Getting Started with the Dolby® Screen Management Interface Java API

Issue 1

Dolby Laboratories, Inc.

Corporate Headquarters

Dolby Laboratories, Inc.
100 Potrero Avenue
San Francisco, CA 94103-4813 USA
Telephone 415-558-0200
Fax 415-863-1373
www.dolby.com

European Headquarters

Dolby Laboratories, Inc.
Wootton Bassett
Wiltshire SN4 8QJ England
Telephone 44-1793-842100
Fax 44-1793-842101

DISCLAIMER OF WARRANTIES:

EQUIPMENT MANUFACTURED BY DOLBY LABORATORIES IS WARRANTED AGAINST DEFECTS IN MATERIALS AND WORKMANSHIP FOR A PERIOD OF ONE YEAR FROM THE DATE OF PURCHASE. THERE ARE NO OTHER EXPRESS OR IMPLIED WARRANTIES AND NO WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, OR OF NONINFRINGEMENT OF THIRD-PARTY RIGHTS (INCLUDING, BUT NOT LIMITED TO, COPYRIGHT AND PATENT RIGHTS).

LIMITATION OF LIABILITY:

IT IS UNDERSTOOD AND AGREED THAT DOLBY LABORATORIES' LIABILITY, WHETHER IN CONTRACT, IN TORT, UNDER ANY WARRANTY, IN NEGLIGENCE, OR OTHERWISE, SHALL NOT EXCEED THE COST OF REPAIR OR REPLACEMENT OF THE DEFECTIVE COMPONENTS OR ACCUSED INFRINGING DEVICES, AND UNDER NO CIRCUMSTANCES SHALL DOLBY LABORATORIES BE LIABLE FOR INCIDENTAL, SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, DAMAGE TO SOFTWARE OR RECORDED AUDIO OR VISUAL MATERIAL), COST OF DEFENSE, OR LOSS OF USE, REVENUE, OR PROFIT, EVEN IF DOLBY LABORATORIES OR ITS AGENTS HAVE BEEN ADVISED, ORALLY OR IN WRITING, OF THE POSSIBILITY OF SUCH DAMAGES.

Issue 1
S07/18483

Getting Started

This document introduces the Java application programming interface (API) for the Dolby® Screen Management Interface. The API enables you to:

- Control Dolby Digital Cinema presentation systems.
- Manage content ingest and movement.
- Conduct in-theatre operations such as dimming lights, closing curtains, and so on.
- Remotely monitor and control theatre operations from a Network Operations Center (NOC) through Simple Network Management Protocol (SNMP).

Developers should read this document before using the Screen Management Interface Java API to learn basic information on how to:

- Configure and install the Screen Management Interface.
- Use major interfaces in the `DolbyClientServices` interface, the starting point for Screen Management Interface development.
- Understand listener events in the Screen Management Interface.

Failure to read this document and the information provided in the Dolby Screen Management Interface Java API could result in errors during code development.

The classes in the Java API for the Screen Management Interface consist of three major packages. Table 1 describes each package.

Table 1 Screen Management Interface Packages

Package	Contents
`com.dolby.smi.api`	Interfaces and classes that communicate with the server.
`com.dolby.smi.exception`	Exception classes that indicate a deviation from the normal flow of program execution.
`com.dolby.util`	Miscellaneous utility classes such as `Duration` interface, which represents a length of time.

The com.dolby.smi.api package contains all the interfaces and classes that control a Dolby Digital Cinema playback device. One of the interfaces is `DolbyClientServices`, which is contained in the `DolbyClientServicesLibrary`. You must start with `DolbyClientServices` to write programs that manage a Dolby Digital Cinema playback device. It provides access to other interfaces that control major Screen Management Interface functionality.

Figure 1 shows some of the main interfaces of the `DolbyClientServicesLibrary` class hierarchy. It reveals that:

- `DolbyClientServicesLibrary` is a container for the `DolbyClientServices` interface.
- `DolbyClientServices` interface is a container for `ShowLibrary`, `ClipLibrary`, `ServerStatusMonitor`, `Scheduler`, `LicenseLibrary`, and `AuditoriumLibrary`.
- `Auditorium` represents a single screen that is used to obtain and manipulate Dolby Digital Cinema devices, such as `ContentPlayer` and `ContentManager`.

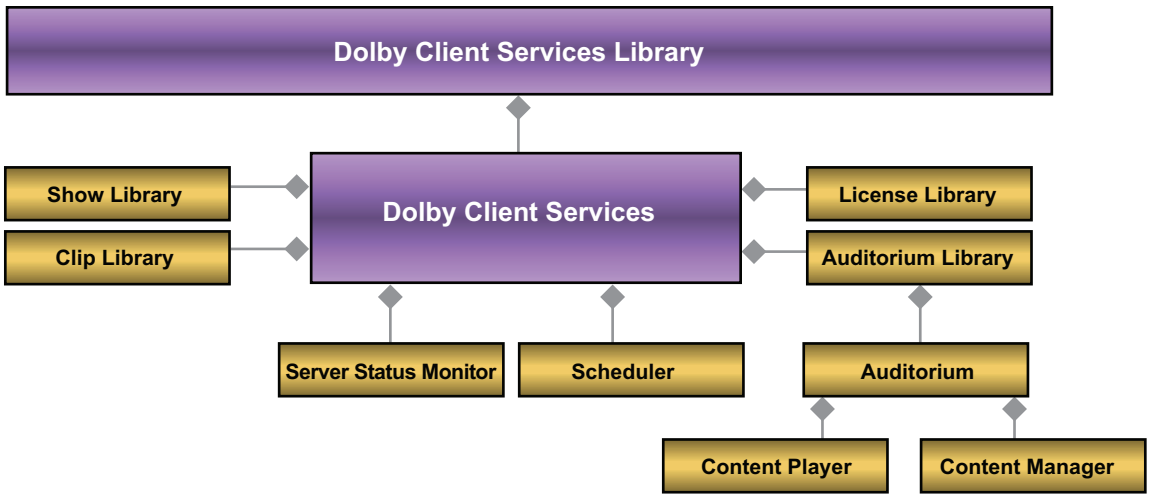

Figure 1 DolbyClientServicesLibrary Class Hierarchy

The class architecture in Figure enables a theatre management system to remotely access functions of Dolby Digital Cinema devices through the Screen Management Interface. For example, a user from the Screen Management Interface could remotely get the IP address or the name of a Dolby Digital Cinema device.

The Screen Management Interface provides three ways to receive information and control a Dolby Digital Cinema playback device:

- Get methods (pull)—Provide information on the status of an object on a Dolby Digital Cinema device.
- Action methods—Enable an API user to control Dolby Digital Cinema devices.
- Listener events (push)—Provide information on changes to an object on a Dolby Digital Cinema device to registered handlers. Each interface uses listener events to receive information from one or more event objects.

Section 2 discusses each `DolbyClientServices` interface in Figure 1. Each section describes how each interface uses get methods, action methods, and listener events to manage and control a Dolby Digital Cinema device.

Before you can use the Screen Management Interface, you must know how to install and configure it. See Section 1 for information on how to perform these tasks.

Note: For information on all the interfaces and classes, see the Dolby Screen ManagementInterface Java API.

1 Installation and Configuration

The *Java Dolby Screen Management Interface Software Development Kit (SDK)* contains documentation, libraries, configuration files, and sample code. This section provides information on how to install and configure the Screen Management Interface on your system.

1.1 Installation

Table 2 describes the hardware, software, and network requirements necessary to run the Screen Management Interface.

Table 2 Screen Management Interface System Requirements

Hardware	Software	Network
Recommendation A machine with at least 1 GB of RAM	**Version** Java 1.5. JRE or higher	**Configuration** Intranet setup capable of serving HTTP traffic (default port is 8080)
	Operating System Any operating system that supports Java 1.5 JRE	

1.2 Configuration

You must perform these tasks to configure the Screen Management Interface to manage devices:

- Copy the contents of .lib to a directory accessible to your application.
- Copy the contents of .etc to the directory where you start your application.
- Add `smi-api.jar` to your class path. The manifest references the other required .jar files. As long as they are in the same directory as `smi-api.jar` all the necessary classes should be found with no further additions to your class path.
- Add the following argument to the command line used to start your application: `-Djava.security.auth.login.config=auth.conf`.
- Set up the smi.properties file: The IP address and the name of the Screen Management Interface server reside in the smi.properties file in the class path directory. To connect to one or more Screen Management Servers, you can modify the smi.properties file, which contains a list of either static IP addresses or server names. Specifying the IP address enables the Screen Management Interface to explicitly know the location of

the server. Specifying the server name enables the Screen Management Interface to dynamically discover the server location through the Service Location Protocol (SLP).

Note: For every entry in the smi.properties file, an instance of `DolbyClientServices` is created in a single `DolbyClientServiceLibrary`.

Note: Be sure to configure your network to allow multicast traffic between your application and the Dolby Digital Cinema server.

Note: All of the required .jar files are found in the .lib directory. Additional configuration files are found in .etc directory.

2 DolbyClientServices Interface

This section describes the major interfaces contained in the `DolbyClientServices` interface (see Figure 1). Each section defines the function of the interface and describes how the interface uses get methods, action methods, and listener events to control and manage Dolby Digital Cinema devices.

The listener event descriptions describe the listener event and the event objects. For more information about listener events, see Section 3.

Note: This section contains only a few of the interfaces, methods, and event objects in the Screen Management Interface API. For a complete list and description, see the Dolby Screen Management Interface Java API.

2.1 Interface ServerStatusMonitor

`ServerStatusMonitor` checks for changes to the status of the Screen Management Interface server.

The Screen Management Interface can register with this class to find out when the Screen Management Interface server is available or when it becomes disconnected. It must indicate ready state before the libraries are used.

Get Methods

Table 3 describes one of the get methods used to retrieve information about the status of the server.

Table 3 ServerStatusMonitor Get Method

Get Method
`ServerStatus getServerStatus()` Returns the current status of the Screen Management Interface server in this session.

Listener Event

Table 4 describes one of the listener events and event objects used to receive updates when the status of the Screen Management Interface server changes.

Table 4 ServerStatusMonitor Listener Event

Event Listener	Event Object
`ServerStatusListener` Serves as the interface to be implemented by classes to get updates on the Screen Management Interface server status.	`ServerStatus` Indicates the state of the Screen Management Interface server.

2.2 Interface AuditoriumLibrary

`AuditoriumLibrary` provides access to all auditoriums in the Screen Management Interface server.

Get Method

Table 5 describes one of the get methods used to find all the auditoriums in a Screen Management Interface server.

Table 5 AuditoriumLibrary Get Method

Get Method
`Auditorium[] findAll()` Returns all the auditoriums in the Screen Management Interface server.

Listener Event

Table 6 describes one of the listener events and event objects used to find all the auditoriums in the Screen Management Interface server.

Table 6 AuditoriumLibrary Listener Event

Event Listener	Event Object
`PersistenceEventListener` Handles persistence events. Implement this interface to find out when things change in the persistent store.	`PersistenceRegion. AUDITORIUM` Indicates if the auditorium was added, deleted, or modified.

2.3 Interface Auditorium

An auditorium is a single screening room that contains cinema equipment, such as a content player.

Get Methods

Table 7 describes some of the get methods used to get information about content in an auditorium.

Table 7 Auditorium Get Methods

Get Methods
`ContentManager getContentManager()` Returns the content manager for this auditorium.
`ContentPlayer getContentPlayer()` Returns the content player for this auditorium.

Listener Event

Table 8 describes one of the listener events and event objects used to get information about content in an auditorium.

Table 8 Auditorium Event Listener

Event Listener	Event Object
`PersistenceEventListener` Handles persistence events. Implement this interface to find out when things change in the persistent store.	`PersistenceRegion. AUDITORIUM` Indicates if the auditorium was added, deleted, or modified.

2.4 Interface ContentPlayer

`ContentPlayer` controls and monitors a Dolby Digital Cinema playback device.

Get Methods

Table 9 describes some of the get methods used to manage content on a device.

Table 9 ContentPlayer Get Methods

Get Methods
`TransportState getTransportState()` Returns the current transport state of the player such as play, stop, and so on.
`Clip getClip()` Returns the selected clip for playback.
`Clip getShow()` Returns the currently selected show for playback.
`Duration getPositionInClip()` Returns the current playback position relative to the start of the current clip.

Action Methods

Table 10 describes some of the action methods used to manage content on a device.

Table 10 ContentPlayer Action Methods

Action Methods
`play()` Starts playback of the currently selected show or clip from the current position. Has no effect if no content is selected or if the media player is already playing.
`stop()` Stops playback and displays a black screen. Has no effect if no content is selected or if playback is already stopped.
`pause()` Pauses playback of the currently selected show from the current position.
`goToPositionInShow(Duration pPosition)` Sets the playback to a position in the current show. If the player is playing, it continues playing from the new position. If the player is stopped, it remains stopped. Use this method for fast forward and rewind functions.
`setCurrentShow(Show pShow)` Sets the show to be played. Deselects the selected show if `pShow` is null.
`setAutoSelect(boolean pAutoSelect)` Configures the player to automatically select scheduled shows. Note that shows start automatically only when both this property is set to true and the ServerSettings are set up for automatic operation. Use this method to set schedule mode to Auto or Manual.

Listener Events

Table 11 describes the listener events and event objects used to receive updates about content on a device.

Table 11 ContentPlayer Listener Events

Event Listener	Event Object
`DeviceEventListener` Handles device events. Implement this interface to find out changes to the device.	`DeviceEventType. PLAYER_AUTO_SELECT` Indicates whether the autoselect setting of a content player has changed.
	`DeviceEventType. PLAYER_CONTENT_POSITION` Indicates the position where the content was changed.
	`DeviceEventType. PLAYER_CONTENT_SELECTION` Indicates that different content was selected.
	`DeviceEventType. PLAYER_NEXT_CUE` Indicates the next cue to be triggered in the playing show has changed.
	`DeviceEventType. PLAYER_PERFORMANCE_SELECTION` Indicates a different performance (showtime) was selected.
	`DeviceEventType. PLAYER_TRANSPORT_STATE` Indicates the transport state of a content player has changed.

2.5 Interface ShowLibrary

`ShowLibrary` enables you to access all installed shows on the same Screen Management Interface server. You use `ShowLibrary` to search for shows, find all the shows that match specific criteria, and create shows.

Get Method

Table 12 describes some of the get methods used to access all installed shows on the same Screen Management Interface server.

Table 12 ShowLibrary Get Method

Get Methods
`Show[] findAll()` Returns all the shows in the Screen Management Interface server.

Action Method

Table 13 describes one of the action methods used to manage shows on the same Screen Management Interface server.

Table 13 ShowLibrary Action Method

Action Methods
`DigitalShow.createDigitalShow(String pName, Duration pEstimatedDuration)` Creates an empty digital show.

Listener Event

Table 14 describes one of the listener events and event objects used to receive updates on all installed shows on the same Screen Management Interface server.

Table 14 ShowLibrary Listener Event

Event Listener	Event Object
`PersistenceEventListener` Handles persistence events. Implement this interface to find out when changes occur in the persistent store.	`PersistenceRegion. SHOW` Indicates if a show was added, deleted, or modified.

Tip: For an example of how to use the `ShowLibrary` interface, see ShowLibraryTestCase.java in the sample folder in the Java Docs API directory (dolby-smi\sample\src\com\dolby\smi\api\sample).

2.6 Interface LicenseLibrary

`LicenseLibrary` enables you to access licenses on the same Screen Management Interface server.

Get Method

Table 15 describes one of the get methods used to access licenses.

Table 15 LicenseLibrary Get Method

Get Method
`License[] findAll()` Returns all the licenses in the Screen Management Interface server.

Listener Event

Table 16 describes one of the listener events and event objects used to receive updates about licenses.

Table 16 LicenseLibrary Listener Event

Event Listener	Event Object
`PersistenceEventListener` Handles persistence events. Implement this interface to find out when things change in the persistent store.	`PersistenceRegion.LICENSE` Indicates if a license was added, deleted, or modified.

Tip: For an example of how to use the `LicenseLibrary` interface, see LicenseLibraryTestCase.java in the sample folder in the Java Docs API directory (dolby-smi\sample\src\com\dolby\smi\api\sample).

2.7 Interface Scheduler

`Scheduler` enables you to schedule shows for presentation in auditoriums at specified times and provides access to scheduled shows.

Get Method

Table 17 describes one of the get methods used to find all scheduled shows on the Screen Management Interface server.

Table 17 Scheduler Get Method

Get Method
`ShowTime[] findAll()` Returns scheduled shows for presentation in auditoriums at specified times.

Action Method

Table 18 describes one of the action methods used to schedule shows on the Screen Management Interface server.

Table 18 Scheduler Action Method

Action Method
`ShowTime.schedule(Auditorium pAuditorium, Show pShow, Date pStartTime)` Schedules a show in a specified auditorium at a specified time.

Listener Event

Table 19 describes one of the listener events and event objects used to receive updates of changes to the show schedule.

Table 19 Scheduler Listener Event

Event Listener	Event Object
`PersistenceEventListener` Handles persistence events. Implement this interface to find out when things change in the persistent store.	`PersistenceRegion.SHOW_TIME` Indicates if the time of the show was added, deleted, or modified.

Tip: For an example of how to use the `Scheduler` interface, see ShowAndScheduleTestCase.java in the sample folder in the Java Docs API directory (dolby-smi\sample\src\com\dolby\smi\api\sample).

2.8 Interface ClipLibrary

`ClipLibrary` provides access to clips in the Screen Management Interface server. This interface can be used to search for specific clips, find all the clips that match particular criteria, and so on.

Get Methods

Table 20 describes some of the get method used to get clips from the Screen Management Interface server.

Table 20 ClipLibrary Get Methods

Get Methods
`Clip[] findAll()` Returns all the clips in the Screen Management Interface server. Use this method to list all ingested composition playlists (CPLs).
`String Clip.getCompositionPlaylist()` Returns the CPL for this clip.

Listener Event

Table 21 describes one of the listener events and event objects used to receive updates about clips on the Screen Management Interface server.

Table 21 ClipLibrary Listener Event

Event Listener	Event Object
`PersistenceEventListener` Handles persistence events. Implement this interface to find out when things change in the persistent store.	`PersistenceRegion. CLIP` Indicates if the clip was added, deleted, or modified.

Tip: For an example of how to use the `ClipLibrary` interface, see ClipLibraryTestCase.java in the sample folder in the Java Docs API directory (dolby-smi\sample\src\com\dolby\smi\api\sample).

2.9 Interface ContentManager

`ContentManager` enables you to manage and transfer content in an auditorium. You can use the `ContentManager` interface to:

- Transfer content between two Dolby Show Stores or between a Dolby Show Store and a Dolby Show Library in a theatre management system.
- Transfer content from an FTP server to a Dolby Show Store or Dolby Show Library.
- Transfer local content between removable media and a Dolby Show Store (or Dolby Show Library).

Get Methods

Table 22 describes some of the get methods used to receive updates about content on a device.

Table 22 ContentManager Get Methods

Get Methods
`ContentTransfer findTransferByUUID(String pTransferUUID)` Returns a content transfer that is identified by the specified universally unique identifier (UUID).
`ContentTransfer[] getTransfers()` Returns the content transfers that were started in this `ContentManager`.

Action Methods

Table 23 describes some of the action methods used to receive updates about content on a device.

Table 23 ContentManager Action Methods

Action Methods
`ContentTransfer[] startTransfer(ClipInstance[] pClipInstances)` Initiates the transfer jobs of clips to this content manager so content can be transferred between Show Stores and the Show Library.
`ContentTransfer startTransfer(TransferableContent pContent)` Initiates the transfer jobs to this content manager so content can be transferred between an external FTP server and a Show Store or Show Library.

Action Methods
`ContentTransfer[] startTransfer(ClipInstance[] pClipInstances, ContentStoreType pDestinationContentStoreType)` Initiates the transfer jobs of clips to this content manager so content can be transferred from removable media to a Dolby Show Store or Dolby Show Library.

Listener Event

Table 24 describes some of the listener events and event objects used to receive updates about clips on a device.

Table 24 ContentManager Listener Event

Event Listener	Event Objects
`DeviceEventListener` Handles device events. Implement this interface to find changes to the persistent store.	`DeviceEventType CONTENT_STORES_CHANGED` Indicates that the content stores of the player or manager have changed. (Examples of content stores are RAID disks, USB flash drives, and so on.)
	`DeviceEventType CONTENT_TRANSFER_ADDED` A new transfer has started.
	`DeviceEventType CONTENT_TRANSFER_CHANGED` The state, such as size copied, or percentage complete of a content transfer changed.
	`DeviceEventType CONTENT_TRANSFER_DELETED` A content transfer was cleared and removed.
	`DeviceEventType CONTENT_TRANSFER_STATUS_CHANGED` The status of a transfer changed.

Tip: For an example of how to use the `ContentManager` interface to transfer content from an FTP server to a Dolby Show Store or Dolby Show Library, see ContentTransferTestCase.java in the sample folder in the Java Docs API directory (dolby-smi\sample\src\com\dolby\smi\api\sample).

Tip: For an example of how to use the `ContentManager` interface to transfer content between two Dolby Show Stores or between a Dolby Show Store and a Dolby Show Library in a theatre management system, see ScreenToScreenTransferCase.java (dolby-smi\sample\src\com\dolby\smi\api\sample).

3 Screen Management Events

This section details how the Screen Management Interface uses listener events. Listener events notify users of the Screen Management Interface when changes occur to objects on a Dolby Digital Cinema device.

The Screen Management Interface has two general types of events: persistence and device.

Note: When using events, remember that events must be handled faster than they can be received. If events are not handled in a 1-minute time frame, then the event will be canceled by the Screen Management Interface. A set of events will be published at a maximum rate of every half second.

3.1 Persistence Events

Persistence events are generated for objects in a persistent store that remain in the system after it has been restarted.

Table 25 describes the main classes and interfaces associated with persistence events.

Table 25 Persistence Event Classes

Class/Interface	Description
`PersistenceEvent`	Generated by objects that are persisted in the persistent store.
`PersistenceEventConstraints`	Constrains persistence events. Set the properties to constrain and leave the remaining fields at their default values.
`PersistenceEventListener`	Handles persistence events. Implement this interface to find out when changes occur in the persistent store.
`PersistenceEventType`	Identifies the type of persistence event such as `ADDED` and `DELETED`.
`PersistenceRegion`	Identifies the type of data referred to by a persistence event such as `SHOW, SHOW_TIME`.
`Persistent`	Represents the base interface for all classes that will send persistent events.

This discussion centers on the `PersistenceEvent` class, which users of the Screen Management Interface use to discover updates to persistent data objects.

PersistenceEvent Class

Objects that persist in the persistent store generate persistence events. Objects send persistence events when they are created, updated, or deleted in the persistent store. They also send persistence events when the relationship between two persistent objects changes.

The `PersistenceEvent` class uses these objects to get information when persistent data changes in the persistent store:

- `PersistenceEventType`
- `PersistenceRegion`
- `Id`

PersistenceEventType

This class identifies what action occurred to the object. It can indicate whether an object was added, modified, or deleted. It can also indicate if the local copy of the object is stale or has been evicted.

Class	Description
`PersistenceEventType ADDED`	An object was added to the local cache. This event is sent when another SMI adds an object to the persistent store. It is also sent when an existing object that had not been cached locally is added to the local cache
`PersistenceEventType DELETED`	An object was deleted.
`PersistenceEventType MODIFIED`	An object was modified.
`PersistenceEventType STALE`	The local copy of the object is out of date because another Screen Management Interface instance modified the same object.
`PersistenceEventType EVICTED`	An object was evicted from the local cache, although it is still present in the persistent store. No further events will be received from the object.

PersistenceRegion

This class specifies the region of the data indicated by the persistence event.

Examples:

- `PersistenceRegion CLIP`—Identifies the clip.
- `PersistenceRegion AUDITORIUM`—Identifies the auditorium.
- `PersistenceRegion SHOW_TIME`—Identifies the time the show is scheduled.

Note: To identify and get all regions, use `PersistenceRegion[] getAll()`.

ID

This class uniquely identifies an object in a region. It can be used to determine what specifically changed about the data.

3.2 Device Events

Device events refer to events generated by changes to a device.

Table 26 describes the main classes associated with device events.

Table 26 Device Event Classes

Class	Description
`DeviceEvent`	Represents the event sent by a device when its state changes.
`DeviceEventConstraints`	Constrains device events. Set the properties to constrain and leave the remaining fields at their default values.
`DeviceEventListener`	Handles device events. Implement this interface to find out changes to the device.
`DeviceEventSource`	Represents the base interface for all classes that will send device events.
`DeviceEventType`	Identifies which device event occurred.

This discussion centers on the `DeviceEvent` class, which is the mechanism the Screen Management Interface uses to discover updates to a device.

DeviceEvent class

When the state of a device changes, it sends an event. The `DeviceEvent` class uses two methods to indicate what changes occurred to a device:

- `DeviceEventType`—Identifies which device event occurred and indicates what changed on the device.
- `Device`—Acts as the reference to the updated device.

DeviceEventType

This class identifies the specific device event that occurred. It can indicate the status of audio, automation cues, content transfers, devices, players, and projectors.

Examples:

- `DeviceEventType AUDIO_MUTE`—The audio processor was muted or unmuted.
- `DeviceEventType AUTOMATION_AVAILABLE_CUES`—The available automation cues changed.
- `DeviceEventType CONTENT_TRANSFER_ADDED`—A new transfer has started.
- `DeviceEventType DEVICE_ADDED`—A device was added.
- `DeviceEventType PLAYER_CONTENT_POSITION`—The position in the content changed.
- `DeviceEventType PLAYER_CONTENT_SELECTION`—Different content was selected.
- `DeviceEventType PROJECTOR_LAMP`—The projector's lamp was turned on or off.

Device

This interface enables you to retrieve the new state of a device. Depending on the `DeviceEventType`, a Screen Management Interface developer should class cast a device to a specific device. This action enables the developer to get the specific device when its state, such as playback position, has changed .