access permission for the root recording request.

getState()

```
public int getState()
```

Returns the state of the recording request.

Returns: State of the recording request.

isRoot()

```
public boolean isRoot()
```

Checks whether the recording request was a root recording request generated when the application called the RecordingManager.record(..) method. The implementation should create a root recording request corresponding to each successful call to the record method.

Returns: True, if the recording request is a root recording request, false if the recording request was generated during the process of resolving another recording request.

removeAppData(String)

```
public void removeAppData(java.lang.String key)
        throws AccessDeniedException
```

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

Parameters:

key - the key under which data is to be removed

Throws:

AccessDeniedException - if the calling application is not permitted to perform this operation by RecordingRequest specific security attributes.

java.lang.SecurityException - if the calling application does not have RecordingPermission("modify",...) or RecordingPermission("*",...)

reschedule(RecordingSpec)

```
public void reschedule(org.ocap.shared.dvr.RecordingSpec newRecordingSpec)
        throws AccessDeniedException
```

Modify the details of a recording request. The recording request shall be re-evaluated based on the newly provided RecordingSpec. Rescheduling a root recording request may result in state transitions for the root recording request or its child recording requests. Rescheduling a root recording request

may also result in the scheduling of one or more new child recording requests, or the deletion of one or more pending child recording requests.

Note: If the recording request or one of its child recording request is in `IN_PROGRESS_STATE` or `IN_PROGRESS_INSUFFICIENT_SPACE_STATE`, any changes to the start time shall be ignored. In this case all other valid parameters are applied. If the new value for a parameter is not valid (e.g. the start-time and the duration is in the past), the implementation shall ignore that parameter. In-progress recordings shall continue uninterrupted, if the new recording spec does not request the recording to be stopped.

Parameters:

`newRecordingSpec` - the new recording spec that shall be used to reschedule the root `RecordingRequest`.

Throws:

`java.lang.IllegalArgumentException` - if the new recording spec and the current recording spec for the recording request are different sub-classes of `RecordingSpec`.

`AccessDeniedException` - if the calling application is not permitted to perform this operation by `RecordingRequest` specific security attributes.

`java.lang.SecurityException` - if the calling application does not have `RecordingPermission("modify",...)` or `RecordingPermission("*",...)*`

setRecordingProperties(RecordingProperties)

```
public void setRecordingProperties(org.ocap.shared.dvr.RecordingProperties
    properties)
    throws IllegalStateException, AccessDeniedException
```

Modify the `RecordingProperties` corresponding to the `RecordingSpec` for this recording request. Applications may change any properties associated with a recording request by calling this method. Changing the properties may result in changes in the states of this recording request. Changing the properties of a parent recording request will not automatically change the properties of any of its child recording requests that are already created. Any child recording requests created after the invocation of this method will inherit the new values for the properties.

Parameters:

`properties` - the new recording properties to set.

Throws:

`java.lang.IllegalStateException` - if changing one of the parameters that has been modified in the new recording properties is not legal for the current state of the recording request.

`AccessDeniedException` - if the calling application is not permitted to perform this operation by `RecordingRequest` specific security attributes.

`java.lang.SecurityException` - if the calling application does not have `RecordingPermission("modify",...)` or `RecordingPermission("*",...)`

org.ocap.shared.dvr RecordingSpec

Declaration

```
public abstract class RecordingSpec
    java.lang.Object
    |
    +--org.ocap.shared.dvr.RecordingSpec
```

Direct Known Subclasses: LocatorRecordingSpec, ServiceContextRecordingSpec, ServiceRecordingSpec

Description

Base class for specifying what to record and how to record it.

Member Summary	
Constructors	
	RecordingSpec(RecordingProperties properties) Constructor.
Methods	
	RecordingProperties getProperties() Return the description of how the recording is to be done.

Constructors

RecordingSpec(RecordingProperties)

```
public RecordingSpec(org.ocap.shared.dvr.RecordingProperties properties)
    Constructor.
```

Parameters:

properties - The definition of how the recording is to be done.

Methods

getProperties()

```
public org.ocap.shared.dvr.RecordingProperties getProperties()
    Return the description of how the recording is to be done.
```

Returns: The properties to use for the recording.

org.ocap.shared.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.shared.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.ServiceContext source, int reason)
    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 `RecordingRequest` `stop` method is called.

Constructors

RecordingTerminatedEvent(ServiceContext, int)

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

Constructs the event.

Parameters:

`source` - The `ServiceContext` that generated the event.

Methods

getReason()

```
public int getReason()
```

Returns the reason for the record termination.

Returns: Termination reason; see constants in this class.

org.ocap.shared.dvr ServiceContextRecordingSpec

Declaration

```
public class ServiceContextRecordingSpec extends RecordingSpec
```

```
java.lang.Object
|
|--org.ocap.shared.dvr.RecordingSpec
|   |
|   |--org.ocap.shared.dvr.ServiceContextRecordingSpec
```

Description

Specifies a recording request in terms of what is being presented on a ServiceContext. The streams that are being presented in the indicated ServiceContext parameter are recorded. If the Service being recorded from is tuned away, recording SHALL be terminated. If the startTime is in the past and the source javax.tv.service.selection.ServiceContext is associated with a time shift buffer, the contents of the time-shift buffer may be 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 may be 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.

When instances of this class are passed to RecordingManager.record(..), the following additional failure modes shall apply;

- IllegalArgumentException SHALL be thrown if serviceContext is not presenting a broadcast service or if the startTime is in the future
- SecurityException SHALL be thrown if the application does not have permission to access the service context.

When an instance of this recording spec is passed in as a parameter to the RecordingRequest.reschedule(..) method, an IllegalArgumentException is thrown if the service context parameter is different from the service context specified in the current recording spec for the recording request.

Member Summary	
Constructors	
	ServiceContextRecordingSpec(javax.tv.service.selection.ServiceContext serviceContext, java.util.Date startTime, long duration, RecordingProperties properties) Constructor
Methods	
long	getDuration() Returns the duration passed as an argument to the constructor.
javax.tv.service.selection.ServiceContext	getServiceContext() Returns the ServiceContext to record from

Member Summary

```
java.util.Date getStartTime()  
Returns the start time passed as an argument to the constructor.
```

Constructors

ServiceContextRecordingSpec(ServiceContext, Date, long, RecordingProperties)

```
public ServiceContextRecordingSpec(javax.tv.service.selection.ServiceContext  
    serviceContext, java.util.Date startTime, long duration,  
    org.ocap.shared.dvr.RecordingProperties properties)  
    throws IllegalArgumentException
```

Constructor

Parameters:

`serviceContext` - The ServiceContext to record from.

`startTime` - Start time of the recording. If the start time is in the future when the RecordingManager.record(..) method is called with this ServiceContextRecordingSpec as an argument the record method will throw an IllegalArgumentException.

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

`properties` - the definition of how the recording is to be done

Throws:

`java.lang.IllegalArgumentException` - if duration is negative

Methods

getDuration()

```
public long getDuration()
```

Returns the duration passed as an argument to the constructor.

Returns: the duration passed into the constructor

getServiceContext()

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

Returns the ServiceContext to record from

Returns: the ServiceContext instance passed into the constructor

getStartTime()

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

Returns the start time passed as an argument to the constructor.

Returns: the start time passed into the constructor

org.ocap.shared.dvr ServiceRecordingSpec

Declaration

```
public class ServiceRecordingSpec extends RecordingSpec
```

```
java.lang.Object
|
+--org.ocap.shared.dvr.RecordingSpec
|
+--org.ocap.shared.dvr.ServiceRecordingSpec
```

Description

Specifies a recording request in terms of a Service. When instances of this class are passed to RecordingManager.record(..), no additional failure modes shall apply.

When an instance of this recording spec is passed in as a parameter to the RecordingRequest.reschedule(..) method, an IllegalArgumentException is thrown if the source is different from the source specified in the current recording spec for the recording request and if the recording request is in progress state.

Member Summary	
Constructors	
	ServiceRecordingSpec(javax.tv.service.Service source, java.util.Date startTime, long duration, RecordingProperties properties) Constructor.
Methods	
long	getDuration() Returns the duration passed as an argument to the constructor.
javax.tv.service.Service	getSource() Returns the source of the recording
java.util.Date	getStartTime() Returns the start time passed as an argument to the constructor.

Constructors

ServiceRecordingSpec(Service, Date, long, RecordingProperties)

```
public ServiceRecordingSpec(javax.tv.service.Service source,
    java.util.Date startTime, long duration,
    org.ocap.shared.dvr.RecordingProperties properties)
    throws IllegalArgumentException
```

Constructor.

Parameters:

source - the service to be recorded

`startTime` - Start time of the recording. If the start time is in the past when the record method is called with this `ServiceRecordingSpec` as an argument, the current time is used instead and the duration reduced by the same amount. If the end time as computed as the start time + the duration is in the past when the record method is called, the record method will throw an `IllegalArgumentException`.

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

`properties` - The definition of how the recording is to be done.

Throws:

`java.lang.IllegalArgumentException` - if the source is not a broadcast service or if the duration is negative

Methods

getDuration()

```
public long getDuration()
```

Returns the duration passed as an argument to the constructor.

Returns: the duration passed into the constructor

getSource()

```
public javax.tv.service.Service getSource()
```

Returns the source of the recording

Returns: the source passed into the constructor

getStartTime()

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

Returns the start time passed as an argument to the constructor.

Returns: the start time passed into the constructor

This page intentionally left blank.

Annex G OCAP Shared DVR Navigation API (org.ocap.shared.dvr.navigation)

This section presents the `org.ocap.shared.dvr.navigation` API.

Package

org.ocap.shared.dvr.navigation

Description

Provides support for Navigation of recording lists.

Class Summary	
Interfaces	
<code>RecordingList</code>	RecordingList represents a list of recordings.
<code>RecordingListComparator</code>	This interface represent a sorting criteria to be applied while sorting a RecordingList.
<code>RecordingListIterator</code>	This iterator could be used to traverse entries in a RecordingList.
Classes	
<code>AppIDFilter</code>	Filter to filter based on AppID.
<code>OrgIDFilter</code>	Filter to filter based on OrgID.
<code>RecordingListFilter</code>	Base class for all RecordingListFilters.
<code>RecordingStateFilter</code>	Filter to filter based on values returned by the <code>getState</code> method in <code>org.ocap.shared.dvr.RecordingRequest</code> .

org.ocap.shared.dvr.navigation AppIDFilter

Declaration

```
public class AppIDFilter extends RecordingListFilter
    java.lang.Object
    |
    |--org.ocap.shared.dvr.navigation.RecordingListFilter
    |
    |--org.ocap.shared.dvr.navigation.AppIDFilter
```

Description

Filter to filter based on AppID.

Member Summary	
Constructors	
	AppIDFilter(org.dvb.application.AppID appID) Constructs the filter based on a particular AppID. This page intentionally left blank.
Methods	
boolean	accept(org.ocap.shared.dvr.RecordingRequest entry) Tests if the given RecordingRequest passes the filter.
org.dvb.application.AppID	getFilterValue() 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 RecordingRequests.

Methods

accept(RecordingRequest)

```
public boolean accept(org.ocap.shared.dvr.RecordingRequest entry)
```

Tests if the given RecordingRequest passes the filter.

Overrides: accept in class RecordingListFilter

Parameters:

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

Returns: true if RecordingRequest 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.shared.dvr.navigation OrgIDFilter

Declaration

```
public class OrgIDFilter extends RecordingListFilter
    java.lang.Object
    |
    |--org.ocap.shared.dvr.navigation.RecordingListFilter
    |
    |--org.ocap.shared.dvr.navigation.OrgIDFilter
```

Description

Filter to filter based on OrgID.

Member Summary	
Constructors	
	<pre>OrgIDFilter(int orgID)</pre> <p>Constructs the filter based on a particular organization ID.</p>
Methods	
boolean	<pre>accept(org.ocap.shared.dvr.RecordingRequest entry)</pre> <p>Tests if the given org.ocap.shared.dvr.RecordingRequest passes the filter.</p>
int	<pre>getFilterValue()</pre> <p>Reports the value of the organization ID used to create this filter.</p>

Constructors

OrgIDFilter(int)

```
public OrgIDFilter(int orgID)
```

Constructs the filter based on a particular organization ID.

Parameters:

orgID - the Organization ID value for matching
org.ocap.shared.dvr.RecordingRequest instances.

Methods

accept(RecordingRequest)

```
public boolean accept(org.ocap.shared.dvr.RecordingRequest entry)
```

Tests if the given org.ocap.shared.dvr.RecordingRequest passes the filter.

Overrides: accept in class RecordingListFilter

Parameters:

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

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

getFilterValue()

```
public int getFilterValue()
```

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

Returns: The organization ID used to filter.

org.ocap.shared.dvr.navigation RecordingList

Declaration

```
public interface RecordingList
```

Description

RecordingList represents a list of recordings.

Member Summary

Methods

	boolean	contains(org.ocap.shared.dvr.RecordingRequest entry)	Tests if the indicated RecordingRequest object is contained in the list.
RecordingListIterator		createRecordingListIterator()	Generates an iterator on the RecordingRequest elements in this list.
RecordingList		filterRecordingList(RecordingListFilter filter)	Creates a new RecordingList object that is a subset of this list, based on the conditions specified by a RecordingListFilter object.
org.ocap.shared.dvr .RecordingRequest		getRecordingRequest(int index)	Reports the RecordingRequest at the specified index position.
	int	indexOf(org.ocap.shared.dvr.RecordingRequest entry)	Reports the position of the first occurrence of the indicated RecordingRequest object in the list.
	int	size()	Reports the number of RecordingRequest objects in the list.
RecordingList		sortRecordingList(RecordingListComparator sortCriteria)	Creates a new RecordingList that contains all the elements of this list sorted according to the criteria specified by a RecordingListComparator.

Methods

contains(RecordingRequest)

```
public boolean contains(org.ocap.shared.dvr.RecordingRequest entry)
```

Tests if the indicated RecordingRequest object is contained in the list.

Parameters:

entry - The RecordingRequest object for which to search.

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

createRecordingListIterator()

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

Generates an iterator on the RecordingRequest elements in this list.

Returns: A RecordingListIterator on the RecordingRequests in this list.

filterRecordingList(RecordingListFilter)

```
public org.ocap.shared.dvr.navigation.RecordingList  
    filterRecordingList(org.ocap.shared.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 `RecordingRequest` 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 `RecordingRequest` 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.

getRecordingRequest(int)

```
public org.ocap.shared.dvr.RecordingRequest getRecordingRequest(int index)
```

Reports the `RecordingRequest` at the specified index position.

Parameters:

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

Returns: The `RecordingRequest` at the specified index.

Throws:

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

indexOf(RecordingRequest)

```
public int indexOf(org.ocap.shared.dvr.RecordingRequest entry)
```

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

Parameters:

`entry` - The `RecordingRequest` 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 `RecordingRequest` objects in the list.

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

sortRecordingList(RecordingListComparator)

```
public org.ocap.shared.dvr.navigation.RecordingList  
    sortRecordingList(org.ocap.shared.dvr.navigation.RecordingListComparator  
        sortCriteria)
```

Creates a new `RecordingList` that contains all the elements of this list sorted according to the criteria specified by a `RecordingListComparator`.

Parameters:

`sortCriteria` - the sort criteria to be applied to sort the entries in the recording list.

Returns: A sorted copy of the recording list.

org.ocap.shared.dvr.navigation RecordingListComparator

Declaration

```
public interface RecordingListComparator
```

Description

This interface represent a sorting criteria to be applied while sorting a RecordingList.

Member Summary

Methods

```
int compare(org.ocap.shared.dvr.RecordingRequest first,  
            org.ocap.shared.dvr.RecordingRequest second)  
    Compares two entries to check whether the first entry should be placed ahead of  
    the second entry in the iterator list.
```

Methods

compare(RecordingRequest, RecordingRequest)

```
public int compare(org.ocap.shared.dvr.RecordingRequest first,  
                  org.ocap.shared.dvr.RecordingRequest second)
```

Compares two entries to check whether the first entry should be placed ahead of the second entry in the iterator list.

Parameters:

`first` - the first entry to compare

`second` - the second entry to compare

Returns: positive integer if the first argument should be placed ahead of the second argument;
negative integer if the second argument should be placed ahead of the first entry; zero if the current order should be retained.

org.ocap.shared.dvr.navigation RecordingListFilter

Declaration

```
public abstract class RecordingListFilter
    java.lang.Object
    |
    +--org.ocap.shared.dvr.navigation.RecordingListFilter
```

Direct Known Subclasses: AppIDFilter, OrgIDFilter, RecordingStateFilter

Description

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

Member Summary	
Constructors	
protected	RecordingListFilter() Constructs the filter.
Methods	
abstract boolean	accept(org.ocap.shared.dvr.RecordingRequest entry) Tests if a particular entry passes this filter.
void	setCascadingFilter(RecordingListFilter filter) Provides a means to cascade filters.

Constructors

RecordingListFilter()

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

Methods

accept(RecordingRequest)

```
public abstract boolean accept(org.ocap.shared.dvr.RecordingRequest 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 RecordingRequest objects.
```

Parameters:

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

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

setCascadingFilter(RecordingListFilter)

```
public void setCascadingFilter(org.ocap.shared.dvr.navigation.RecordingListFilter  
    filter)
```

Provides a means to cascade filters. The accept method of this filter is called only for entries matching the specified filter. Multiple calls to this method will replace the previously set filter.

Parameters:

`filter` - the filter that will be applied before selecting the entries for which the accept() method is called. If the current filter is in the cascade chain of the filter passed in as the argument, this method does nothing.

org.ocap.shared.dvr.navigation RecordingListIterator

Declaration

```
public interface RecordingListIterator
```

Description

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

Member Summary

Methods

org.ocap.shared.dvr	getEntry(int index)	
.RecordingRequest		Gets the RecordingRequest object at the specified position.
int	getPosition()	Gets the current position of the RecordingListIterator.
int	getPosition(org.ocap.shared.dvr.RecordingRequest entry)	Gets the position of a specified recording request in the list.
RecordingList	getRecordingList()	Gets the recording list corresponding to this RecordingListIterator.
boolean	hasNext()	Tests if there is a RecordingRequest in the next position in the list.
boolean	hasPrevious()	Tests if there is a RecordingRequest in the previous position in the list.
org.ocap.shared.dvr	nextEntries(int n)	
.RecordingRequest []		Gets the next 'n' RecordingRequest objects in the list.
org.ocap.shared.dvr	nextEntry()	
.RecordingRequest		Gets the next RecordingRequest object in the list.
org.ocap.shared.dvr	previousEntries(int n)	
.RecordingRequest []		Gets the previous 'n' RecordingRequest objects in the list.
org.ocap.shared.dvr	previousEntry()	
.RecordingRequest		Gets the previous RecordingRequest object in the list.
void	setPosition(int index)	Sets the current position of the RecordingListIterator.
void	toBeginning()	Resets the iterator to the beginning of the list, such that hasNext() returns false and nextEntry() returns the first RecordingRequest 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 previousEntry() returns the last RecordingRequest in the list (if the list is not empty).

Methods

getEntry(int)

```
public org.ocap.shared.dvr.RecordingRequest getEntry(int index)
```

Gets the `RecordingRequest` object at the specified position. This method does not advance the current position within the list.

Parameters:

`index` - the position of the `RecordingRequest` to be retrieved.

Returns: the `RecordingRequest` at the specified position.

Throws:

`java.lang.IndexOutOfBoundsException` - if the index is greater than the size of the list.

getPosition()

```
public int getPosition()
```

Gets the current position of the `RecordingListIterator`. This would be the position from where the next `RecordingRequest` will be retrieved when an application calls the `nextEntry`.

Returns: the current position of the `RecordingListIterator`.

getPosition(RecordingRequest)

```
public int getPosition(org.ocap.shared.dvr.RecordingRequest entry)
```

Gets the position of a specified recording request in the list.

Parameters:

`entry` - The recording request for which the position is sought.

Returns: The position of the specified recording; -1 if the entry is not found.

getRecordingList()

```
public org.ocap.shared.dvr.navigation.RecordingList getRecordingList()
```

Gets the recording list corresponding to this `RecordingListIterator`.

Returns: the `RecordingList` corresponding to this iterator.

hasNext()

```
public boolean hasNext()
```

Tests if there is a `RecordingRequest` in the next position in the list.

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

hasPrevious()

```
public boolean hasPrevious()
```

Tests if there is a `RecordingRequest` in the previous position in the list.

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

nextEntries(int)

```
public org.ocap.shared.dvr.RecordingRequest[] nextEntries(int n)
```

Gets the next 'n' RecordingRequest objects in the list. This method also advances the current position within the list. If the requested number of entries are not available, the remaining elements are returned. If the current position is at the end of the iterator, this method returns an array with length zero.

Parameters:

n - the number of next entries requested.

Returns: an array containing the next 'n' RecordingRequest object from the current position in the list.

nextEntry()

```
public org.ocap.shared.dvr.RecordingRequest nextEntry()
```

Gets the next RecordingRequest object in the list. This method may be called repeatedly to iterate through the list.

Returns: The RecordingRequest object at the next position in the list.

Throws:

java.util.NoSuchElementException - If the iteration has no next RecordingRequest.

previousEntries(int)

```
public org.ocap.shared.dvr.RecordingRequest [] previousEntries(int n)
```

Gets the previous 'n' RecordingRequest objects in the list. This method also changes the current position within the list. If the requested number of entries are not available, the remaining elements are returned. If the current position is at the beginning of the iterator, this method returns an array with length zero.

Parameters:

n - the number of previous entries requested.

Returns: an array containing the previous 'n' RecordingRequest object from the current position in the list.

previousEntry()

```
public org.ocap.shared.dvr.RecordingRequest previousEntry()
```

Gets the previous RecordingRequest object in the list. This method may be called repeatedly to iterate through the list in reverse order.

Returns: The RecordingRequest object at the previous position in the list.

Throws:

java.util.NoSuchElementException - If the iteration has no previous RecordingRequest.

setPosition(int)

```
public void setPosition(int index)
    throws IndexOutOfBoundsException
```

Sets the current position of the RecordingListIterator. This would be the position from where the next RecordingRequest will be retrieved when an application calls the nextEntry.

Parameters:

index - the current position of the RecordingListIterator would be set to this value.

Throws:

`java.lang.IndexOutOfBoundsException` - if the index is greater than the size of the list.

toBeginning()

```
public void toBeginning()
```

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

org.ocap.shared.dvr.navigation RecordingStateFilter

Declaration

```
public class RecordingStateFilter extends RecordingListFilter
    java.lang.Object
    |
    |--org.ocap.shared.dvr.navigation.RecordingListFilter
    |
    |--org.ocap.shared.dvr.navigation.RecordingStateFilter
```

Description

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

Member Summary	
Constructors	
	<pre>RecordingStateFilter(int state)</pre> <p>Constructs the filter based on a particular state type (PENDING, FAILED, etc.).</p>
Methods	
boolean	<pre>accept(org.ocap.shared.dvr.RecordingRequest entry)</pre> <p>Tests if the given <code>org.ocap.shared.dvr.RecordingRequest</code> passes the filter.</p>
int	<pre>getFilterValue()</pre> <p>Reports the value of state used to create this filter.</p>

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.shared.dvr.RecordingRequest` instance.

Methods

accept(RecordingRequest)

```
public boolean accept(org.ocap.shared.dvr.RecordingRequest entry)
```

Tests if the given `org.ocap.shared.dvr.RecordingRequest` passes the filter.

Overrides: `accept` in class `RecordingListFilter`

Parameters:

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

Returns: `true` if `org.ocap.shared.dvr.RecordingRequest` contained within the `RecordingRequest` 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 H OCAP DVR Shared Media API (org.ocap.dvr.shared.media)

This section presents the `org.ocap.dvr.shared.media` API.

Package org.ocap.shared.media

Description

Extensions to JMF to support DVR playback.

Class Summary	
Interfaces	
<code>MediaTimeFactoryControl</code>	Provides the ability to obtain media times with various special characteristics when applied to the content being played by this JMF player.
<code>TimeLine</code>	Represents a transmitted time line.
<code>TimeLineControl</code>	Provides access to the transmitted timelines in a piece of content
<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.
Classes	
<code>BeginningOfContentEvent</code>	<code>BeginningOfContentEvent</code> is a <code>RateChangeEvent</code> that is posted when the rate change is due to a rewind hitting the beginning of the media, or due to the time-shift buffer reaching maximum depth.
<code>EndOfContentEvent</code>	<code>EndOfContentEvent</code> is a <code>RateChangeEvent</code> that is posted when the rate change is due to a forward playback hitting the end of the stored context, or a forward playback catching up with the live recording point.
<code>EnteringLiveModeEvent</code>	<code>EnteringLiveModeEvent</code> is a <code>ControllerEvent</code> that is posted when the controller has started playing back a live broadcast stream.
<code>LeavingLiveModeEvent</code>	<code>LeavingLiveModeEvent</code> is a <code>ControllerEvent</code> that is posted when the controller is not playing back a live broadcast stream anymore.
Exceptions	
<code>TimeLineInvalidException</code>	This exception is returned when a time line is no longer valid in a the piece of content for which it was obtained.
<code>TimeOutOfRangeException</code>	This exception is returned when a time or media time is outside the valid range for a particular time line.

org.ocap.shared.media BeginningOfContentEvent

Declaration

public class **BeginningOfContentEvent** extends javax.media.RateChangeEvent

```

java.lang.Object
|
+--java.util.EventObject
    |
    +--javax.media.ControllerEvent
        |
        +--javax.media.RateChangeEvent
            |
            +--org.ocap.shared.media.BeginningOfContentEvent
  
```

All Implemented Interfaces: javax.media.MediaEvent, java.io.Serializable

Description

BeginningOfContentEvent is a RateChangeEvent that is posted when the rate change is due to a rewind hitting the beginning of the media, or due to the time-shift buffer reaching maximum depth.

Member Summary

Constructors

```

BeginningOfContentEvent (javax.media.Controller from,
float newRate)
    Create a BeginningOfContentEvent.
  
```

Constructors

BeginningOfContentEvent(Controller, float)

```
public BeginningOfContentEvent (javax.media.Controller from, float newRate)
```

Create a BeginningOfContentEvent.

Parameters:

from - the controller that is generating the event.

org.ocap.shared.media EndOfContentEvent

Declaration

```
public class EndOfContentEvent extends javax.media.RateChangeEvent
```

```
java.lang.Object
|
+--java.util.EventObject
    |
    +--javax.media.ControllerEvent
        |
        +--javax.media.RateChangeEvent
            |
            +--org.ocap.shared.media.EndOfContentEvent
```

All Implemented Interfaces: javax.media.MediaEvent, java.io.Serializable

Description

EndOfContentEvent is a RateChangeEvent that is posted when the rate change is due to a forward playback hitting the end of the stored context, or a forward playback catching up with the live recording point.

Member Summary

Constructors

```
EndOfContentEvent(javax.media.Controller from, float
newRate)
    Create a EndOfContentEvent.
```

Constructors

EndOfContentEvent(Controller, float)

```
public EndOfContentEvent(javax.media.Controller from, float newRate)
```

Create a EndOfContentEvent.

Parameters:

from - the controller that is generating the event.

org.ocap.shared.media

EnteringLiveModeEvent

Declaration

```
public class EnteringLiveModeEvent extends javax.media.ControllerEvent
```

```

java.lang.Object
|
+--java.util.EventObject
    |
    +--javax.media.ControllerEvent
        |
        +--org.ocap.shared.media.EnteringLiveModeEvent

```

All Implemented Interfaces: javax.media.MediaEvent, java.io.Serializable

Description

EnteringLiveModeEvent is a ControllerEvent that is posted when the controller has started playing back a live broadcast stream. This event is sent to a registered ControllerListener in addition to any RateChangeEvent or MediaTimeSetEvent.

Member Summary

Constructors

```

EnteringLiveModeEvent (javax.media.Controller from)
    Create a EnteringLiveModeEvent.

```

Constructors

EnteringLiveModeEvent(Controller)

```
public EnteringLiveModeEvent (javax.media.Controller from)
```

Create a EnteringLiveModeEvent.

Parameters:

from - the controller that is generating the event.

org.ocap.shared.media LeavingLiveModeEvent

Declaration

```
public class LeavingLiveModeEvent extends javax.media.ControllerEvent
```

```

java.lang.Object
|
+--java.util.EventObject
    |
    +--javax.media.ControllerEvent
        |
        +--org.ocap.shared.media.LeavingLiveModeEvent

```

All Implemented Interfaces: javax.media.MediaEvent, java.io.Serializable

Description

LeavingLiveModeEvent is a ControllerEvent that is posted when the controller is not playing back a live broadcast stream anymore. This event is sent to a registered ControllerListener in addition to any RateChangeEvent or MediaTimeSetEvent.

Member Summary

Constructors

```
LeavingLiveModeEvent(javax.media.Controller from)
    Create a LeavingLiveModeEvent.
```

Constructors

LeavingLiveModeEvent(Controller)

```
public LeavingLiveModeEvent(javax.media.Controller from)
```

Create a LeavingLiveModeEvent.

Parameters:

`from` - the controller that is generating the event.

org.ocap.shared.media MediaTimeFactoryControl

Declaration

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

All Superinterfaces: javax.media.Control

Description

Provides the ability to obtain media times with various special characteristics when applied to the content being played by this JMF player. The behavior of these media times is implementation dependent if used with any other JMF player.

Member Summary

Methods

javax.media.Time	getRelativeTime(long offset)	Obtain a media time relative to the current location
javax.media.Time	setTimeApproximations(javax.media.Time original, boolean beforeOrAfter)	Enables applications to precisely control the position where playback starts following a call to Player.setMediaTime.

Methods

getRelativeTime(long)

```
public javax.media.Time getRelativeTime(long offset)
```

Obtain a media time relative to the current location

Parameters:

`offset` - the offset relative to the current location measured in nanoseconds

Returns: a media time

setTimeApproximations(Time, boolean)

```
public javax.media.Time setTimeApproximations(javax.media.Time original,  
boolean beforeOrAfter)
```

Enables applications to precisely control the position where playback starts following a call to Player.setMediaTime. This method takes an original media time as input and returns a new media time which encapsulates the original media time and an indication of how that original media time is to be interpreted when used in a call to Player.setMediaTime.

Parameters:

`original` - the original media time

`beforeOrAfter` - if true, the media time where playback starts will be on or before the original one (i.e. content at the original media time is guaranteed to be presented in playback). If false, the media time where playback starts will be after the original one (i.e. neither content at the original media time nor any content before that original time will be presented in playback).

Returns: a new media time

org.ocap.shared.media

TimeLine

Declaration

```
public interface TimeLine
```

Description

Represents a transmitted time line. Transmitted time lines start at one media time within a piece of content and finish at a later media time in that content. Transmitted time lines are valid at all media times between these points. They are either increment linearly or are paused. The value of a transmitted time line does not have any discontinuities.

Member Summary

Methods

javax.media.Time	getFirstMediaTime()	Returns the first media time at which this time line is valid.
	long getFirstTime()	Returns the first valid time in this time line.
javax.media.Time	getLastMediaTime()	Returns the last time at which this time line is valid.
	long getLastTime()	Returns the last valid time in this time line.
javax.media.Time	getMediaTime(long time)	Translates a time in this time line into the corresponding media time.
	long getTime(javax.media.Time mediatime)	Translates a media time into the corresponding time in this timeline

Methods

getFirstMediaTime()

```
public javax.media.Time getFirstMediaTime()
    throws TimeLineInvalidException
```

Returns the first media time at which this time line is valid. For a scheduled recording, this is the first point within the piece of content where the time line is valid. For a timeshift recording, if the time line starts within the time shift buffer then the media time where it starts will be returned. If the time line starts before the start of the time shift buffer, the media time of the start of the time shift buffer will be returned. Note that if the time shift buffer is full and time shift recording is in progress, the start of the buffer will be moving as newly written data over-writes the former start of the buffer.

Returns: a media time

Throws:

`TimeLineInvalidException` - if the time line is no longer valid in this piece of content. e.g. the piece of content is a time shift recording and the end of the time line is no longer within the buffer

getFirstTime()

```
public long getFirstTime()  
    throws TimeLineInvalidException
```

Returns the first valid time in this time line. For a scheduled recording, this is the first point within the piece of content where the time line is valid. For a timeshift recording, if the time line starts within the time shift buffer then the time where it starts will be returned. If the time line starts before the start of the time shift buffer, the time of the start of the time shift buffer will be returned. Note that if the time shift buffer is full and time shift recording is in progress, the start of the buffer will be moving as newly written data over-writes the former start of the buffer.

Returns: a time in this time line

Throws:

TimeLineInvalidException - if the time line is no longer valid in this piece of content. e.g. the piece of content is a time shift recording and the end of the time line is no longer within the buffer

getLastMediaTime()

```
public javax.media.Time getLastMediaTime()  
    throws TimeLineInvalidException
```

Returns the last time at which this time line is valid. For a scheduled recording, this is the last point within the piece of content where the time line is valid. For a timeshift recording, if the time line ends within the time shift buffer then the media time where it starts will be returned. If the time line ends after the end of the time shift buffer, the media time of the end of the time shift buffer will be returned. Note that if the time shift buffer is full and time shift recording is in progress, the end of the buffer will be moving as newly written data over-writes the former start of the buffer.

Returns: a media time

Throws:

TimeLineInvalidException - if the time line is no longer valid in this piece of content. e.g. the piece of content is a time shift recording and the end of the time line is no longer within the buffer

getLastTime()

```
public long getLastTime()  
    throws TimeLineInvalidException
```

Returns the last valid time in this time line. For a scheduled recording, this is the last point within the piece of content where the time line is valid. For a timeshift recording, if the time line ends within the time shift buffer then the media time where it end will be returned. If the media time ends after the end of the time shift buffer, the media time of the end of the time shift buffer will be returned. Note that if the time shift buffer is full and time shift recording is in progress, the end of the buffer will be moving as newly written data over-writes the former start of the buffer.

Returns: a time in this time line

Throws:

TimeLineInvalidException - if the time line is no longer valid in this piece of content. e.g. the piece of content is a time shift recording and the end of the time line is no longer within the buffer

getMediaTime(long)

```
public javax.media.Time getMediaTime(long time)
    throws TimelineInvalidException, TimeOutOfRangeException
```

Translates a time in this time line into the corresponding media time. If the time is one where the time line pauses, the returned media time shall be the highest media time corresponding to the time specified.

Parameters:

time - a time in this time line

Returns: the corresponding media time

Throws:

`TimelineInvalidException` - if the time line is no longer valid in this piece of content. e.g. the piece of content is a time shift recording and the end of the time line is no longer within the buffer

`TimeOutOfRangeException` - if the time specified is not within this timeline

getTime(Time)

```
public long getTime(javax.media.Time mediatime)
    throws TimelineInvalidException, TimeOutOfRangeException
```

Translates a media time into the corresponding time in this timeline

Parameters:

mediatime - a media time

Returns: the corresponding time in this timeline

Throws:

`TimelineInvalidException` - if the time line is no longer valid in this piece of content. e.g. the piece of content is a time shift recording and the end of the time line is no longer within the buffer

`TimeOutOfRangeException` - if the media time specified is not within this timeline

org.ocap.shared.media TimeLineControl

Declaration

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

All Superinterfaces: javax.media.Control

Description

Provides access to the transmitted timelines in a piece of content

Member Summary

Methods

```
TimeLine[] getTimeLines()  
Returns all the transmitted timelines found in a piece of content.
```

Methods

getTimeLines()

```
public org.ocap.shared.media.TimeLine[] getTimeLines()
```

Returns all the transmitted timelines found in a piece of content. If no transmitted timelines are present then an array of length 0 is returned.

Returns: an array of timelines

org.ocap.shared.media TimeLineInvalidException

Declaration

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

```
java.lang.Object
|
+--java.lang.Throwable
|
+--java.lang.Exception
|
+--org.ocap.shared.media.TimeLineInvalidException
```

All Implemented Interfaces: java.io.Serializable

Description

This exception is returned when a time line is no longer valid in a the piece of content for which it was obtained. For example, the piece of content is a time shift recording and the end of the time line is no longer within the buffer.

Member Summary

Constructors

<pre>TimeLineInvalidException() Constructs a TimeLineInvalidException with no detail message</pre>
--

Constructors

TimeLineInvalidException()

```
public TimeLineInvalidException()
```

Constructs a TimeLineInvalidException with no detail message

org.ocap.shared.media TimeOutOfRangeException

Declaration

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

```
java.lang.Object
|
+--java.lang.Throwable
|
|   +--java.lang.Exception
|   |
|   |   +--org.ocap.shared.media.TimeOutOfRangeException
```

All Implemented Interfaces: java.io.Serializable

Description

This exception is returned when a time or media time is outside the valid range for a particular time line.

Member Summary

Constructors

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

Constructors

TimeOutOfRangeException()

```
public TimeOutOfRangeException()
```

Constructs a TimeOutOfRangeException with no detail message

org.ocap.shared.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 is 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	getDuration()	Get the duration of content currently in the time-shift buffer.
javax.media.Time	getEndOfBuffer()	Get the media time corresponding to the end of the time-shift buffer.
javax.media.Time	getMaxDuration()	Get the estimated value for the maximum duration of content that could be buffered using this 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.

getDuration()

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

Get the duration of content currently in the time-shift buffer. The value returned is the content's duration when played at a rate of 1.0.

Returns: A Time object representing the duration.

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.

getMaxDuration()

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

Get the estimated value for the maximum duration of content that could be buffered using this time-shift buffer. The value returned is the content's duration when played at a rate of 1.0.

Returns: A Time object representing the maximum value for duration.

This page intentionally left blank.

Appendix I Revisions (informative)

I.1 ECNs included in OC-SP-OCAP-DVR-I02-050524

Table I-1 ECNs Incorporated in OC-SP-OCAP-DVR-I02-050524

ECN	Date Accepted	Summary
OCAP-DVR-N-04.0647-3	8/23/04	Improving design of RecordingManager.record().
OCAP-DVR-N-04.0657-1	8/23/04	Support for JMF controls not specified when playing recorded content.
OCAP-DVR-N-04.0658-1	8/23/04	Support for recorded content playback via JMF.
OCAP-DVR-N-04.0695-2	11/2/04	Time shift options (All Java).
OCAP-DVR-N-04.0707-1	11/18/04	Recorded Applications.
OCAP-DVR-N-04.0719-1	1/6/05	AES Security Change
OCAP-DVR-N-05.0738-3	5/3/05	Establishment of a Common Core.

