Cisco Conductor SRM

API Guide

Please Read

Important

Please read this entire guide. If this guide provides installation or operation instructions, give particular attention to all safety statements included in this guide.

Notices

Trademark Acknowledgments

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. A listing of Cisco's trademarks can be found at **www.cisco.com/go/trademarks**.

Third party trademarks mentioned are the property of their respective owners.

The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1009R)

Publication Disclaimer

Cisco Systems, Inc. assumes no responsibility for errors or omissions that may appear in this publication. We reserve the right to change this publication at any time without notice. This document is not to be construed as conferring by implication, estoppel, or otherwise any license or right under any copyright or patent, whether or not the use of any information in this document employs an invention claimed in any existing or later issued patent.

Copyright

Contents

About This Guide

Purpose

This document defines the software functional specifications for the session and resource management (SRM) support subsystem for Cisco Conductor.

Audience

The audience for this document includes system administrators, operators, and installation engineers who deploy Videoscape Conductor systems.

Document Version

This is the first formal release of this document.

1

Session State Manager Usage Guide

Introduction

This chapter provides the design approach and describes the architecture for developing Cisco Conductor software with high availability and scale.

In This Chapter

Session Management Introduction

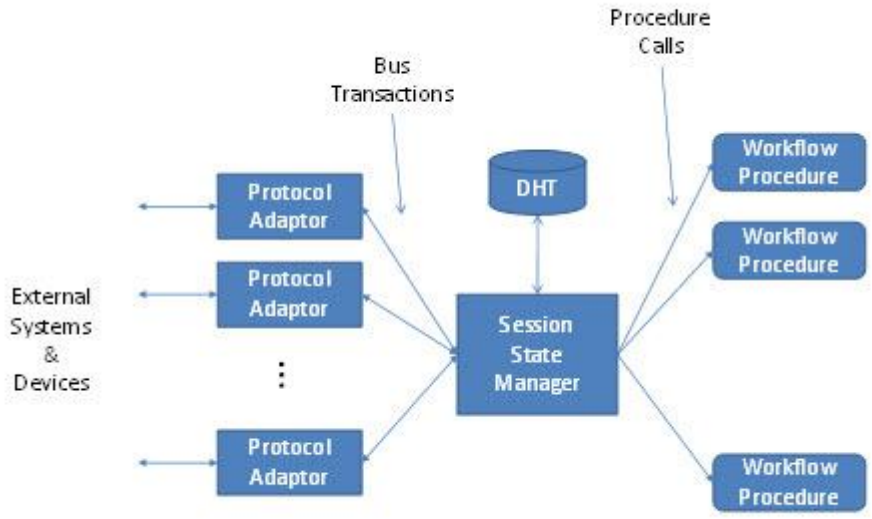
This section defines the software functional specifications for the session and resource management (SRM) support subsystem on Cisco Conductor. On Conductor, a specific session manager application is not provided. Instead software infrastructure enables development of session management applications using simple plug-in workflow procedures developed in c/c++ or python. A Resource Broker subsystem with associated Resource Adaptors to interface to specific resource devices is provided to work in conjunction with the session management subsystem.

Session manager applications are atypical software applications in terms of requirements. Because session manager applications are typically used to set up services for users on demand, the fundamental requirements are speed, scale, high availability, and interface to external third-party systems. These requirements are accomplished using a variety of techniques that add complexity to the application. For example, high availability is usually accomplished using hot-standby systems that mirror the state of active servers in standby devices. The result is complexity, as well as extensive operations and management personnel support requirements. The Conductor SRM architecture simplifies session management applications by providing facilities that accomplish the requirements using a horizontal scaling approach.

Modern Internet applications use a different approach to meet these requirements that dramatically simplifies the problems of scale and availability. These applications accomplish scale and availability using a horizontal scaling approach whereby multiple stateless instances operate in parallel to share the load. These approaches are based on the Representational State Transfer (REST) technique where clients maintain state enabling stateless software server instances to process transaction steps. The Conductor session management subsystem software enables horizontal scaling using a Session State Manager (SSM) to manage and maintain state as a service for session management applications. The SSM uses the services of a Distributed Hash Table (DHT) datastore to cache session state in the form of session state records (SSR). The SSR is an XML document that maintains the complete history of all states for a transaction and provides the relevant state information to the stateless instance procedures that form the basis for a specific session management application.

A high-level block diagram of the SSM approach is shown in the following diagram. An internal XMPP bus provides an internal messaging fabric. Protocol Adaptors (PA) convert incoming/outgoing events and messages to the internal bus format. The SSM receives incoming events, fetches the associated session state record and invokes the workflow procedure configured for the received event. This approach enables completely stateless server instances to process steps of a transaction which, in turn, enables horizontal scaling with the resultant simplification of software development and operations. The approach also simplifies the software development since state management and persistence is provided by the SSM.

Figure 1: SSM High Level Block Diagram

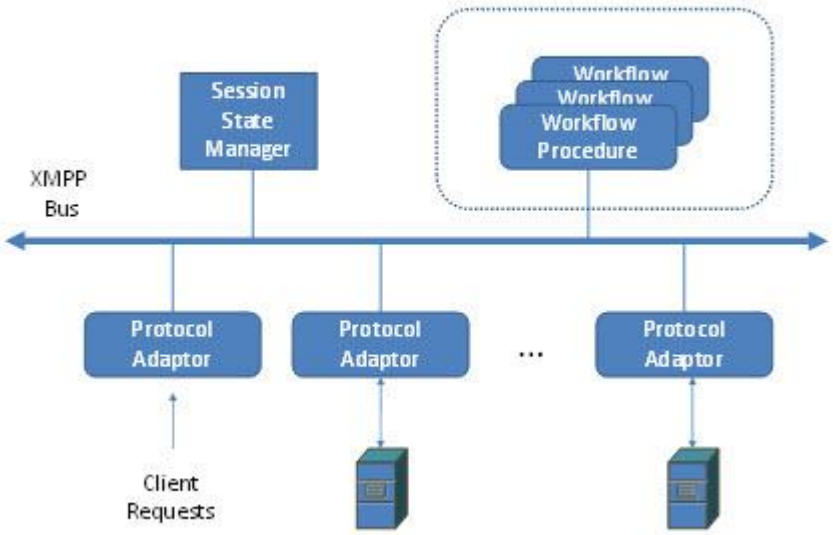


The SSM and the DHT are provided as Conductor infrastructure components. Protocol Adaptors and Workflow Procedures represent application-specific software deliverables. Protocol Adaptors are created as component instances by name. Workflow Procedures are configured in an event mapping table in the SSM. The SSM locates the Workflow Procedures and creates dynamic linkages to those components.

In the current release, workflow for session management applications is supported using the Workflow Procedure plug-in model described here. In later releases, this is supplemented with a full JSR223-compliant workflow engine. The existing model is used for legacy applications, but new session management applications will be based on the workflow engine. For newer applications, the SSM invokes the workflow engine instead of plug-in procedures.

Software components use the XMPP bus for component-to-component communication. A block diagram showing component communication is shown in the following diagram. Protocol Adaptors interface to external devices and systems. The design approach is that the PA is merely a protocol translator. Incoming protocol messages or events are decoded in their entirely into an XML stanza for transport via XMPP to the SSM. Conversely, messages received by the Protocol Adaptor are converted to the external protocol. Workflow Procedures are simple single-step transaction functions that process one phase of a transaction using the supplied SSR for state information. The Workflow Procedure typically generates an XML stanza corresponding to the next phase of a transaction that is relayed to the Workflow Procedure or Protocol Adaptor needed for the next step of the transaction. For example, upon receiving a client request, the next step is typically a call to an authorization system using the services of a Protocol Adaptor to send the request to the authorization system.

Figure 2: Software Component Interconnect



Session Management Transaction Processing

Session management is typically a multi-step sequence of steps or phases needed to authorize, allocate, and grant a requested service. A session setup for a user is typically initiated by a request from a client device. Upon receiving the request, the session manager implements a multi-step process interfacing to multiple external and internal systems to grant the requested service. As the steps are accomplished, state data are generated and used in subsequent steps to accomplish the session setup.

The session state record provides a complete history and sequence of state information generated in the multi-step process. The design goal is that all information about a session is available at all times in the SSR. The approach is that a Protocol Adaptor decodes an incoming event and generates an XML representation of the event with all parameters encoded in XML form. The goal is that all information available about a transaction is provided in the SSR and thus is available to each Workflow Procedure. An example incoming request produced by a Protocol Adaptor is shown below. The form of the message output by the Protocol Adaptor is an XML stanza. In the examples below, XML tags are omitted for readability, and indent supplies structure. The parent element of a stanza indicates the event name or state.

This section provides an example of the first few steps of a transaction to illustrate the concept. The first phase of a session setup is usually a session request. An example session setup request is shown below. The Protocol Adaptor receives the message in an external protocol, decodes all information in the message, and encodes that information as an XML stanza. The Protocol Adaptor dispatches the message to a virtual session management service.

```
RecordingSessionRequest
    Sid=RM_13807
    recording
        id=13807
        deviceId=12345
        accountId=23456
        start=2012-04-12T15:13:44Z
        end=2012-04-12T15:18:44Z
        stationId=1
        title=MTB
```

The session identification parameter “Sid” provides a correlation key for events associated with the session. Upon receipt of the message, the SSM tests for the existence of a corresponding SSR. If an SSR does not exist, the SSM creates one. The resultant SSR is shown below. The Protocol Adaptor inserts an identifier used as the session correlation key. Clients generate a session key that is unique for the service. The Protocol Adaptor in some cases adds a header to guarantee that the identifier is unique in the network.

```
SessionStateRecord
    Sid=RM_13807
    CreateTime=2012/04/12 11:13:44.939
    RecordingSessionRequest
        Sid=RM_13807
        recording
            id=13807
            deviceId=12345
            accountId=23456
            start=2012-04-12T15:13:44Z
            end=2012-04-12T15:18:44Z
            stationId=1
            title=MTB
        Time=2012/04/12 11:13:44.940
```

After creation of the SSR, or upon receiving a stanza corresponding to an existing session, the SSM looks up the workflow procedure associated with the RecordingSessionRequest event, and invokes the corresponding procedure with the SSR as one of the parameters passed to the procedure. The SSM appends the incoming stanza to the SSR with the convention that the last stanza of the SSR represents the current state. The workflow procedure processes the phase of the transaction which typically results in a call-out to an internal or external system. In the example below, the workflow procedure requests resources for the session by generating an XML stanza corresponding to the resource request.

```
SessionStateRecord
    Sid=RM_13807
    CreateTime=2012/04/12 11:13:44.939
    RecordingSessionRequest
        Sid=RM_13807
        recording
            id=13807
            deviceId=12345
            accountId=23456
            start=2012-04-12T15:13:44Z
            end=2012-04-12T15:18:44Z
            stationId=1
```

```
            title=MTB
        Time=2012/04/12 11:13:44.940
    ResourceRequest
        Sid=RM_13807
        RequestType=VideoRecorder
        ResponseEvent=RecordingSession
        SourceId=1
        RecordType=Unique
        StartTime=2012-04-12T15:13:44Z
        EndTime=2012-04-12T15:18:44Z
        BillingAccountId=202
        Resource
            SessionId=54251cf2634d949a6ebe908aec02a4c6
            ContentType=Mpeg2Cbr
            Profile
                Name=Profile1
                AvgBitRate=3000000
                MaxBitRate
                Input
                    SourceUrl=udp://232.36.254.221:39221/
                    SourceIp=10.252.251.74
                Input
                    SourceUrl=udp://232.36.254.221:39221/
                    SourceIp=10.252.251.75
```

Allocation of resources is accomplished using a ResourceRequest stanza. The previous example shows how a resource is requested for a session. The resource request is merely a step in the workflow defined for the corresponding service.

Event Collection

This section describes the event collection facilities of the SSM. The SSM is intended to serve as a session and resource management facility, but also provides powerful functions for event collection.

Overview

This section is an overview of the event collection facilities. Specific APIs are provided in later sections. As mentioned previosuly, the SSM also serves as an event collection mechanism without requiring additional software. The only requirement is that session creation and the update of event names are defined in the SSM event table. The SSM will create a session state record on the initiation of a session. Mid-session updates, including the end-of-session indication will be appended to the record. The result is a complete record of the history of the session. The data in the record includes all information provided by the device providing session creation and update events.

The sessionID is the primary key for a session state record. The CreateDate operation forms a secondary key. Queries of the session state record datastore can include a combination of keys consisting of the primary and multiple secondary keys. Any session event can specify secondary keys about the session. The SSM tests for the existence of secondary key definitions in the session create event. In addition, mid-session update events can also specify additional secondary keys.

An SSM API call can be handled by any of the SSMs in a given cluster. To use HTTP, the XML can be sent via HTTP POST to http://<SSM IP>/bus/xml. To use XMPP, send the XML stanza as an IQ payload to the JID of the SSM virtual service. For XMPP, the Conductor Bus supplies redundancy and load balancing among the Session State Managers; if using HTTP, the calling application should implement redundancy and load balancing logic when calling the SSM.

Events

Request Direction: External elements to Session State Manager

Method: HTTP Post or XMPP

Description:

An external element, such as a workflow, sends events to Session State Manager to be persisted in the session state record associated with the session id (Sid) supplied in the event. The schema of these events is determined by the sender, but must conform to the basic requirements of the Session State Manager.

Location:

(XMPP) Session State Manager Virtual Service JID

(HTTP) **`http://<session state manager IP >/bus/xml`**

Request:

Element: sender defined

Children: **Sid (1)**

SessionKeys (0..1)

sender defined (0..N)

The stanza name and all attributes are sender-defined.

Data Type	Description
xs:string	Sender-defined event **Example: `SessionCreate`**

The sender must supply a session ID in the "Sid" element. This specifies the session ID, which is used as the primary key for the session state record.

Element: **Sid (1)**

Children: **none**

Data Type	Description
xs:string	Session ID **Example: `54251cf2634d949a6ebe908aec02a4c6`**

The sender may supply a SessionKeys element which contains any number of key/value pairs which will be used as secondary keys. If a specified secondary key does not exist, it will be added. If a specified secondary key already exists, it will be updated to the supplied value.

Element: **SessionKeys (0..1)**

Children: **sender defined (0..N)**

Data Type	Description
xs:string	Session ID **Example:** **<SessionKeys> <accountId>131232123</accountId> <app>VOD</app> </SessionKeys>**

The sender may supply any other valid XML elements/children elements.

Request example:

```
POST /bus/xml HTTP/1.1
Content-Type: application/xml
Content-Length: 234

<SessionCreate>
<Sid>1123123-1231235-1231231241</Sid>
<SessionKeys>
<AccountId>A4R2341</AccountId>
<Application>VOD</Application>
<Status>Pending</Status>
</SessionKeys>
...
Sender supplied XML
...
</SessionCreate>
```

Response:

HTTP Status Code	200	Ok
	404	URL not found
	500	Internal Server Error

A *200* response indicates that the event was successfully received. A *404* error indicates a problem processing the URL specified in the request. A *500* error indicates an unspecified processing error.

The payload of the response will be an XML stanza. The root element of the XML stanza will be the stanza name of the event with "Response" appended.

Element: **Status (1)**

Children: **none**

The SSM will indicate the status of the event using the Status element. The value of the Status element is the status of the event processing / storage. Valid values are "Success" or "Failure".

Data Type	Description
xs:string	Status of the event. Valid values are "Success" or "Failure".

Element: **sender defined plus Response**
Children: **Status (1)**

Response Example:

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 234

<SessionCreateResponse>
<Status>Success</Status>
</SessionCreateResponse>
```

Session Record Retrieval

SsrGet

Request Direction: External elements to Session State Manager

Method: HTTP Post or XMPP

Description:

An external element, such as an analytics engine, may get session state records with the SsrGet call.

Location:

(XMPP) Session State Manager Virtual Service JID

(HTTP) **`http://<session state manager IP >/bus/xml`**

Request:

Element: **SsrGet**

Children: **Ssr (1)**

The sender must supply a session ID in the Ssr element. This is the session ID that was supplied as Sid in the Event messages.

Element: **Ssr (1)**

Children: **none**

Data Type	Description
xs:string	Session ID **Example:** **54251cf2634d949a6ebe908aec02a4c6**

Request:

```
POST /bus/xml HTTP/1.1
Content-Type: application/xml
Content-Length: 234
<SsrGet>
<Ssr>RM_3000</Ssr>
</SsrGet>
```

Response:

HTTP Status Code	200	Ok
	404	URL not found
	500	Internal Server Error

A 200 response indicates that the request was successfully processed and a valid response will be generated. A 404 error indicates a problem processing the URL specified in the request. A 500 error indicates an unspecified processing error.

Element: **SessionStateRecord**
Children: **Sid(1)**
CreateTime (1)
Sender defined events (1..N)

The Sid element indicates the session ID of the Session State Record.

Element: **Sid (1)**
Children: **none**

Data Type	Description
xs:string	Session ID **Example:** **54251cf2634d949a6ebe908aec02a4c6**

The CreateDate element indicates the timestamp during which the Session State Record was created.

Element: **CreateDate (1)**

Children: **none**

Data Type	Description
xs:string	Timestamp of the creation of the Session State Record **Example:** **2012/04/13 12:17:14.752**

Response:

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 234

<SessionStateRecord>
    <Sid>RM_3000</Sid>
    <CreateTime>2012/04/13 12:17:14.752</CreateTime>
    <RecordingSessionRequest>
        <Sid>RM_3000</Sid>
        <recording>
            <id>3000</id>
            <deviceId>12345</deviceId>
            <accountId>23456</accountId>
       </recording>
        <Time>2012/04/13 12:17:14.752</Time>
    </RecordingSessionRequest>
    <ResourceRequest>
        <Sid>RM_3000</Sid>
        <RequestType>VideoRecorder</RequestType>
        <ResponseEvent>RecordingSession</ResponseEvent>
        <EndTime>2012-04-13T16:17:24Z</EndTime>
    </ResourceRequest>
    ...
</SessionStateRecord>
```

Session Record Querying

SsrQuery

Request Direction: External elements to Session State Manager

Method: HTTP Post or XMPP

Description:

An external element, such as an analytics engine, may get a list of session state records that match a query.

Location:

(XMPP) Session State Manager Virtual Service JID

(HTTP) **`http://<session state manager IP >/bus/xml`**

Request:

Element: **SsrQuery**

Children: **User defined key value pairs (1..N)**

The sender must supply one or more key value pairs as children of the SsrQuery element. All queries will use an "AND" for all key/value pairs, such that all records will be returned where key1 == value1 AND key2 == value2 AND ... AND keyN == valueN.

Element: **User defined key value pairs (1..N)**

Children: **none**

Data Type	Description
xs:string	Key value pairs to query **Example:** **`<accountId>131232123</accountId>`** **`<app>VOD</app>`**

Request:

```
POST /bus/xml HTTP/1.1
Content-Type: application/xml
Content-Length: 234
<SsrQuery>
<AccountId>A4R2341</AccountId>
<Application>VOD</Application>
<Status>Pending</Status>
</SsrQuery>
```

Response:

HTTP Status Code	200	Ok
	404	URL not found
	500	Internal Server Error

A 200 response indicates that the request was successfully processed and a valid response will be generated. A 404 error indicates a problem processing the URL specified in the request. A 500 error indicates an unspecified processing error.

Element: **SsrQueryResponse**

Children: **Ssr(0..N)**

Matching SSRs, if any, will be returned as the values of the Ssr elements.

Element: **Ssr (0..N)**

Children: **none**

Data Type	Description
xs:string	Session ID **Example:** **54251cf2634d949a6ebe908aec02a4c6**

Response:

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 234
<SsrQueryResponse>
    <Ssr>RM_3000</Ssr>
    <Ssr>RM_3002</Ssr>
    <Ssr>RM_3010</Ssr>
</SsrQueryResponse>
```

Session Record Deletion

SsrDelete

Request Direction: External elements to Session State Manager

Method: HTTP Post or XMPP

Description:

An external element, such as an analytics engine, may delete one or more Session State Records.

Location:

(XMPP) Session State Manager Virtual Service JID

(HTTP) **`http://<session state manager IP >/bus/xml`**

Request:

Element: **SsrDelete**

Children: **Ssr(1..N)**

SSRs to delete:

Element: **Ssr(1..N)**

Children: **none**

Data Type	Description
xs:string	Session ID **Example:** **`54251cf2634d949a6ebe908aec02a4c6`**

Request Example:

```
POST /bus/xml HTTP/1.1
Content-Type: application/xml
Content-Length: 234

<SsrDelete>
    <Ssr>RM_3000</Ssr>
    <Ssr>RM_3002</Ssr>
    <Ssr>RM_3010</Ssr>
</SsrDelete>
```

Response:

HTTP Status Code	200	Ok
	404	URL not found
	500	Internal Server Error

A 200 response indicates that the request was successfully processed and a valid response will be generated. A 404 error indicates a problem processing the URL specified in the request. A 500 error indicates an unspecified processing error.

Element: **SsrDeleteResponse**
Children: **Ssr(1..N)**

All SSRs specified in the SsrDelete message must be supplied in the SsrDeleteResponse message with a status attribute.

Element: **Ssr (1..N)**
Children: **none**

Attributes	Use	Data Type	Description
status	Required	xs:string	Status of the delete attempt. Valid values are: ■ Success - The SSR was found and deleted ■ NotFound - The SSR does not exist ■ Failure - An unspecified error occurred while retrieving the SSR

Data Type	Description
xs:string	Session ID **Example:** **54251cf2634d949a6ebe908aec02a4c6**

Response Example:

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 234

<SsrDeleteResponse>
    <Ssr status="Success">RM_3000</Ssr>
    <Ssr status="NotFound">RM_3002</Ssr>
    <Ssr status="Failure">RM_3010</Ssr>
</SsrDeleteResponse>
```

Session Management Workflow Procedures

The SSM invokes configured workflow procedures to process a phase of a transaction. The procedures are supplied as plug-in procedures dynamically located at run-time. The SSM is configured with a list of libraries in search order. Upon configuring a workflow event and procedure, the SSM searches the configured libraries to locate the procedure. An alarm status is asserted if the workflow procedure cannot be located.

A Protocol Adaptor or Workflow Procedure invokes the SSM API by merely sending a message to the SSM service. The SSM service looks up the Workflow Procedure for the event identified by the XML stanza parent element and invoking the associated procedure.

c/c++ Calling Convention

The prototype for a c/c++ workflow procedure is as follows:

```
uint workflowProc(char *sid,
      UDS_ELEMENT_T *ssrPtr,
      UDS_ELEMENT_T *workBuffer,
      char *nextJid);
```

- The first argument provides a pointer to the session correlation key, referred to as the Sid. Each event in a workflow sequence supplies the Sid as a correlation tag. Upon entry to the workflow procedure, this parameter is provided.
- The second argument is a pointer to a decoded session state record. The incoming XML session state record is decoded into the internal Universal Data Structure (UDS) format. UDS is a lossless internal data representation that provides rapid access to elements and attributes in an XML environment. By convention, the last stanza in the SSR representation corresponds to the current state. The SSM appended the current state to the SSR prior to the invocation of the workflow procedure.
- The third argument provides a pointer to a workspace. Session management transactions are typically a sequence of steps/phases where each step requires a call to an external system (validation, authorization, resolution, resource allocation, etc.). The supplied workspace can be used to encode an XML stanza for the next phase. The return status, defined later, can be used to indicate that the next phase should be invoked using the supplied workspace buffer.
- The fourth argument is used to provide the JID corresponding to the next phase of the transaction. The JID can refer to a specific instance (e.g. Protocol Adaptor), a virtual service, or NULL to indicate that the next phase can merely operate on the local node.

The returned status indicates processing for the next phase of the transaction. Two status types are supported in the current release. One status type indicates that the next stanza is defined in the workspace buffer with the result that the workspace data is appended to the SSR and the next phase of the transaction is invoked. The other status indicates that no further action is required by the SSM.

Python Calling Convention

The parameter list is similar for a Python procedure. The difference is that parameters are encoded in XML instead of UDS. Another important difference is that the workspace buffer, which is encoded as an XML stanza for the next phase, is returned as an output. This is needed, as Python does not support pass-by-reference semantics. The returned status is included as a first tag element of the workspace buffer and the nextJid is included as a second tag element in the returned output.

- The first parameter of the Python script provides the session correlation key. The second argument provides the XML representation of the SSR. The third argument provides the suggested jid corresponding to the next phase of the transaction.
- The second argument provides a pointer to the XML representation of the SSR.
- The third argument provides a pointer to a local workspace buffer. This buffer can be used to encode an XML stanza for the next phase of the transaction.

2

ERM WebService Interface

Introduction

The Edge Resource Manager (ERM) is responsible for the management of the network resources needed for setting up video sessions. In the traditional HFC plant, the managed resources for applications, like Video-on-Demand (VOD) and Switched Digital Video (SDV), were RF QAM and Encryptor resources. For Video over DOCSIS (VDOC) applications, an ERM may manage CMTS devices. For a DVR application, an ERM may manage recorder devices.

This chapter defines a unified Web Service interface for the ERM, which includes allocation and management of edge resources in all possible environments.

In This Chapter

Interface Overview

This interface attempts to be the unified interface for all on-demand applications for resource allocation. The Session Manager to Edge Resource Manager interface defined for NGOD, based on RTSP, is known as the S6 interface. In the ISA architecture, a Session Setup Protocol (SSP) extension known as "Session Setup Protocol – Server Initiated Session (SSP-SIS)", an interface equivalent of S6, is defined in terms of DSMCC resource descriptors. As both NGOD and ISA environments manage similar kind of edge resources (QAMs and Encryptors), this web service interface can be used in both of these environments, and can supersede both of these protocols.

Furthermore, this interface can be used for enabling deterministic Quality of Service (QoS) for on-demand applications such as video-on-demand (VoD), time-shifted TV, video conferencing, and other IP-based streaming applications in environments such as Video-over-DOCSIS (VDOC).

Container Parameter Definitions

Parameter Requirements

The component interfaces defined in this section need to support a number of architectures, including:

- NGOD VOD
- ISA VOD
- SDV
- QoS Admission Control

Method parameters may be required for one or more architectures, or they may be optional. The following table shows the letters assigned to define the requirements.

Letter	Requirement
R	Required for all architectures
N	Required for NGOD VOD
I	Required for ISA VOD
S	Required for SDV
Q	Required for QoS Admission Control
O	Optional for all architectures

Example: If the "R/O" (Required/Optional) field shows the letters "NI", it means that that parameter is required for both NGOD VOD and ISA VOD.

Conditional Access

The Conditional Access container stores all of the encryption parameters. This is used in the Session Setup Request (SSR) to specify the Conditional Access parameters for the requested session. For clear sessions, this container is not included.

Child Element	Type	R/O	Description
CaSystemType	String	R/O	Identifies the conditional access system Permissible values: ■ NONE ■ NDS ■ POWER_KEY ■ MEDIA_CIPHER ■ DVB_CA ■ Unknown-decimal value
DigitalCopyProtection	String	O	This is part of the Copy Control Information (CCI) as defined by SCTE 41 (2011) [8]. Permissible Values: ■ No-Restriction ■ No Further Copy ■ One-Generation-Copy-Permitted ■ Copying-Prohibited
AnalogProtectionSystem	String	O	This is part of the Copy Control Information (CCI) as defined by SCTE 41 (2011) [8]. Permissible Values: ■ Copy-Protection-Encoding-Off ■ AGC-On-Split-Burst-Off ■ AGC-On-2-Line-Split-Burst-On ■ AGC-On-4-Line-Split-Burst-On
AccessCriteria	Hexadecimal	O	Specifies the Access Criteria to be delivered to the encrypting device. The format is defined by the user. This is Useuul for encryption methods like Simulcrypt.

Examples:

- Here is an example where AccessCriteria is specified for setting up an NDS encryption:

```
<ConditionalAccess>
   <CaSystemType>NDS</CaSystemType>
<AccessCriteria>00030002002A0000F00481008300830100 1C000400010E
00000D00012D0000000000000000000000000FFFFFFFFFFF000001001F0600
00000C0018C70431303239F10106F2050100031828F2020201F2020307FFFE
002F010000000C0028A0000400000405A700020101A700020201A7000503FF
FFFFFFA70003040008A400020001AE000107</AccessCriteria>
</ConditionalAccess>
```

- Here is an example of a Conditional Access container for setting up a PowerKEY session:

```
<ConditionalAccess>
   <CaSystemType>POWER_KEY</CaSystemType>
   <DigitalCopyProtection>Copying-
Prohibited</DigitalCopyProtection>
   < AnalogProtectionSystem>AGC-On-Split-Burst-
Off</AnalogProtectionSystem>
</ConditionalAccess>
```

Client Conditional Access

The Client Conditional Access container stores all of the encryption parameters needed for an STB to decrypt a real-time encrypted stream. This is used in the response sent back to the Session Setup Request.

Child Element	Type	R/O	Description
CaSystemType	String	R	Identifies the conditional access system. Permissible values: ■ NONE ■ NDS ■ POWER_KEY ■ MEDIA_CIPHER ■ DVB_CA
DigitalCopyProtection	String	O	Permissible Values: ■ No-Restriction ■ No Further Copy ■ One-Generation-Copy-Permitted ■ Copying-Prohibited

Child Element	Type	R/O	Description
AnalogProtectionSystem	String	O	Permissible Values: ■ Copy-Protection-Encoding-Off ■ AGC-On-Split-Burst-Off ■ AGC-On-2-Line-Split-Burst-On ■ AGC-On-4-Line-Split-Burst-On
ClientCaData	Hexadecimal	O	Specifies the conditional access (CA) data for the client to decrypt the incoming encrypted stream.

Example:

```
<ClientConditionalAccess>
      <CaSystemType>Power_Key</CaSystemType>
      <DigitalCopyProtection>Copying-
Prohibited</DigitalCopyProtection>
      < AnalogProtectionSystem>AGC-On-Split-Burst-
Off</AnalogProtectionSystem>
<ClientCaData>
000000000001861d1e220000011f111d060e000c9802960021be77c7990100000
001000208002886923cb0df4a18a082d7e16af4c6f5124338e047b92e16360c9b
aa5414e4f63dfc6f1bd49d38129f6782e2ecc24d248ce326b7cd5bd97275691d1
cac41880ef72f1b9f44847a7a014b13df2d39c49ed31c9a6973caa8a85d677f56
78f58b98790202bc2278a636ed04e8f1d68806e16cec23fa987749133c179ba90
26fcebf875c8dafc6b2d52efccda941d85f2f094e498b193e071459f7cacd618a
747439a4e8ac46e7cffdec9cfc9ca42c5ef80f662e0f31b09ee761f823a6883f4
16e890fc4a3587e0066cc8996d7dbe8e545f6379103085ae7b6a7ce713726c55c
15c92d16e2c0d1ee4fdd755725e9df56af5b49fe91b1fd31bf6a6ef5b3bbf34ce
46c229dd480aeb452
</ClientCaData>
</ClientConditionalAccess>
```

QAM Transport

The QamTransport container stores details about a single QAM transport element. This is used during the resource negotiation, reservation, and reporting process.

Name	Type	R/O	Description
Delivery	String (take off)	R	unicast|multicast
Frequency	Integer	R	Tune frequency in kHz
MpegProgram	Integer	R	MPEG program number of the content. Range is 1-65535.

Name	Type	R/O	Description
ModulationMode	String	R	QAM64|QAM256|QAM1024
QamName	String	N	Name of the QAM
QamGroup	String	O	QAM Group name
TSID	Integer	R	Transport stream ID. Range is 1-65535.
Annex	String	O	A|B|C
ChannelWidth	Integer	O	QAM channel width. 6|7|8 MHz
Interleaver	String	O	I128J1 | 128J2 | I64J2 | I128J3 | I32J4 | I128J4 | I16J8 | I128J5 | I8J16 | I128J6 | I128J7 | I128J8

Example:

```
<QamTransport>
      <Delivery>unicast</Delivery>
      <Frequency>611000</Frequency>
      <MpegProgram>18</MpegProgram>
      <ModulationMode>QAM256</ModulationMode>
</QamTransport>
```

UDP Transport

The UDPTransport container stores details about a single UDP transport element. This is used during the resource negotiation, reservation, and reporting process.

Name	Type	R/O	Description
DestIP	IP Address	R	IPv4 or IPv6 address of the destination of the stream
DestPort	Integer	R	Destination UDP port number
SourceIP	IP Address	O	IPv4 or IPv6 address of the source of the stream
SourcePort	Integer	O	Source port number
EdgeInputGroup	String	O	Group name associated with the input port on the edge device

Example:

```
<UdpTransport>
      <DestIP>10.1.2.193</DestIP>
      <EdgeInputGroup>UR3.21</EdgeInputGroup>
      <DestPort>384</DestPort>
</UdpTransport>
```

Multicast Transport

The Multicast Transport container stores details about a multicast specification. This is used during the resource negotiation, reservation, and reporting process.

Name	Type	R/O	Description
SourceAddress	IP Address	R/O	IPv4 or IPv6 address of the source of the stream
GroupAddress	IP Address	R	IPv4 or IPv6 address of the group multicast of the stream
ReceiveAddress	IP Address	O	IPv4 or IPv6 address of the destination of the stream
GroupPort	Integer	O	Group Destination port number
Priority	Integer	O	Priority of the input source and destination combination.

Example:

```
<MulticastTransport>
<SourceAddress>10.30.1.24</SourceAddress>
<GroupAddress>224.2.1.103<GroupAddress>
<ReceiveAddress>10.28.4.13</ReceiveAddress>
<GroupPort>1901</GroupPort>
</MulticastTransport>
```

Response

The Response container stores an integer value and an optional text string. This is used in the response sent back to any of the request messages.

Name	Type	R/O	Description
ResponseCode	Integer	R	Value indicates success or failure of the operation. A zero is success and any other value is an error code.
ResponseText	String	O	Associated response text.

Reason

The Reason container stores an integer value and an optional text string. This is used in the Session Teardown and SessionNotify messages.

Name	Type	R/O	Description
ReasonCode	Integer	R	Value indicates the reason code of the operation.
ReasonText	String	O	Associated text for the reason code.

Session List

The SessionList container stores a list of session IDs. This is typically used for synchronizing session lists between two peers or for sending Session Ping messages for keep-alive.

Child Element	Type	R/O	Description
SessionId	String	R	Unique ID of the session.

Example:

```
<SessionList>
      <SessionId>be074250-cc5a-11d9-8cd5-0800200c9a66</SessionId>
      <SessionId>be074250-cd6a-11d9-8cd5-53239928cd8f</SessionId>
</SessionList>
```

Qam List

The QamList container stores a list of QAM Names. This is used for specifying a list of QAM devices visible to the client, typically in a setup request. It is rarely used for ISA applications, as the Session Manager typically has no knowledge of the active QAM list. It is the job of ERM to manage edge devices and to make a selection based on the Service Group information provided by the Session Manager.

Child Element	Type	R/O	Description
QamName	String	N	Name of a QAM (or TSID)
TSID	Integer	I	TSID

Examples:

- Here is an example of the QAM List provided for NGOD applications:

```
<QamList>
        <QamName>Philly.132</QamName>
        <QamName>Philly.134</QamName>
        <QamName>Philly.136</QamName>
</QamList>
```

- Here is an example of the QAM List provided for ISA applications:

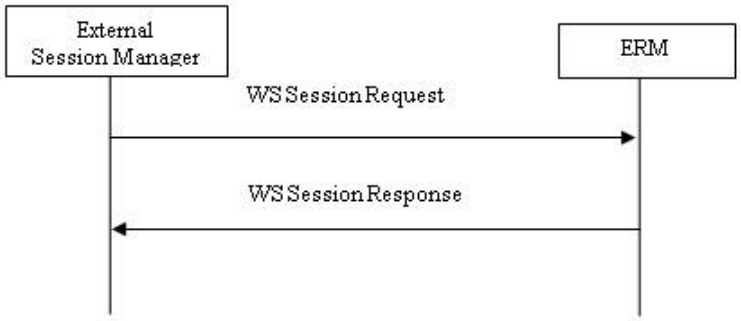
```
<QamList>
        <TSID>132</QamName>
</QamList>
```

Session Manager - Edge Resource Manager Interface

Session Setup Request

The Session Manager sends a SessionSetup request to the Edge Resource Manager to allocate resources on the Edge QAM Devices. The Session Manager must supply a list of candidate QAMs that can reach the client devices (such as set-top boxes), or it must specify a service group.

- The service group identified by a list of QAMs or a Service Group ID
- The requested bandwidth
- The multicast group and source addresses (switched multicast only)
- Preferred encryption mechanisms and CA types



Request Parameters

Name	Type	R/O	Description
SessionId	String	NI	Session Manager assigned session ID. **Example:** NGOD On Demand Session ID. 36 bytes of string or 10 bytes of DSMCC.
TransportType	String	O	Permissible Values: ■ MPEG-2 ■ DOCSIS If the transport type is not included, MPEG-2 transport is assumed.
ClientId	String	O	Mac Address or unique identifier of the Client or CPE device. 6 binary bytes = 12 ASCII bytes.

Name	Type	R/O	Description
Bandwidth	Integer	NI	Maximum bit rate of the content. bits per second.
MulticastTransport	Container	O	Multicast transport of the input. Useful for multicast sessions (for example for Switched Digital Video).
ConditionalAccess	Container	O	Encryption information.
ServiceGroup	String	NI	Service group identifier. Required for ISA, and conditionally required for NGOD for SA settops. This string could be an integer representing the Service Group ID.
StreamingZone	String	N	Identifies the Streaming Zone (NGOD) or Headend associated with this client request.
QamList	Container	NI	List of QAM Names visible to the Client or CPE device from Motorola set-tops. In case of ISA, this could provide the list of specific TSID(s) on which the session is requested.
Policy	String	O	Specifies the policy and priority information. For this release, the policy is mostly determined by the name of the application.
ApplicationName	String	NI	Specifies the name of the application. Useful for policy determination.
SessionGroup	String	N	The session group associated with the SM making the setup request.
SMNotifyURL	String	O	The complete URL to the event-notify service of the SM associated with this session. (Conductor env).
SessionAdjust	String	O	“Yes” or “No” to allow adjusting the parameters for an existing session.
SwitchingQamAllowed	String	O	“Yes” or “No” to allow switching the parameters of the QAM. The default value is “No”. This field is applicable only to adjust an existing session (SessionAdjust is set to “Yes”).

Name	Type	R/O	Description
ServiceType	String	O	Permissible Values: ■ Interactive ■ Static ■ Download ■ Stream Download and Stream are useful for Qos applications such as applications in Video over Docsis (VDOC) domain only.
ServiceId	String	Q/O	String ID of the asset (GUID of the VOD asset).
Delivery	String	R	unicast|multicast
AllowPid Remapping	Boolean	O	Indicates whether the edge device should remap PIDs. The default is to allow for the remap of PIDs.
ClientPort	Integer	Q	Port number of the client making request.
ServerIP	String	Q	IP address of the NIC on the VOD Server handling the request.
ServerPort	String	Q	Port number of the VOD Server handling the request.
SessionExpirationPeriod	Integer	O	Expiration period in seconds after which the resources for this session are torn-down.

Note: An existing session can be modified using the invocation of the same web-service call by setting the SessionAdjust parameter to "Yes". When an existing session is being modified, another parameter QamSwitchingAllowed is used to communicate, whether switching the QAM transport is allowed or not.

Session Setup Response

A Session Setup response must have a response code and, upon success, must contain the following:

- QAM access information - QAM IP address and UDP port
- QAM tuning information - QAM frequency, modulation, and MPEG program number
- Client CA information – Encryption information needed for decryption on the client side.

If the resource allocation fails, the ERM must return a response with an appropriate response error code.

Response Parameters

Name	Type	R/O	Description
Response	Container	R	Value indicates success or failure of the operation. A zero is success and any other value is an error code.
QamTransport	Container	NI	QAM output parameters for the session.
UdpTransport	Container	NI	Input Ethernet port to use for the session. This is only applicable for VOD.
ClientCaData	Hexadecimal	O	Encryption information needed for client to decrypt.
ERMSessionId	String	N	The ERM generated session identifier.

HTTP Status Codes

- `200` — Success/OK
- `400` — Bad Request
- `403` — Forbidden
- `404` — Session Not Found
- `500` — Internal Server Error
- `503` — Service Unavailable
- `504` — Service Timeout

Location

`http://[ERMServer]/erm/SessionSetup`

Example: Unencrypted Unicast (VOD)

The following is an example of Unencrypted Unicast (VOD) Session setup using a list of QAMs:

- Request (In ISA applications)

```
<SessionSetup>
   <ServiceGroup>1001</ServiceGroup>
   <SessionGroup>VodHE1</SessionGroup>
   <SessionId>be074250-cc5a-11d9-8cd5-0800200c9a66
   </SessionId>
   <ClientId>00CF00112281</ClientId>
   <Delivery>Unicast</Delivery>
   <Bandwidth>3750</Bandwidth>
</SessionSetup>
```

- Request (for NGOD applications)

```
<SessionSetup>
   <SessionGroup>VodHE1</SessionGroup>
   <SessionId>be074250-cc5a-11d9-8cd5-0800200c9a66
   </SessionId>
   <ClientId>00CF00112281</ClientId>
   <Delivery>Unicast</Delivery>
   <Bandwidth>3750</Bandwidth>
   <QamList>
         <QamName>Philly.132</QamName>
         <QamName>Philly.134</QamName>
         <QamName>Philly.136</QamName>
   </QamList>
</SessionSetup>
```

Response:

```
<SessionSetupResponse>
      <Response>
            <ResponseCode>0</ResponseCode>
      </Response>
      <QamTransport>
            <Delivery>unicast</Delivery>
            <QamName>Philly.118</QamName>
            <QamGroup>Philly.qg2</QamGroup>
            <Frequency>611000</Frequency>
            <MpegProgram>18</MpegProgram>
            <ModulationMode>QAM256</ModulationMode>
      </QamTransport>
      <UdpTransport>
            <Delivery>unicast</Delivery>
            <DestIP>10.1.2.193</DestIP>
```

```
            <DestPort>384</DestPort>
            <EdgeInputGroup>UR3.21</EdgeInputGroup>
      </UdpTransport>
      <ERMSessionId>af064160-cc5a-11d9-8cd5-0800200c9baa</ERMSessionId>
</SessionSetupResponse>
```

Example: Encrypted Unicast (VOD) Session Setup

Request

```
<SessionSetup>
      <SessionId>be074250-cc5a-11d9-8cd5-0800200c9a66
      </SessionId>
<ServiceGroup>1001</ServiceGroup>
      <ClientId>00CF00112281</ClientId>
      <Bandwidth>3750</Bandwidth>
      <AllowPid Remapping>1</AllowPidRemapping>
<ConditionalAccess>
      <CaSystemType>POWER_KEY</CaSystemType>
      <DigitalCopyProtection>Copying-
Prohibited</DigitalCopyProtection>
      < AnalogProtectionSystem>AGC-On-Split-Burst-
Off</AnalogProtectionSystem>
</ConditionalAccess>
</SessionSetup>
```

Response

```
<SessionSetupResponse>
      <ResponseCode>0</ResponseCode>
      <QamTransport>
            <Delivery>unicast</Delivery>
            <TSID>118</TSID>
            <Frequency>611000</Frequency>
            <MpegProgram>18</MpegProgram>
            <ModulationMode>QAM256</ModulationMode>
      </QamTransport>
      <UdpTransport>
            <Delivery>unicast</Delivery>
            <DestIP>10.1.2.193</DestIP>
            <DestPort>384</DestPort>
```

```
            <EdgeInputGroup>UR3.21</EdgeInputGroup>
        </UdpTransport>
        <ClientCaData>
00000000001861d1e220000011f111d060e000c9802960021be77c799010000
0001000208002886923cb0df4a18a082d7e16af4c6f5124338e047b92e16360c
9baa5414e4f63dfc6f1bd49d38129f6782e2ecc24d248ce326b7cd5bd9727569
1d1cac41880ef72f1b9f44847a7a014b13df2d39c49ed31c9a6973caa8a85d67
7f5678f58b98790202bc2278a636ed04e8f1d68806e16cec23fa987749133c17
9ba9026fcebf875c8dafc6b2d52efccda941d85f2f094e498b193e071459f7ca
cd618a747439a4e8ac46e7cffdec9cfc9ca42c5ef80f662e0f31b09ee761f823
a6883f416e890fc4a3587e0066cc8996d7dbe8e545f6379103085ae7b6a7ce71
3726c55c15c92d16e2c0d1ee4fdd755725e9df56af5b49fe91b1fd31bf6a6ef5
b3bbf34ce46c229dd480aeb452
    </ClientCaData>
    <ERMSessionId>af064160-cc5a-11d9-8cd5-0800200c9baa</ERMSessionId>
</SessionSetupResponse>
```

Example: Multicast Session Setup (SDV)

Request

```
<SessionSetup>
      <ServerSessionId>be074250-cc5a-11d9-8cd5-
0800200c9a66</ServerSessionId>
      <ClientId>00CF00112281</ClientId>
      <Bandwidth>3750</Bandwidth>
      <ServiceGroup>57</ServiceGroup>
    <MulticastTransport>
            <SourceAddress>10.30.1.24</SourceAddress>
            <GroupAddress>224.2.1.103</GroupAddress>
    </MulticastTransport>
    <MulticastTransport>
            <SourceAddress>10.30.1.25</SourceAddress>
            <GroupAddress>224.2.1.103</GroupAddress>
   </MulticastTransport>
    <MulticastTransport>
            <SourceAddress>10.30.1.26</SourceAddress>
            <GroupAddress>224.2.1.103</GroupAddress>
    </MulticastTransport>
</SessionSetup>
```

Note: For the current implementations, only one GDA.

Response

```
<SessionSetupResponse>
        <ResponseCode>0</ResponseCode>
        <QamTransport>
     <Delivery>unicast</Delivery>
              <TSID>118</TSID>
              <Frequency>611000</Frequency>
              <MpegProgram>18</MpegProgram>
              <ModulationMode>QAM256</ModulationMode>
        </QamTransport>
</SessionSetupResponse>
```

Example: QoS Request

Request

```
<SessionSetup>
       <ServerSessionId>12345678-1111-2222-3333-000000000001
       </ServerSessionId>
       <ServiceId>01020304-0001-0002-0003-010101010101</ServiceId>
       <ClientId>00CF00112281</ClientId>
       <Delivery>Unicast</Delivery>
       <Bandwidth>100</Bandwidth>
       <ClientIP>192.168.0.100</ClientIP>
       <ClientPort>5000</ClientPort>
       <ServerIP>192.168.0.1</ServerIP>
       <ServerPort>6000</ServerPort>
</SessionSetup>
```

Response

```
<SessionSetupResponse>
       <ResponseCode>0</ResponseCode>
</SessionSetupResponse>
```

Session Teardown

Only the Session Manager initiates a session teardown. The Session Manager sends a SessionTeardown message to the ERM to terminate a session. If the session is known to the ERM, the ERM will release the resources assigned to that session.

Request Parameters

Name	Type	R/O	Description
SessionId	String	NI	Session Manager assigned session ID. This value must match the value sent in the SessionSetup message.
Reason	Container	R	Reason for terminating the session.

Response Parameters

Name	Type	R/O	Description
Response	Container	NI	Value indicates success or failure of the operation. A zero is success and any other value is an error code.

HTTP Status Codes

- `200` — Success/OK
- `400` — Bad Request
- `403` — Forbidden
- `404` — Session Not Found
- `500` — Internal Server Error
- `503` — Service Unavailable
- `504` — Service Timeout

Location

`http://[ERMServer]/erm/SessionTeardown`

Request

```
<SessionTeardown>
      <ServerSessionId>be074250-cc5a-11d9-8cd5-
0800200c9a66</ServerSessionId>
      <Reason>
            <ReasonCode>550</ReasonCode>
      </Reason>
</SessionTeardown>
```

Response

```
<SessionTeardownResponse>
        <Response>
                <ResponseCode>0</ResponseCode>
        </Response>
</SessionTeardownResponse>
```

Session List

When a Session Manager and Edge Resource Manager initially start, the devices must synchronize session lists with its peer. Either the SM or ERM may send a SessionList request to obtain a list of session IDs currently managed by its peer.

The peer must respond with a list of active session IDs associated with the specified SessionGroup.

Request Parameters

Name	Type	R/O	Description
SessionGroup	String	R	Group identifier associated with the Edge Resource Manager. For example, this may be the unique device name or IP address of the ERM. This parameter is not needed if there is a single logical SM and ERM.
ApplicationName	String	O	Name of the application. Currently the name of the application is sometimes encoded into the SessionGroup.
StreamingZone	String	N	Identifies the Streaming Zone (NGOD) or Headend associated with this client request.

Response Parameters

Name	Type	R/O	Description
SessionList	Container	R	Stores the list of session IDs for all active sessions in the specified SessionGroup.

Location

```
http://[ERMServer]/erm/SessionList
```

Request

```
<SessionListQuery>
      <SessionGroup>10.3.12.192</SessionGroup>
</SessionListQuery>
```

Response

```
<SessionListResponse>
      <SessionList>
            <SessionId>be074250-cc5a-11d9-8cd5-0800200c9a66</SessionId>
            <SessionId>be074250-cd6a-11d9-8cd5-53239928cd8f</SessionId>
      </SessionList>
</SessionListResponse>
```

Session Ping

A Session Manager may issue a Session Ping to notify the ERM that one or more sessions are still active. This is a session keep-alive mechanism. The SM typically performs this task if a group of sessions has been in existence for an extended period of time. The issuer of the request must include a session header with the session IDs of the sessions that are being kept alive. The ERM will reset its session activity timer for each session listed.

Note: The Conductor VBO Session Manager will not issue a Session Ping. Instead, the ERM is expected to send an EventNotify command if the session activity timer in the ERM expires.

Request Parameters

Name	Type	R/O	Description
SessionList	Container	NI	List of session IDs for which a Session Ping is issued.

Response Parameters

Name	Type	R/O	Description
Response	Container	NI	Value indicates success or failure of the operation. A zero is success and any other value is an error code.

Location

```
http://[ERMServer]/erm/SessionPing
```

Request

```
<SessionPing>
      <SessionList>
<SessionId>be074250-cc5a-11d9-8cd5-0800200c9a66</SessionId>
<SessionId>be074250-cd6a-11d9-8cd5-53239928cd8f</SessionId>
      </SessionList>
</SessionPing>
```

Response

```
<SessionPingResponse>
<Response>
      <ResponseCode>0</ResponseCode>
</Response>
</SessionPingResponse>
```

Event Notify

The ERM sends an EventNotify request to the Session Manager to notify the SM of an important session-related event. This is equivalent to the NGOD Announce message. The SM may act upon the event, for example, tearing the session down, or it may ignore the event.

Request Parameters

Event Name	Event Code	Description
Output Failure	5401	Encryptor or Edge Device output failure. For an Encryptor, the GigE output failed. For an Edge Device, the QAM failed. SM will tear down impacted sessions.
Network Delivery Failure	5403	Encryptor or Edge Device lost the input stream. SM will tear down impacted sessions.
Multicast Join Failure	5404	Encryptor or Edge Device failed to join any multicast source. SM will tear down impacted sessions.
Input Port Failure	5405	Encryptor or Edge Device GigE input port failed. SM will tear down impacted sessions.

Event Name	Event Code	Description
Multicast Source Change	5406	Encryptor or Edge Device failed over to alternate multicast source. New multicast source specified by ERM. SM may log the event, but sessions not impacted.
Stream Delivery Failure	5407	Generic Encryptor or Edge Device error occurred preventing delivery of the stream to the client. SM tears down impacted sessions.
Bit Rate Exceeds Limit	6001	Encryptor or Edge Device reports the bit rate on specified sessions exceeds limit specified in session setup. SM will tear down impacted sessions.
Session in Progress	5700	ERM uses this code to verify that a particular session is still active. The ERM must specify a single session in the session list. If the session exists, the SM returns 204 No Content. Otherwise, the SM returns 404 Not Found. Note that the SM will send a Session Teardown request to the ERM for each unknown session.
Encryption Engine Failure	6000	Generic encryption failure. SM will tear down impacted sessions.
Destination Unreachable	6004	Encryptor cannot reach selected edge device. SM will tear down impacted sessions.
ECMG Session Failure	6006	ECMG failure. SM will tear down impacted sessions.
SessionTornDown	9001	The session(s) is torn-down on ERM. The probable causes for such teardown could be: ■ StaleSession – ERM determined the session to be active for too long. ■ ExternalTeardown – ERM received a teardown session request from an external entity such as an administrator releasing the session from the GUI.

Location:

`http://<device-url>/sm/EventNotify`

Request Parameters

If the event is related to one or more sessions, the list of impacted sessions is supplied in the SessionList parameter. This allows for efficient notification of events that impact many sessions (for example, QAM failure).

The EventNotify element may include a MulticastTransport child element. This element is only specified for SDV. It is used to indicate the multicast that is being delivered to the clients. This is used when an SDV session is established and when a failover to a redundant source occurs.

Note that while the ERM is expected to utilize the "SessionInProgress" event to validate the status on a session, the ERM must be allowed to tear down a session after a configured time period has elapsed on a session to avoid stale sessions on the ERM.

Name	Type	R/O	Description
EventCode	Integer	R	Integer value of the event code.
EventText	String	O	String description of the event.
EventDate	String	R	String in Date format indicating the date and time of the event occurance.
MulticastTransport	Container	O	Multicast transport of the input. Useful for multicast sessions (for example for Switched Digital Video).
SessionList	Container	NI	List of session IDs for which the event notification is applicable.

Request

```
<EventNotify>
      <EventCode>5401</EventCode>
      <EventText>Downstream Failure</EventText>
      <EventDate>2010-10-23T16:04:37Z</EventDate>
      <SessionList>

      <SessionId>be074250cc5a11d98cd50800200c9a66</SessionId>
      </SessionList>
</EventNotify>
```

HTTP Status Code

- 204 — No Content
- 400 — Bad Request
- 403 — Forbidden
- 404 — Not Found
- 500 — Internal Server Error

A status code of 204 (No Content) indicates that the Session Manager successfully received the request and will take action on the request. No XML body is included.

A 404 (Not Found) code is returned if one or more sessions are unknown to the Session Manager. Separately, the SM will send a Session Teardown request to the ERM for each unknown session.

On all failure responses, no XML data is returned. The HTTP status code and status text specify the error.

3

Customer Information

If You Have Questions

If you have technical questions, call Cisco Services for assistance. Follow the menu options to speak with a service engineer.

Access your company's extranet site to view or order additional technical publications. For accessing instructions, contact the representative who handles your account. Check your extranet site often as the information is updated frequently.

Cisco Systems, Inc.
5030 Sugarloaf Parkway, Box 465447
Lawrenceville, GA 30042

678 277-1120
800 722-2009
www.cisco.com

November 2012 Printed in United States of America Part Number OL-26998-01