Cisco VDS-TV API Guide
Release, 4.6

April, 2017

Cisco Systems, Inc.
www.cisco.com

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco website at www.cisco.com/go/offices.
Preface xiii
  Document Revision History xiii
  Audience xiii
  Document Organization xiv
  Document Conventions xiv
  Related Documentation xv
  Obtaining Documentation and Submitting a Service Request xvi

CHAPTER 1

Introduction to Cisco VDS-TV Software APIs 1-1
  Monitoring and RTSP Stream Diagnostic Interfaces 1-3
    Connections 1-3
    HTTP Headers 1-3
      Request Messages 1-3
    Response Messages 1-6
      Monitoring Response 1-6
      RTSP Stream Diagnostics 1-6
    Message Flow for RTSP Stream Diagnostics APIs 1-7
      Request and Response 1-7
      Request Timeout 1-7
      Invalid Sequence Number 1-8
      Connection Lost 1-8
    TV Playout Interface 1-8

CHAPTER 2

Monitoring APIs 2-1
  MaxErrorModifiedTime 2-2
  MaxStreamModifiedTime 2-2
  StreamHistory 2-3
  StreamListHistory 2-5
  TrickModeHistory 2-7
  TrickModeListHistory 2-9
  PlayServerHistory 2-11
  Errors 2-13
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>3</strong></td>
<td>Maintenance API</td>
</tr>
<tr>
<td></td>
<td>Service restart</td>
</tr>
<tr>
<td><strong>4</strong></td>
<td>RTSP APIs</td>
</tr>
<tr>
<td></td>
<td>Stream Diagnostic APIs</td>
</tr>
<tr>
<td></td>
<td>StreamsBySmartcard</td>
</tr>
<tr>
<td></td>
<td>GetStreamDetailsBySessionId</td>
</tr>
<tr>
<td></td>
<td>GetStreamTrickmodesBySessionId</td>
</tr>
<tr>
<td><strong>5</strong></td>
<td>D5 Interface APIs</td>
</tr>
<tr>
<td></td>
<td>SetStreamingServerConfig</td>
</tr>
<tr>
<td></td>
<td>GetStreamingServerStatus</td>
</tr>
<tr>
<td><strong>6</strong></td>
<td>Package APIs</td>
</tr>
<tr>
<td></td>
<td>Package List</td>
</tr>
<tr>
<td></td>
<td>Package Status List</td>
</tr>
<tr>
<td></td>
<td>Package History List</td>
</tr>
<tr>
<td></td>
<td>Package Stats List</td>
</tr>
<tr>
<td></td>
<td>SOAP Exports</td>
</tr>
<tr>
<td></td>
<td>IngestPackage</td>
</tr>
<tr>
<td></td>
<td>DeletePackage</td>
</tr>
<tr>
<td></td>
<td>UpdatePackage</td>
</tr>
<tr>
<td></td>
<td>GetPackageStatus</td>
</tr>
<tr>
<td></td>
<td>GetAllPackages</td>
</tr>
<tr>
<td></td>
<td>GetAvailableDiskSpace</td>
</tr>
<tr>
<td></td>
<td>SOAP Imports</td>
</tr>
<tr>
<td></td>
<td>AIMPackageNotification</td>
</tr>
<tr>
<td></td>
<td>SOAP Faults</td>
</tr>
</tbody>
</table>

- Devices: 2-15
- ServiceGroups: 2-16
- ContentState: 2-17
- Version: 2-18
- ContentRange: 2-19
- CapacityPlanning: 2-21
- Trick-mode Event Reconciliation: 2-23
WSDL Definitions  6-19
  CiscoAIM.wsdl  6-19
  'CiscoAIMNotification.wsdl'  6-24
GET Publishing Queue Management List  6-26
POST Packages in Publishing Queue  6-27

CHAPTER 7
Replication Group APIs  7-1
  Add Replication Group  7-1
  Replication Error Codes  7-2
  Delete Replication Group  7-3
  Assign members to Replication Group  7-4
  Un-assigning members from Replication Group  7-5
  Replication Group Configuration List  7-6
  Unassigned Replication Servers List  7-7

CHAPTER 8
PVR Scheduler Interface APIs  8-1
  Provisioning a Home  8-1
  PVR Server Reason Codes  8-3
  Adding an STB  8-4
  Updating an STB  8-5
  Deprovisioning the Home  8-7
  Removing an STB  8-8
  Get Home Profile Details  8-9
  Get STB Profile Details  8-10
  Scheduling a Recording  8-11
  Deleting a Recording  8-14
  Notifying the Recording Completion  8-17

CHAPTER 9
VBO APIs  9-1
  Get Content List  9-1
  GetSessionPlay History  9-4

CHAPTER 10
TV Playout APIs  10-1
  TV Playout Schedule Exporter  10-2
  TV Playout Schedule Importer  10-4
  TV Playout Contents Currently Playing  10-7
<table>
<thead>
<tr>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Barker Streams Currently Playing</strong></td>
</tr>
<tr>
<td><strong>TV Playout Channels</strong></td>
</tr>
<tr>
<td><strong>TV Playout Stream Report</strong></td>
</tr>
<tr>
<td><strong>Create Barker Streams</strong></td>
</tr>
<tr>
<td><strong>Create Playlist</strong></td>
</tr>
<tr>
<td><strong>Start/Stop Barker Streams</strong></td>
</tr>
<tr>
<td><strong>Get All Barker Streams</strong></td>
</tr>
<tr>
<td><strong>TV Playout Errors</strong></td>
</tr>
</tbody>
</table>

## CHAPTER 11

### Configuration APIs  11-1

**System Level Configuration API**  11-1

- **System DNS**  11-2
  - GET ALL System DNS Configuration  11-2
  - POST System DNS Configuration  11-2
- **System NTP Server**  11-3
  - GET ALL System NTP Server Configuration  11-3
  - POST System NTP Server Configuration  11-3
- **System Host Service**  11-4
  - GET ALL System Host Service Configuration  11-4
  - POST System Host Service Configuration  11-4
- **System Array Name**  11-5
  - GET ALL System Array Name Configuration  11-5
  - POST System Array Name Configuration  11-5
- **System VOD Market List**  11-6
  - GET VOD Market Configuration  11-6
  - POST VOD Market Configuration  11-7
- **System Content Distribution Rules**  11-8
  - GET Content Distribution Rules Configuration  11-8
  - GET Content Distribution Rules list (based on filters)  11-9
  - POST Content Distribution Rules Configuration  11-10
- **System Priority Rule List**  11-10
  - GET Priority Rule list  11-10
  - POST Priority Rule list  11-11
- **System Metadata Normalization Rules**  11-12
  - GET Metadata Normalization Rules list  11-12
  - POST Metadata Normalization Rules Configuration  11-13
- **System QAM Gateway**  11-14
  - GET ALL System QAM Gateway Configuration  11-14
  - POST System QAM Gateway Configuration  11-14
System Headend Setup  11-15
  GET ALL System Headend Setup Configuration  11-15
  POST System Headend Setup Configuration  11-15
System Stream Destination  11-16
  GET ALL System Stream Destination Configuration  11-16
  POST System Stream Destination Configuration  11-16
System Distributed/Shared ISA Setup  11-17
  GET ALL System Distributed/Shared ISA Setup Configuration  11-17
  POST System Distributed/Shared ISA Setup Configuration  11-17
System Ingest Manager  11-18
  GET ALL System Ingest Manager Configuration  11-18
  POST System Ingest Manager Configuration  11-18
System Authentication Manager  11-19
  GET ALL System Authentication Manager Configuration  11-19
  POST System Authentication Manager Configuration  11-20
System Ingest Tuning  11-20
  GET ALL System Ingest Tuning Configuration  11-20
  POST System Ingest Tuning Configuration  11-21
System MPEG Tuning  11-21
  GET ALL System MPEG Tuning Configuration  11-21
  POST System MPEG Tuning Configuration  11-22
System IP Nicknames  11-22
  GET ALL System IP Nicknames Configuration  11-22
  POST System IP Nicknames Configuration  11-23
System Media Importer/Exporter  11-23
  GET ALL System Media Importer/Exporter Configuration  11-23
  POST System Media Importer/Exporter Configuration  11-24
System Input Channels  11-24
  GET ALL System Input Channels Configuration  11-24
  POST System Input Channels Configuration  11-25
System Source Output Port  11-25
  GET ALL System Source Output Port Configuration  11-25
  POST System Source Output Port Configuration  11-26
System Output Channels  11-26
  GET ALL System Output Channels Configuration  11-26
  POST System Output Channels Configuration  11-27
System Callsign Setup  11-27
  GET ALL System Callsign Setup Configuration  11-27
  POST System Callsign Setup Configuration  11-28
System Ingest Driver Server  11-28
GET ALL System Ingest Driver Server Configuration  11-28
POST System Ingest Driver Server Configuration  11-29

System Logging  11-29
GET ALL System Logging Configuration  11-29
POST System Logging Configuration  11-30

System Syslog  11-30
GET ALL System Syslog Configuration  11-30
POST System Syslog Configuration  11-31

System VBO Setup  11-31
GET ALL System VBO Setup Configuration  11-31
POST System VBO Setup Configuration  11-32

System Backup Configurations  11-32
GET ALL System Backup Configuration  11-32
POST System Backup Configuration  11-33

Array Level Configuration API  11-34

Array Level DNS  11-34
GET ALL Array DNS Configuration  11-35
POST Array DNS Configuration  11-35

Array Level NTP Server  11-35
GET ALL Array NTP Configuration  11-36
POST Array NTP Configuration  11-36

Streamer for BMS Connectivity Configuration  11-36
GET ALL Streamer BMS Configuration  11-36
POST Streamer BMS Configuration  11-37

Vault for BMS Connectivity Configuration  11-38
GET ALL Vault BMS Configuration  11-38
POST Vault BMS Configuration  11-38

Stream Groups Setup  11-39
GET ALL Stream Groups Setup Configuration  11-39
POST Stream Groups Setup Configuration  11-39

SSV Groups Setup  11-40
GET ALL SSV Groups Setup Configuration  11-40
POST SSV Groups Setup Configuration  11-40

VHO Setup  11-41
GET ALL VHO Setup Configuration  11-41
POST VHO Setup Configuration  11-41

Vault Groups Setup  11-42
GET ALL Vault Groups Setup Configuration  11-42
POST Vault Groups Setup Configuration  11-42

Ingest Steering  11-43
GET ALL Ingest Steering Configuration  11-43
POST Ingest Steering Configuration  11-43

Cache Groups Setup  11-44
GET ALL Cache Groups Setup Configuration  11-44
POST Cache Groups Setup Configuration  11-44

Mapping Cache Groups to Cache Groups  11-45
GET ALL Mapping Cache Groups to Cache Groups Configuration  11-45
POST Mapping Cache Groups to Cache Groups Configuration  11-45

Mapping Cache Groups to Vault Groups  11-46
GET ALL Mapping Cache Groups to Vault Groups Configuration  11-46
POST Mapping Cache Groups to Vault Groups Configuration  11-46

D5 Interface Settings  11-47
GET ALL D5 Interface Configuration  11-47
POST D5 Interface Configuration  11-48

Locating Cache Groups  11-48
GET ALL Cache Group Locator Configuration  11-48
POST Cache Group Locator Configuration  11-49

Locating CDN Groups  11-49
GET ALL CDN Group Locator Configuration  11-49
POST CDN Group Locator Configuration  11-50

Mapping Stream to Cache  11-50
GET ALL Mapping Stream to Cache Configurations  11-50
POST Mapping Stream to Cache Configuration  11-51

Mapping Stream to CDN  11-51
GET ALL Mapping Stream to CDN Configurations  11-51
POST Mapping Stream to CDN Configuration  11-52

Mapping Vault Groups for Redundancy  11-52
GET ALL Mapping Vault Groups for Redundancy Configuration  11-52
POST Mapping Vault Groups for Redundancy Configuration  11-53

Master Vault Group  11-53
GET ALL Master Vault Group Configuration  11-53
POST Master Vault Group Configuration  11-54

Control and Setup IPs  11-54
GET ALL Control and Setup IPs Configuration  11-54
POST Control and Setup IPs Configuration  11-55

VHO ISA Setup  11-55
GET ALL VHO ISA Setup Configuration  11-55
POST VHO ISA Setup Configuration  11-56

Site Setup  11-56
GET ALL Site Setup Configuration  11-56
POST Site Setup Configuration 11-57
Thin Pipe Mapping 11-57
  GET ALL Thin Pipe Mapping Configuration 11-57
  POST Site Setup Configurations 11-58
Ingest Driver Client 11-58
  GET ALL Ingest Driver Client Configuration 11-58
  POST Ingest Driver Client Configuration 11-58
Array Level Configuration Backup 11-59
  GET ALL Array Level Backup Configuration 11-59
  POST Array Level Backup Configuration 11-60
Server Level Configuration API 11-61
Server Interface Setup 11-62
  GET Server Interface Setup Configuration 11-62
  GET ALL Server Interface Setup Configuration 11-62
  POST Server Interface Setup Configurations 11-63
Server Setup 11-64
  GET Server Setup Configuration 11-64
  GET ALL Server Setup Configuration 11-65
  POST Server Setup Configuration 11-65
Recorder Setup 11-66
  GET Recorder Setup Configuration 11-66
  GET ALL Recorder Setup Configuration 11-67
  POST Recorder Setup Configuration 11-68
Routes 11-69
  GET Server Routes Configuration 11-69
  GET ALL Server Routes Configuration 11-69
  POST Server Routes Configuration 11-70
SNMP Agent 11-71
  GET Server SNMP Agent Configuration 11-71
  GET ALL Server SNMP Agent Configuration 11-71
  POST Server SNMP Agent Configuration 11-72
Server Level DNS 11-72
  GET Server Level DNS Configuration 11-72
  GET ALL Server Level DNS Configuration 11-73
  POST Server Level DNS Configuration 11-73
Server Level NTP 11-73
  GET Server Level NTP Configuration 11-73
  GET ALL Server Level NTP Configuration 11-74
  POST Server Level NTP Configuration 11-74
Server Level RTSP 11-75
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET Server Level RTSP Configuration</td>
<td>11-75</td>
</tr>
<tr>
<td>GET ALL Server Level RTSP Configuration</td>
<td>11-76</td>
</tr>
<tr>
<td>POST Server Level RTSP Configuration</td>
<td>11-76</td>
</tr>
<tr>
<td>Server Level FSI Setup</td>
<td>11-77</td>
</tr>
<tr>
<td>GET Server Level FSI Setup Configuration</td>
<td>11-77</td>
</tr>
<tr>
<td>GET ALL Server Level FSI Setup Configuration</td>
<td>11-78</td>
</tr>
<tr>
<td>POST Server Level FSI Setup Configuration</td>
<td>11-78</td>
</tr>
<tr>
<td>Server Level Logging</td>
<td>11-78</td>
</tr>
<tr>
<td>GET Server Level Logging Configuration</td>
<td>11-78</td>
</tr>
<tr>
<td>GET ALL Server Level Logging Configuration</td>
<td>11-79</td>
</tr>
<tr>
<td>POST Server Level Logging Configuration</td>
<td>11-79</td>
</tr>
<tr>
<td>Server Level Syslog</td>
<td>11-80</td>
</tr>
<tr>
<td>GET Server Level Syslog Configuration</td>
<td>11-80</td>
</tr>
<tr>
<td>GET ALL Server Level Syslog Configuration</td>
<td>11-80</td>
</tr>
<tr>
<td>POST Server Level Syslog Configuration</td>
<td>11-80</td>
</tr>
<tr>
<td>Server Level Backup Configuration</td>
<td>11-81</td>
</tr>
<tr>
<td>GET ALL Server Level Backup Configuration</td>
<td>11-81</td>
</tr>
<tr>
<td>POST Server Level Backup Configuration</td>
<td>11-83</td>
</tr>
<tr>
<td>Maintain Section Configuration API</td>
<td>11-86</td>
</tr>
<tr>
<td>System Thresholds</td>
<td>11-87</td>
</tr>
<tr>
<td>GET ALL System Thresholds Configuration</td>
<td>11-87</td>
</tr>
<tr>
<td>POST System Thresholds Configuration</td>
<td>11-87</td>
</tr>
<tr>
<td>Application Configuration</td>
<td>11-88</td>
</tr>
<tr>
<td>GET ALL Application Configuration</td>
<td>11-88</td>
</tr>
<tr>
<td>POST System Application Configuration</td>
<td>11-88</td>
</tr>
<tr>
<td>CDSM/VVIM Setup</td>
<td>11-89</td>
</tr>
<tr>
<td>GET ALL CDSM/VVIM Setup Configuration</td>
<td>11-89</td>
</tr>
<tr>
<td>POST System CDSM/VVIM Configuration</td>
<td>11-90</td>
</tr>
<tr>
<td>System Configuration</td>
<td>11-91</td>
</tr>
<tr>
<td>GET ALL System Configuration</td>
<td>11-91</td>
</tr>
<tr>
<td>POST System Configuration</td>
<td>11-91</td>
</tr>
<tr>
<td>Database Configuration</td>
<td>11-92</td>
</tr>
<tr>
<td>GET ALL Database Configuration</td>
<td>11-92</td>
</tr>
<tr>
<td>POST Database Configuration</td>
<td>11-92</td>
</tr>
<tr>
<td>Configuration Generator</td>
<td>11-93</td>
</tr>
<tr>
<td>GET ALL Configuration Generator</td>
<td>11-93</td>
</tr>
<tr>
<td>POST Configuration Generator</td>
<td>11-93</td>
</tr>
<tr>
<td>ID Management</td>
<td>11-94</td>
</tr>
<tr>
<td>GET ALL ID Management</td>
<td>11-94</td>
</tr>
<tr>
<td>POST ID Management</td>
<td>11-94</td>
</tr>
</tbody>
</table>
Stream Monitor Listener  11-95
  GET ALL Stream Monitor Listener  11-95
  POST Stream Monitor Listener  11-95
Maintain Section Backup  11-96
  GET ALL Maintain Section Backup  11-96
  POST ID Maintain Section Backup  11-97
Preface

This preface describes the audience, use, and organization of the Cisco VDS-TV API Guide. The preface also outlines the document conventions and support information. It contains the following sections:

- Document Revision History, page xiii
- Audience, page xiii
- Document Organization, page xiv
- Document Conventions, page xiv
- Related Documentation, page xv
- Obtaining Documentation and Submitting a Service Request, page xvi

Document Revision History

The Document Revision History table records technical changes to this document.

<table>
<thead>
<tr>
<th>Date</th>
<th>Change Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>April 2017</td>
<td>Added System, Array, Server and maintain section configuration APIs.</td>
</tr>
<tr>
<td>September 2016</td>
<td>Initial release</td>
</tr>
</tbody>
</table>

Audience

This application program interface (API) guide is written for the knowledgeable application programmer who understands the basic architecture of the Cisco TV Videoscape Distribution Suite (VDS) software product, eXtensible Markup Language (XML), and HTTP POST calls. This guide is not intended to direct the user on how to program in XML but limits itself to describing how related VDS APIs are used.
Document Organization

This document contains the following chapters:

<table>
<thead>
<tr>
<th>Chapter or Appendix</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 1, “Introduction to Cisco VDS-TV Software APIs”</td>
<td>Provides an overview of the Cisco TV VDS API messages.</td>
</tr>
<tr>
<td>Chapter 2, “Monitoring APIs”</td>
<td>Describes the Monitoring APIs, including streams, content, and devices.</td>
</tr>
<tr>
<td>Chapter 3, “Maintenance API”</td>
<td>Describes the Maintenance APIs.</td>
</tr>
<tr>
<td>Chapter 4, “RTSP APIs”</td>
<td>Describes API messages for the RTSP(^1) environment.</td>
</tr>
<tr>
<td>Chapter 5, “D5 Interface APIs”</td>
<td>Describes the D5 Interface APIs.</td>
</tr>
<tr>
<td>Chapter 6, “Package APIs”</td>
<td>Describes the Package APIs.</td>
</tr>
<tr>
<td>Chapter 7, “Replication Group APIs”</td>
<td>Describes the Replication group APIs.</td>
</tr>
<tr>
<td>Chapter 8, “PVR Scheduler Interface APIs”</td>
<td>Describes the PVR Scheduler APIs.</td>
</tr>
<tr>
<td>Chapter 9, “VBO APIs”</td>
<td>Describes the VBO APIs.</td>
</tr>
<tr>
<td>Chapter 10, “TV Playout APIs”</td>
<td>Describes the TV Playout API messages.</td>
</tr>
<tr>
<td>Chapter 11, “Configuration APIs”</td>
<td>Describes the Configuration API messages.</td>
</tr>
</tbody>
</table>

1. RTSP = real time streaming protocol.

Document Conventions

This guide uses the following conventions for command syntax descriptions and textual emphasis:

<table>
<thead>
<tr>
<th>Conventions</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>boldface</strong> font</td>
<td>Commands and keywords are in <strong>boldface</strong>.</td>
</tr>
<tr>
<td><em>italic</em> font</td>
<td>Arguments for which you supply values are in <em>italics</em>.</td>
</tr>
<tr>
<td>[ ]</td>
<td>Elements in square brackets are optional.</td>
</tr>
<tr>
<td>{ x</td>
<td>y</td>
</tr>
<tr>
<td>[ x</td>
<td>y</td>
</tr>
<tr>
<td>string</td>
<td>A nonquoted set of characters. Do not use quotation marks around the string or the string will include the quotation marks.</td>
</tr>
<tr>
<td><strong>screen</strong> font</td>
<td>Terminal sessions and information the system displays are in <strong>screen</strong> font.</td>
</tr>
<tr>
<td><strong>boldface screen</strong> font</td>
<td>Information you must enter is in <strong>boldface screen</strong> font.</td>
</tr>
<tr>
<td><em>italic screen</em> font</td>
<td>Arguments for which you supply values are in <em>italic screen</em> font.</td>
</tr>
<tr>
<td>^</td>
<td>The symbol ^ represents the key labeled Control—for example, the key combination ^D in a screen display means hold down the Control key while you press the D key.</td>
</tr>
</tbody>
</table>
Caution

Means reader be careful. In this situation, you might do something that could result in equipment damage or loss of data.

Note

Means reader take note. Notes contain helpful suggestions or references to materials not contained in this publication.

Tip

Means the following information might help you solve a problem.

Related Documentation

These documents provide complete information about the VDS, and are available at Cisco.com:

- Release Notes for the Cisco VDS-TV 4.6
- Cisco VDS-TV RTSP Software Configuration Guide, Release 4.6
- Cisco VDS-TV ISA Software Configuration Guide, Release 4.6
- Cisco VDS-TV Installation, Upgrade and Maintenance Guide, Release 4.6
- Cisco VDS-TV API Guide, Release 4.6
- Cisco Content Delivery Engine 110 Hardware Installation Guide
- Cisco Content Delivery Engine 205/220/250/280/420/460/470 Hardware Installation Guide
- Cisco UCS C220 M4 Server Installation and Service Guide
- Regulatory Compliance and Safety Information for Cisco Content Delivery Engines

You can access the software documents at:

You can access the hardware documents at:
Obtaining Documentation and Submitting a Service Request

For information on obtaining documentation, submitting a service request, and gathering additional information, see the monthly What’s New in Cisco Product Documentation, which also lists all new and revised Cisco technical documentation, at:


Subscribe to the What’s New in Cisco Product Documentation as a Really Simple Syndication (RSS) feed and set content to be delivered directly to your desktop using a reader application. The RSS feeds are a free service and Cisco currently supports RSS version 2.0.
Introduction to Cisco VDS-TV Software APIs

Cisco TV Videoscape Distribution Suite (VDS) software provides three sets of application program interfaces (APIs):

- Monitoring
- Maintenance
- Real Time Streaming Protocol (RTSP) Stream Diagnostics
- D5 Interface
- Package Details
- Replication Group
- PVR Scheduler Interface
- Video Backoffice (VBO)
- TV Playout
- Configuration

The Monitoring APIs use Hypertext Transfer Protocol (HTTP) GET message format for sending control messages between any HTTP client and the Cisco VDS.

The Maintenance APIs use Hypertext Transfer Protocol (HTTP) GET message format for restarting services such as AIM.

The RTSP Stream Diagnostics APIs use eXtensible Markup Language (XML) over HTTP for sending control messages between any HTTP client and the Cisco VDS.

The D5 Interface APIs use HTTP GET/POST message format for exchanging server configuration and server statistics between On Demand Resource Manager (ODRM) and Streaming Server component of Next Generation On Demand (NGOD) system.

The Package APIs use HTTP GET message format to retrieve the package status details from the Cisco VDS.

The Replication Group APIs use HTTP GET/POST message format to add/delete replication groups, assign servers to a replication group, un-assign servers from a replication group and retrieve the replication groups configured in the system.

The PVR Scheduler Interface APIs use HTTP POST message format to interact with the PVR Server.

The TV Playout APIs use the `curl` utility to retrieve content from the Cisco VDS using HTTP.

Configuration APIs use a REST client to upload and download configuration settings.

The APIs can be used to monitor the states of specified VDS functions, report on VDS configurations, or create VDS entities. Table 1-1 describes these APIs.
<table>
<thead>
<tr>
<th>API</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitoring</td>
<td>Returns information on content states, stream history, trick-mode history, as well as other information about the devices.</td>
</tr>
<tr>
<td>Maintenance</td>
<td>Provides a way to restart services such as AIM.</td>
</tr>
<tr>
<td>RTSP stream diagnostics</td>
<td>Returns information about streams and trick modes for a specified session ID in an RTSP environment, as well as stream information about a specified smart card. A smart card is a unique ID that represents a set-top box (STB).</td>
</tr>
<tr>
<td>D5 Interface</td>
<td>This interface is used by Streaming Server to notify the ODRM of streaming server configuration and streaming server statistics/current resource usage and is also used by ODRM to query the streaming server status and current resource usage.</td>
</tr>
<tr>
<td>Package Details</td>
<td>Returns information on package status, package history, package stats and packages in publishing queue management. It also provides details on SOAP Exports and SOAP Imports functions.</td>
</tr>
<tr>
<td>Replication Group</td>
<td>Returns information on replication group configuration list, unassigned replication servers list. Provide a way to add a replication group, delete a replication group, assign serves to a replication group and un-assign servers from a replication group.</td>
</tr>
<tr>
<td>VBO APIs</td>
<td>Returns information on content list and session play history.</td>
</tr>
<tr>
<td>TV Playout</td>
<td>Returns information on TV Playout schedules, content and barker streams currently playing, configured TV Playout channels, content and barker streams that were playing during a specified time interval, and all barker streams. Provides a way to import TV Playout schedules, create barker streams and playlists, and start and stop barker streams.</td>
</tr>
<tr>
<td>Configuration</td>
<td>Provides a way to upload and download configuration settings.</td>
</tr>
</tbody>
</table>
This chapter contains the following sections:

- Monitoring and RTSP Stream Diagnostic Interfaces, page 1-3
- TV Playout Interface, page 1-8

**Monitoring and RTSP Stream Diagnostic Interfaces**

This section describes the Monitoring and the RTSP Stream Diagnostic APIs.

**Connections**

Any HTTP client that can send a request to the Content Delivery System Manager (CDSM) in the proper format can be used to send the API messages.

Connections can be semi-persistent or persistent; that is, the connection can be used for a single request-response pair or multiple request-response pairs.

A standard set of HTTP headers is used for all HTTP requests and responses. These headers include content type, content length, and a sequence number.

The HTTP request transmitter is considered the client and the HTTP request receiver is considered the server. The client always initiates the connection. The client must either receive a response or time out a request before sending another request on the same connection. The same connection can be used for multiple requests. However, when the client sends a new request, if there is already a connection established for a previous request for which the client is expecting a response from the server, then the client must open a new connection for the new request.

The API messages require a bidirectional socket connection for sending an HTTP request and receiving an HTTP response. The persistent protocol maintains a connection between requests until either the client or server indicates that the socket should close. This is typical of HTTP/1.0.

Persistent connections are handled using the traditional mechanisms specified in HTTP. In HTTP/1.0, connections are persistent by default. The HTTP response must include the header “Connection: close” in order to indicate that the connection will be closed at the end of transmission. During periods when no messages are being exchanged, the client or server may close the connection to conserve resources. The recommended connection approach is to use the default HTTP/1.0 behavior and to use the same connection for all requests.

**HTTP Headers**

This section covers the HTTP header formats for the request messages and the response messages.

**Request Messages**

There are two different request message formats:

- Monitoring and Trick-mode Event Reconciliation requests use an HTTP GET message format
- RTSP Stream Diagnostics requests use an HTTP POST message format
Chapter 1  Introduction to Cisco VDS-TV Software APIs

Monitoring Requests

The Monitoring APIs use an HTTP GET message format. All parameters are included in the HTTP GET message.

Table 1-2 provides an overview of the expected input parameters for each request message.

Table 1-2  Monitoring Request Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Required or Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>messageType</td>
<td>Required</td>
<td>Always required.</td>
</tr>
<tr>
<td>fromDate</td>
<td>Required for time-based messages</td>
<td>Required for StreamHistory, TrickModeHistory, PlayServerHistory, Errors, and CapacityPlanning. Length of time between fromDate and toDate must not exceed one hour.</td>
</tr>
<tr>
<td>toDate</td>
<td>Required for time-based messages</td>
<td>Required for StreamHistory, TrickModeHistory, PlayServerHistory, Errors, and CapacityPlanning.</td>
</tr>
<tr>
<td>maxRows</td>
<td>Optional</td>
<td>Specifies the maximum number of rows to return for this result set. Available for all messages apart from Trick-mode Event Reconciliation.</td>
</tr>
<tr>
<td>fromOffset</td>
<td>Optional</td>
<td>Specifies the row offset to start returning for this result set. A zero-based offset. Available for all messages apart from Trick-mode Event Reconciliation.</td>
</tr>
<tr>
<td>dateFormat</td>
<td>Optional</td>
<td>Specifies the formatting of the fromDate and toDate parameters to be seconds or milliseconds since epoch. The default is seconds. The options are sec or ms.</td>
</tr>
<tr>
<td>Session</td>
<td>Optional</td>
<td>Specifies what type of error messages to retrieve from the VDS. The options are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1—Retrieve only session-related error messages.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 2—Retrieve only error messages not session-related (no session ID).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 3—Retrieve all error messages.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The default is 3.</td>
</tr>
<tr>
<td>StreamID</td>
<td>Mandatory for Trick-mode Event Reconciliation</td>
<td>ID of the stream for which a list of trick-modes events is being requested.</td>
</tr>
<tr>
<td>Action</td>
<td>Optional</td>
<td>Specifies the range of Streamers to include in the report. Available for the CapacityPlanning message. The options are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• OVERALL—Present capacity data for all Streamers and ISVs(^1) (also known as SSVs).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SERVICEGROUP—Filter the capacity data by the specified Service Group.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SERVER—Filter the capacity data by the specified Streamer or ISV.</td>
</tr>
<tr>
<td>serviceGroup</td>
<td>Mandatory if action is set to SERVICEGROUP</td>
<td>Service Group identifier.</td>
</tr>
<tr>
<td>serverID</td>
<td>Mandatory if action is set to SERVER</td>
<td>Streamer or ISV identifier.</td>
</tr>
</tbody>
</table>
### Table 1-2  Monitoring Request Parameters (continued)

| timeType | Optional | Valid values for the timeType parameter are as follows:
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>• HOUR—Provide peak bandwidth and stream count per hour for the specified date range.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• DAY—Provide peak bandwidth and stream count per day for the specified date range.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• WEEK—Provide peak bandwidth and stream count per week for the specified date range. Incomplete weeks are not returned. The start date determines the first day of the week. For example, if you specify Tuesday, the 2nd of November 2010 as the start date, the first week is calculated as spanning from Tuesday, the 2nd of November 2010 to Monday, the 8th of November 2010.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• MONTH—Provide peak bandwidth and stream count per month. Incomplete months are not returned. The start date determines the first day of the month. For example, if you specify Tuesday, the 2nd of November 2010 as the start date, the first month is calculated as spanning from Tuesday, the 2nd of November 2010 to Tuesday, the 30th of November 2010.</td>
</tr>
</tbody>
</table>
|           |          | • DAYHOURVIEW—Provide peak bandwidth and stream count per hour for each standard week within the specified date range.  
|           |          | • DAYHALFHOURVIEW—Provide peak bandwidth and stream count per thirty minute intervals for each standard week within the specified date range. |
|           |          | • DAYFIFTEENVIEW—Provide peak bandwidth and stream count per fifteen minute intervals for each standard week within the specified date range. |
|           |          | • DAYMINUTEVIEW—Provide peak bandwidth and stream count per minute for each standard week within the specified date range. |
|           |          | • DAYFIVEMINVIEW—Provide peak bandwidth and stream count per five minute intervals for each standard week within the specified date range. |

| asmConfig | Mandatory if the messageType is set to one of the below: StreamHistory, StreamListHistory, TrickModeHistory, TrickModeListHistory, PlayServerHistory, ContentState, ContentRange | Specifies the installation type. 1 denotes ISA and 2 denotes RTSP. The default is set to 1. |

1. ISV = Integrated Streamer-Vault.
2. A standard week is from Sunday to Saturday.
RTSP Stream Diagnostics Requests

For RTSP Stream Diagnostic requests, regardless of the request type, the same HTTP request header format is used. Basic request requirements include the following:

- All requests are sent by means of POST.
- Uniform Resource Identifier (U) specifies the root XML tag.
- HTTP version is 1.0.
- Entity bodies are used to convey XML data.

Required entity headers are the following:

- content-length: Bytes (length of XML data)
- content-type: text/xml
- cseq: Unique numeric ID

Other entity headers are optional. For example, the HTTP header may include a date header. However, non-required entity headers may be ignored by the server.

The specifies the root service for the message followed by the query string. The query string has the syntax of action=<xml root tag>. The always begins with the service name. For example, if the service name is PlayoutDetails and the XML root tag is <GetCurrentlyPlayingDetails>, the is /apis/PlayoutDetails?action=GetCurrentPlayingDetails.

The following example is the complete HTTP POST message:

```
POST /apis/PlayoutDetails?action=GetCurrentPlayingDetails HTTP/1.0
User-Agent: HTTPTool/1.01
Content-Type: text/xml
Content-Length: 60
CSeq: 123

<?xml version="1.0" encoding="utf-8"?> <GetCurrentlyPlayingDetails />
```

Response Messages

There are two different response message formats:

- Monitoring responses
- RTSP Stream Diagnostics responses

Monitoring Response

All response messages for the Monitoring APIs return an XML document.

RTSP Stream Diagnostics

The same HTTP response header is used regardless of the response type. The status code and status text in the response indicate whether the server received and processed the request. Some HTTP response messages consist of only the HTTP header, while others consist of both the HTTP header and XML message body.

The only required entity header is cseq, which is a unique numeric ID.
The only required entity header when doing a POST with the XML message body is the content-type, text/xml.

Other entity headers are optional. For example, the HTTP header may include a date header. However, non-required entity headers may be ignored by the client when processing the response.

The sequence number specified in the HTTP-response must match the sequence number in the HTTP request.

The status codes and status text are specific to the HTTP-response. Chapter 4, “RTSP APIs” provides a list of appropriate status codes for each RTSP message.

The following example is an RTSP Stream Diagnostics HTTP response with no XML body:

HTTP/1.0 200 OK
Date: Mon, 02 Jun 2008 22:50:45 GMT
CSeq: 123

The following example is an RTSP Stream Diagnostics HTTP response with the XML body (body not shown):

HTTP/1.0 200 OK
Date: Mon, 02 Jun 2008 22:50:45 GMT
Server: Apache/1.3.33 (UNIX) PHP/4.4.8
X-Powered-By: PHP/4.4.8
Connection: close
Content-Type: text/xml
CSeq: 123

**Message Flow for RTSP Stream Diagnostics APIs**

This section covers the HTTP message flow and possible causes for incomplete message transactions.

**Request and Response**

The client connects to the server, sends an HTTP request, and waits for an HTTP response. During this time, the client must not send any other requests using this connection. The client can establish a separate connection to send another request in parallel.

The server processes the request and sends an HTTP response to the client on the same connection on which it received the request. In the case of RTSP messages, the response must use the same sequence number that was specified in the request. When the client receives the response, it validates the sequence number before processing the response.

Based on the HTTP headers, the client or server may close the connection or the connection may be left open for subsequent requests from the client. The client can have several open connections to the server at any given time.

**Request Timeout**

The client sets a timer when it sends an HTTP request. This timer represents the maximum amount of time the client waits for a response from the server. For an RTSP request, a typical timeout period is five seconds.
If the client fails to receive an HTTP response before the timer expires, the client must consider the request as failed. The client can discard the request, immediately retry the request, or retry the request at a later time. The method of handling the failure is implementation-specific, and may vary by importance of the request type.

**Invalid Sequence Number**

When the client receives an HTTP response with a sequence number that differs from the request sequence number, the client discards the message and continues to wait for a valid response until the timeout threshold has been reached.

**Connection Lost**

During the HTTP transaction, if the connection to the server is lost at any time prior to receiving a valid HTTP response message, the client must consider the request as having failed. The client should immediately retry the request. If the retry fails, the client can discard the request or retry the request at a later time. The method of handling the failure is implementation-specific, and may vary by the importance of the request type.

In most cases, the server has started processing the request prior to the connection loss. The server should finish processing the request. The server must not establish a connection with the client in order to return the response. It is up to the client to reconnect and retransmit the request.

**TV Playout Interface**

The TV Playout APIs use the `curl` utility to retrieve content from the Cisco VDS using HTTP. The basic `curl` syntax is as follows:

```bash
curl [options] "url"
```

In the following example, `curl` is used to return a list of TV Playout channels from the specified URL and to send this output to the file `reply_1_5.xml`:

```bash
curl -o reply_1_5.xml "http://209.165.201.1/api/services/configure/system/outputchannels"
```

An HTTP POST method is used to provide the VDS with information. Using `curl`, the syntax of the POST message is as follows:

```bash
curl -o filename -F "fileupload=@xml_filename" *url*
```

The data is specified in the XML file. In the following example, the file `barker.xml`, which contains the configuration for creating a barker stream, is posted to the VDS:

```bash
curl -o reply_1_2.xml -F "fileupload=@xml_barker.xml" "http://209.165.201.1/api/services/configure/array/barkerstream"
```

The `barker.xml` file contains the following content:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<List xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:ws="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"
Type="Barker"
Name="Barker1"
Channel="NBC">
  <Content Name="BBX_00_102000004.mpg" Loops="2"/>
  <Content Name="BBX_00_102000005.mpg" Loops="4"/>
</List>
```
All response messages for the TV Playout APIs return an XML document.

For more information on the curl utility, see the manual (man) pages located at the following URL:

http://curl.haxx.se/docs/manpage.html

The TV Playout interface uses method overloading to allow some parameters to be specified in the request message. To overload an HTTP method, you specify the intended method using the key "_method=" appended to the URL. The following is an example of overloading an HTTP POST method with an HTTP GET method:

curl -o reply_1_3.xml -F "fileupload=@request.xml" "http://209.165.201.1/api/services/report/system/streams/playout/_method=GET"

In this example, the client requests a report of TV Playout streams for a period of time specified in the file last_modified.xml. The file includes the following elements:

<param name="FromDate" value="2010-5-30"/>
<param name="ToDate" value="2010-5-30"/>

The “TV Playout Errors” section on page 10-17 provides an overview of status codes and status text for TV Playout APIs.
Monitoring APIs

This chapter describes the format and content of the Monitoring API messages. The Monitoring API messages consist of the following:

- MaxErrorModifiedTime, page 2-2
- MaxStreamModifiedTime, page 2-2
- StreamHistory, page 2-3
- StreamListHistory, page 2-5
- TrickModeHistory, page 2-7
- TrickModeListHistory, page 2-9
- PlayServerHistory, page 2-11
- Errors, page 2-13
- Devices, page 2-15
- ServiceGroups, page 2-16
- ContentState, page 2-17
- Version, page 2-18
- ContentRange, page 2-19
- CapacityPlanning, page 2-21
- Trick-mode Event Reconciliation, page 2-23

Note

Devices, ServiceGroups, ContentState, Version, ContentRange, and Trick-mode Event Reconciliation response messages are not time-sensitive and always return the current state of the requested information.

All HTTP request messages for the Monitoring APIs follow the HTTP GET format. All response messages for the Monitoring APIs return an XML document.
MaxErrorModifiedTime

The MaxErrorModifiedTime request-response message returns the timestamp of the latest error in the database.

Request
Required: MaxErrorModifiedTime

Request Example
http://<VVIM/CDSM_HostIP>/everstream/api.php?messageType=MaxErrorModifiedTime

Response
One of the following HTTP status codes is returned:
- 200 Ok—Request was successful.
- 400 Bad Request—Request parameters were incomplete or invalid.
- 404 Not Found—MessageType was invalid.

The MaxErrorModifiedTimeResponse element is returned in the XML body response. Table 2-1 describes the XML body elements and attributes returned in the MaxErrorModifiedTimeResponse element.

Table 2-1 MaxErrorModifiedTimeResponse

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxErrorModifiedTime</td>
<td>—</td>
<td>list element</td>
<td>Element that contains the information.</td>
</tr>
</tbody>
</table>
| maxTime          | xs:integer |            | The latest time (since start of UNIX epoch time) that an error occurred in the database.
| curTime          | xs:integer |            | The current time on the CDSM (since the start of UNIX epoch time).¹           |

¹ UNIX epoch time is 1970-01-01T00:00:00Z.

Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<MaxErrorModifiedTimeResponse
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="../src/xsd/MaxErrorModifiedTime.xsd">
  <MaxErrorModifiedTime maxTime="1233884342" curTime="1233884888" />
</MaxErrorModifiedTimeResponse>
```

MaxStreamModifiedTime

The MaxStreamModifiedTime request-response message returns the timestamp of the latest stream that was modified in the database.

Request
Required: MaxStreamModifiedTime
StreamHistory

The StreamHistory request-response message returns a list of streams that were served during the specified period of time.

Note

We recommend using the Errors API call in association with the StreamHistory API call in order to view any error records pertaining to the session ID.

Request

Required: StreamHistory
Required: fromDate
Required: toDate

The length of time between fromDate and toDate must not exceed one hour. The request example uses the maxRows, fromOffset, and dateFormat optional parameters.
StreamHistory

Request Example
For ISA environment

http://<VVIM/CDSM_HostIP>/everstream/api.php?fromDate=1459745558&dateFormat=sec&toDate=1459746457&messageType=StreamHistory&asmConfig=1

For RTSP environment

http://<VVIM/CDSM_HostIP>/everstream/api.php?fromDate=1459745558&dateFormat=sec&toDate=1459746457&messageType=StreamHistory&asmConfig=2

Response
One of the following HTTP status codes are returned:
- 200 Ok—Request was successful.
- 400 Bad Request—Request parameters were incomplete or invalid.
- 404 Not Found—MessageType was invalid.

The StreamsResponse element is returned in the XML body response.

Table 2-3 describes the XML body elements and attributes returned in the StreamsResponse element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stream</td>
<td></td>
<td>list element</td>
<td>Element that contains the stream information.</td>
</tr>
<tr>
<td>startTime</td>
<td>xs:integer</td>
<td></td>
<td>Stream start time (since start of UNIX epoch time).</td>
</tr>
<tr>
<td>endTime</td>
<td>xs:integer</td>
<td></td>
<td>Stream end time (since start of UNIX epoch time).</td>
</tr>
<tr>
<td>dbTime</td>
<td>xs:integer</td>
<td></td>
<td>Time (since start of UNIX epoch time) this stream was recorded in the database.</td>
</tr>
<tr>
<td>tsidOutputs</td>
<td>xs:string</td>
<td></td>
<td>TSID² output.</td>
</tr>
<tr>
<td>macAddress</td>
<td>xs:string</td>
<td></td>
<td>MAC address of the STB³.</td>
</tr>
<tr>
<td>serverIpAddress</td>
<td>xs:string</td>
<td></td>
<td>IP address of the Streamer.</td>
</tr>
<tr>
<td>serviceGroup</td>
<td>xs:integer</td>
<td></td>
<td>Service Group used to send the stream.</td>
</tr>
<tr>
<td>sessionID</td>
<td>xs:string</td>
<td></td>
<td>Session ID associated with this stream.</td>
</tr>
<tr>
<td>streamType</td>
<td>xs:string</td>
<td></td>
<td>Stream type (either user or sys) is used to differentiate between user-requested streams and system streams.</td>
</tr>
<tr>
<td>contentName</td>
<td>xs:string</td>
<td></td>
<td>Name of the content.</td>
</tr>
<tr>
<td>bitrate</td>
<td>xs:integer</td>
<td></td>
<td>Rate the content is being streamed, in bits per second.</td>
</tr>
<tr>
<td>assetName</td>
<td>xs:string</td>
<td></td>
<td>Name used in the ADI⁴ metadata for a group of content objects that make up one asset.</td>
</tr>
</tbody>
</table>

1. UNIX epoch time is 1970-01-01T00:00:00Z.
2. TSID = Transport Stream ID.
3. STB = set-top box.
4. ADI = asset distribution interface.

Response Example

`<?xml version="1.0" encoding="UTF-8"?>
<StreamsResponse>`
StreamListHistory

The StreamListHistory API provides the ability to list content playlists for each session that occurred in the specified time period. The StreamListHistory API augments the StreamHistory API by providing information on single-content sessions and multi-content (playlist) sessions. The StreamHistory API provides information for single-content sessions.

The StreamListHistory request-response message returns a list of single content streams and multi-content streams (playlists) for all sessions that occurred during the specified period of time.

We recommend using the Errors API call in association with the StreamListHistory API call in order to view any error records pertaining to the session ID. See the “Errors” section on page 2-13 for more information.

Request

Required: StreamListHistory

Required: fromDate

Required: toDate

The length of time between fromDate and toDate must not exceed one hour. The request example uses the maxRows, fromOffset, and dateFormat optional parameters.

Request Example

For ISA environment

http://<VVIM/CDSM_HostIP>/everstream/api.php?fromDate=1459745558&toDate=1459746457&messageType=StreamListHistory&asmConfig=1

For RTSP environment

http://<VVIM/CDSM_HostIP>/everstream/api.php?fromDate=1459745558&toDate=1459746457&messageType=StreamListHistory&asmConfig=2
Response

One of the following HTTP status codes are returned:

- 200 Ok—Request was successful.
- 400 Bad Request—Request parameters were incomplete or invalid.
- 404 Not Found—MessageType was invalid.

The StreamsResponse element is returned in the XML body response.

Table 2-4 describes the XML body elements and attributes returned in the StreamsResponse element.

Table 2-4  StreamsResponse for StreamListHistory

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stream</td>
<td>—</td>
<td>list element</td>
<td>Element that contains the stream information.</td>
</tr>
<tr>
<td></td>
<td>startTime</td>
<td>xs:integer</td>
<td>Stream start time (since start of UNIX epoch time).&lt;sup&gt;1&lt;/sup&gt;</td>
</tr>
<tr>
<td></td>
<td>endTime</td>
<td>xs:integer</td>
<td>Stream end time (since start of UNIX epoch time).&lt;sup&gt;1&lt;/sup&gt;</td>
</tr>
<tr>
<td></td>
<td>dbTime</td>
<td>xs:integer</td>
<td>Time (since start of UNIX epoch time) this stream was recorded in the database.</td>
</tr>
<tr>
<td></td>
<td>tsidOutputs</td>
<td>xs:string</td>
<td>TSID&lt;sup&gt;2&lt;/sup&gt; output.</td>
</tr>
<tr>
<td></td>
<td>macAddress</td>
<td>xs:string</td>
<td>MAC address of the STB&lt;sup&gt;3&lt;/sup&gt;.</td>
</tr>
<tr>
<td></td>
<td>serverIpAddress</td>
<td>xs:string</td>
<td>IP address of the Streamer.</td>
</tr>
<tr>
<td></td>
<td>serviceGroup</td>
<td>xs:integer</td>
<td>Service Group used to send the stream.</td>
</tr>
<tr>
<td></td>
<td>sessionID</td>
<td>xs:string</td>
<td>Session ID associated with this stream.</td>
</tr>
<tr>
<td></td>
<td>streamType</td>
<td>xs:string</td>
<td>Stream type (either user or sys) is used to differentiate between user-requested streams and system streams.</td>
</tr>
<tr>
<td>Content</td>
<td>contentName</td>
<td>xs:string</td>
<td>Name of the content.</td>
</tr>
<tr>
<td></td>
<td>bitrate</td>
<td>xs:integer</td>
<td>Rate the content is being streamed, in bits per second.</td>
</tr>
<tr>
<td></td>
<td>assetName</td>
<td>xs:string</td>
<td>Name used in the ADI&lt;sup&gt;4&lt;/sup&gt; metadata for a group of content objects that make up one asset.</td>
</tr>
<tr>
<td></td>
<td>contentLength</td>
<td>xs:integer</td>
<td>Duration, in milliseconds, of the content.</td>
</tr>
<tr>
<td></td>
<td>segmentIndex</td>
<td>xs:integer</td>
<td>Identifies the content location in a playlist; for example, a value of 0 is the first content, 1 is the second content, and so on.</td>
</tr>
</tbody>
</table>

1. UNIX epoch time is 1970-01-01T00:00:00Z.
2. TSID = Transport Stream ID.
3. STB = set-top box.
4. ADI = asset distribution interface.

Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<StreamsResponse
  dateFormat="sec"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="../../src/xsd/Streams.xsd">
```

Cisco VDS-TV API Guide
TrickModeHistory

The TrickModeHistory request-response message returns a list of trick-mode actions for all sessions during the specified period of time.

Request

Required: **TrickModeHistory**

Required: **fromDate**

Required: **toDate**

The length of time between fromDate and toDate must not exceed one hour. The request example uses the maxRows, fromOffset, and dateFormat optional parameters.

Request Example

For ISA environment

http://<VVIM/CDSM_HostIP>/everstream/api.php?fromDate=1459745558&dateFormat=sec&toDate=1459746457&messageType=TrickModeHistory&asmConfig=1&fromOffset=0&maxRows=100000

For RTSP environment

http://<VVIM/CDSM_HostIP>/everstream/api.php?fromDate=1459745558&toDate=1459746457&messageType=TrickModeHistory&asmConfig=2&fromOffset=0&maxRows=100000
Response

One of the following HTTP status codes are returned:

- 200 Ok—Request was successful.
- 400 Bad Request—Request parameters were incomplete or invalid.
- 404 Not Found—Message Type was invalid.

The TrickModesResponse element is returned in the XML body response. Table 2-5 describes the XML body elements and attributes returned in the TrickModesResponse.

**Table 2-5  TrickModesResponse**

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>TrickMode</td>
<td>—</td>
<td>list element</td>
<td>Element that contains the trick-mode information.</td>
</tr>
<tr>
<td>NPT</td>
<td>xs:integer</td>
<td></td>
<td>The current normal play time. NPT$^1$ starts at 0 at the start of the video, advances in real time in normal play mode, decrements in reverse mode, and is fixed when the video is paused.</td>
</tr>
<tr>
<td>errorText</td>
<td>xs:string</td>
<td></td>
<td>Information on any errors that occurred during trick play.</td>
</tr>
<tr>
<td>eventTime</td>
<td>xs:integer</td>
<td></td>
<td>Time (since start of UNIX epoch time)$^2$ this trick mode was recorded in the database.</td>
</tr>
<tr>
<td>scale</td>
<td>xs:string</td>
<td></td>
<td>The direction and speed of play.</td>
</tr>
<tr>
<td>sessionID</td>
<td>xs:string</td>
<td></td>
<td>Session ID associated with this stream.</td>
</tr>
</tbody>
</table>

1. NPT = normal play time.
2. UNIX epoch time is 1970-01-01T00:00:00Z.

Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<TrickModesResponse
  dateTimeFormat="sec"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="../../src/xsd/TrickModes.xsd">
  <TrickMode
    NPT="0"
    errorText=""
    eventTime="1235013676"
    scale="0"
    sessionID="C0A800F0DB2A::1235013676:664506::13"/>
  <TrickMode
    NPT="0"
    errorText=""
    eventTime="1235013676"
    scale="1"
    sessionID="C0A800F0DB2A::1235013676:664506::13"/>
  <TrickMode
    NPT="0"
    errorText=""
    eventTime="1235014331"
    scale="0"
    sessionID="C0A800F0DB2A::1235013676:664506::13"/>
</TrickModesResponse>
```
The TrickModeListHistory API provides the ability to list content playlists for each session that occurred in the specified time period. The TrickModeListHistory API augments the TrickModeHistory API by providing information on single content session and multi-content (playlist) sessions. The TrickModeHistory API provides information for single-content sessions.

The TrickModeHistory request-response message returns a list of trick-mode actions for single content streams and multi-content streams (playlists) for all sessions that occurred during the specified period of time.

**Request**

**Required:** TrickModeListHistory

**Required:** fromDate

**Required:** toDate

The length of time between fromDate and toDate must not exceed one hour. The request example uses the maxRows, fromOffset, and dateFormat optional parameters.

**Request Example**

**For ISA environment**

http://<VVIM/CDSM_HostIP>/everstream/api.php?fromDate=1459745558&dateFormat=sec&toDate=1459746457&messageType=TrickModeListHistory&asmConfig=1&fromOffset=0&maxRows=100000

**For RTSP environment**

http://<VVIM/CDSM_HostIP>/everstream/api.php?fromDate=1459745558&dateFormat=sec&toDate=1459746457&messageType=TrickModeListHistory&asmConfig=2&fromOffset=0&maxRows=100000

**Response**

One of the following HTTP status codes are returned:

- 200 Ok—Request was successful.
- 400 Bad Request—Request parameters were incomplete or invalid.
- 404 Not Found—MessageType was invalid.

The TrickModesResponse element is returned in the XML body response.
Table 2-6 describes the XML body elements and attributes returned in the TrickModesResponse.

Table 2-6  TrickModesResponse for TrickModeListHistory

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>TrickMode</td>
<td>—</td>
<td>list element</td>
<td>Element that contains the trick-mode information.</td>
</tr>
<tr>
<td>NPT</td>
<td>xs:integer</td>
<td>The current normal play time. NPT(^1) starts at 0 at the start of the video, advances in real time in normal play mode, decrements in reverse mode, and is fixed when the video is paused.</td>
<td></td>
</tr>
<tr>
<td>errorText</td>
<td>xs:string</td>
<td>Information on any errors that occurred during trick play.</td>
<td></td>
</tr>
<tr>
<td>eventTime</td>
<td>xs:integer</td>
<td>Time (since start of UNIX epoch time)(^2) this trick mode was recorded in the database.</td>
<td></td>
</tr>
<tr>
<td>scale</td>
<td>xs:string</td>
<td>The direction and speed of play.</td>
<td></td>
</tr>
<tr>
<td>sessionID</td>
<td>xs:string</td>
<td>Session ID associated with this stream.</td>
<td></td>
</tr>
<tr>
<td>segmentIndex</td>
<td>xs:integer</td>
<td>Identifies the content segment the trick-mode occurred in. If the content object is not part of a playlist, but instead a single content object, the value is zero (0).</td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>xs:integer</td>
<td>LSCP op code.</td>
<td></td>
</tr>
<tr>
<td>mode</td>
<td>xs:integer</td>
<td>LSCP stream mode.</td>
<td></td>
</tr>
<tr>
<td>response</td>
<td>xs:integer</td>
<td>LSCP response code.</td>
<td></td>
</tr>
</tbody>
</table>

**Note**  Currently, the response attribute is not supported and always returns zero (0).

1. NPT = normal play time.
2. UNIX epoch time is 1970-01-01T00:00:00Z.

Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<TrickModesResponse
dateFormat="sec"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="../../src/xsd/TrickModes.xsd">
   <TrickMode NPT="0" errorText="" eventTime="1331582456" scale="0" sessionID="1331582456:693585001" segmentIndex="0" type="0" mode="0" response="0" />
   <TrickMode NPT="0" errorText="" eventTime="1331582456" scale="1" sessionID="1331582456:693585001" segmentIndex="0" type="160" mode="4" response="0" />
   <TrickMode NPT="0" errorText="" eventTime="1331582456" scale="1" sessionID="1331582456:693585001" segmentIndex="0" type="6" mode="4" response="0" />
   <TrickMode NPT="576920" errorText="" eventTime="1331583034" scale="10" sessionID="1331582456:693585001" segmentIndex="0" type="160" mode="4" response="0" />
   <TrickMode NPT="614372" errorText="" eventTime="1331583038" scale="0" sessionID="1331582456:693585001" segmentIndex="0" type="1" mode="1" response="0" />
   <TrickMode NPT="615791" errorText="" eventTime="1331583046" scale="-10" sessionID="1331582456:693585001" segmentIndex="0" type="6" mode="4" response="0" />
   <TrickMode NPT="576920" errorText="" eventTime="1331583034" scale="6" sessionID="1331582456:693585001" segmentIndex="0" type="6" mode="4" response="0" />
   <TrickMode NPT="614372" errorText="" eventTime="1331583038" scale="0" sessionID="1331582456:693585001" segmentIndex="0" type="1" mode="1" response="0" />
   <TrickMode NPT="615791" errorText="" eventTime="1331583046" scale="-10" sessionID="1331582456:693585001" segmentIndex="0" type="6" mode="4" response="0" />
   <TrickMode NPT="503550" errorText="" eventTime="1331583057" scale="1" sessionID="1331582456:693585001" segmentIndex="0" type="6" mode="4" response="0" />
   <TrickMode NPT="6334785" errorText="" eventTime="1331588888" scale="0" sessionID="1331582456:693585001" segmentIndex="0" type="161" mode="4" response="0" />
   <TrickMode NPT="6334785" errorText="" eventTime="1331588888" scale="0" sessionID="1331582456:693585001" segmentIndex="0" type="1" mode="1" response="0" />
</TrickModesResponse>
```
PlayServerHistory

The PlayServerHistory request-response message returns a list of play servers for each trick-mode action for all sessions during the specified period of time.

Request

Required: PlayServerHistory

Required: fromDate

Required: toDate

The length of time between fromDate and toDate must not exceed one hour. The request example uses the maxRows, fromOffset, and dateFormat optional parameters.
### Request Example

**For ISA environment**

http://<VVIM/CDSM_HostIP>/everstream/api.php?fromDate=1459745558&dateFormat=sec&toDate=1459746457&messageType=PlayServerHistory&asmConfig=1&fromOffset=0&maxRows=100000

**For RTSP environment**

http://<VVIM/CDSM_HostIP>/everstream/api.php?fromDate=1459745558&dateFormat=sec&toDate=1459746457&messageType=PlayServerHistory&asmConfig=2&fromOffset=0&maxRows=100000

### Response

One of the following HTTP status codes are returned:

- 200 Ok—Request was successful.
- 400 Bad Request—Request parameters were incomplete or invalid.
- 404 Not Found—MessageType was invalid.

The PlayServerResponse element is returned in the XML body response.

**Table 2-7 PlayServerResponse**

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>PlayServer</td>
<td>—</td>
<td>list element</td>
<td>Element that contains the trick-mode information.</td>
</tr>
<tr>
<td>NPT</td>
<td>xs:integer</td>
<td></td>
<td>The current normal play time. NPT(^1) starts at 0 at the start of the video, advances in real time in normal play mode, decrements in reverse mode, and is fixed when the video is paused.</td>
</tr>
<tr>
<td>errorText</td>
<td>xs:string</td>
<td></td>
<td>Information on any errors that occurred during trick play.</td>
</tr>
<tr>
<td>eventTime</td>
<td>xs:integer</td>
<td></td>
<td>Time (since start of UNIX epoch time) this trick mode was recorded in the database.(^2)</td>
</tr>
<tr>
<td>playServerID</td>
<td>xs:integer</td>
<td></td>
<td>Play server ID associated with this trick play.</td>
</tr>
<tr>
<td>scale</td>
<td>xs:string</td>
<td></td>
<td>The direction and speed of play.</td>
</tr>
<tr>
<td>sessionID</td>
<td>xs:string</td>
<td></td>
<td>Session ID associated with this stream.</td>
</tr>
</tbody>
</table>

1. NPT = normal play time.
2. UNIX epoch time is 1970-01-01T00:00:00Z.
Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<PlayServerResponse
dateFormat="sec"
xmласn xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="../../src/xsd/PlayServer.xsd">
<PlayServer NPT="0" errorText="" eventTime="1261026446" playServerID="51" scale="0"
sessionID="CO8A00F0DB2A:1261026446:569384:4725" />
<PlayServer NPT="0" errorText="" eventTime="1261026447" playServerID="52" scale="1"
sessionID="CO8A00F0DB2A:1261026446:569384:4725" />
<PlayServer NPT="0" errorText="" eventTime="1261026751" playServerID="51" scale="0"
sessionID="CO8A00F0DB2A:1261026446:569384:4725" />
<PlayServer NPT="0" errorText="" eventTime="1261026760" playServerID="51" scale="0"
sessionID="CO8A00F0DB2A:1261026761:49751:1276" />
<PlayServer NPT="0" errorText="" eventTime="1261026760" playServerID="52" scale="1"
sessionID="CO8A00F0DB2A:1261026761:49751:1276" />
<PlayServer NPT="16" errorText="" eventTime="1261026760" playServerID="52" scale="1"
sessionID="CO8A00F0DB2A:1261026761:49751:1276" />
<PlayServer NPT="6316" errorText="" eventTime="1261026768" playServerID="52" scale="2"
sessionID="CO8A00F0DB2A:1261026761:49751:1276" />
<PlayServer NPT="0" errorText="" eventTime="1261026824" playServerID="51" scale="0"
sessionID="CO8A00F0DB2A:1261026833:382133:1272" />
<PlayServer NPT="0" errorText="" eventTime="1261026833" playServerID="51" scale="0"
sessionID="CO8A00F0DB2A:1261026833:382133:1272" />
<PlayServer NPT="14" errorText="" eventTime="1261026833" playServerID="52" scale="1"
sessionID="CO8A00F0DB2A:1261026833:382133:1272" />
<PlayServer NPT="83528" errorText="" eventTime="1261026918" playServerID="52" scale="4"
sessionID="CO8A00F0DB2A:1261026833:382133:1272" />
<PlayServer NPT="217365" errorText="" eventTime="1261026953" playServerID="52" scale="8"
sessionID="CO8A00F0DB2A:1261026833:382133:1272" />
<PlayServer NPT="296029" errorText="" eventTime="1261026964" playServerID="52" scale="15"
sessionID="CO8A00F0DB2A:1261026833:382133:1272" />
<PlayServer NPT="398677" errorText="" eventTime="1261026973" playServerID="52" scale="-15"
sessionID="CO8A00F0DB2A:1261026833:382133:1272" />
<PlayServer NPT="320690" errorText="" eventTime="1261026980" playServerID="52" scale="-8"
sessionID="CO8A00F0DB2A:1261026833:382133:1272" />
<PlayServer NPT="0" errorText="" eventTime="1261027020" playServerID="52" scale="-8"
sessionID="CO8A00F0DB2A:1261026833:382133:1272" />
<PlayServer NPT="0" errorText="" eventTime="1261027087" playServerID="51" scale="0"
sessionID="CO8A00F0DB2A:1261026833:382133:1272" />
</PlayServerResponse>
```

Errors

The Errors request-response message returns a list of stream errors that occurred during the specified period of time.

**Request**

Required: Errors

Required: fromDate

Required: toDate

The length of time between fromDate and toDate must not exceed one hour. The request example uses the maxRows, fromOffset, and dateFormat optional parameters.
Request Example

http://<VIM/CDSM_HostIP>/everstream/api.php?fromDate=1459745558&dateFormat=sec&toDate=1459746457&messageType=Errors&fromOffset=0&maxRows=100000

Response

One of the following HTTP status codes is returned:

- 200 Ok—Request was successful.
- 400 Bad Request—Request parameters were incomplete or invalid.
- 404 Not Found—Message Type was invalid.

The ErrorsResponse element is returned in the XML body response. Table 2-8 describes the XML body elements and attributes returned in the ErrorsResponse element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error</td>
<td>—</td>
<td>list element</td>
<td>Element that contains the error information.</td>
</tr>
<tr>
<td></td>
<td>errorCode</td>
<td>xs:integer</td>
<td>Stream error code.</td>
</tr>
<tr>
<td></td>
<td>errorText</td>
<td>xs:string</td>
<td>Stream error description.</td>
</tr>
<tr>
<td></td>
<td>errorTime</td>
<td>xs:integer</td>
<td>Time (since start of UNIX epoch time¹) this error was recorded in the database.</td>
</tr>
<tr>
<td></td>
<td>macAddress</td>
<td>xs:string</td>
<td>MAC address of the STB.</td>
</tr>
<tr>
<td></td>
<td>serverIpAddress</td>
<td>xs:string</td>
<td>IP address of the Streamer.</td>
</tr>
</tbody>
</table>

¹ UNIX epoch time is 1970-01-01T00:00:00Z.

Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ErrorsResponse
dateFormat="sec"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:Errors="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="../src/xsd/Errors.xsd">
  <Error errorCode="5005" errorText="AVS_CMPT_OUT_OF_SVC" errorTime="1233875550"
    macAddress="" serverIpAddress="209.165.201.12" serviceGroup="934282940"
    sourceEventID="1233875550::1::1::75::70::4044::112" sessionId="" />
  <Error errorCode="5005" errorText="AVS_CMPT_OUT_OF_SVC" errorTime="1233875580"
    macAddress="" serverIpAddress="209.165.201.12" serviceGroup="934282940"
    sourceEventID="1233875580::1::1::75::70::4044::113" sessionId="" />
  <Error errorCode="5005" errorText="AVS_CMPT_OUT_OF_SVC" errorTime="1233875610"
    macAddress="" serverIpAddress="209.165.201.12" serviceGroup="934282940"
    sourceEventID="1233875610::1::1::75::70::4044::114" sessionId="" />
  <Error errorCode="5005" errorText="AVS_CMPT_OUT_OF_SVC" errorTime="1233875640"
    macAddress="" serverIpAddress="209.165.201.12" serviceGroup="934282940"
    sourceEventID="1233875640::1::1::75::70::4044::115" sessionId="" />
</ErrorsResponse>
```
Devices

The Devices request-response message returns a list of the VDS devices.

**Request**

**Required:** Devices

**Request Example**

http://<VVIM/CDSM_HostIP>/everstream/api.php?messageType=Devices

**Response**

One of the following HTTP status codes is returned:

- 200 Ok—Request was successful.
- 400 Bad Request—Request parameters were incomplete or invalid.
- 404 Not Found—MessageType was invalid.

The DevicesResponse element is returned in the XML body response.

**Table 2-9** describes the XML body elements and attributes returned in the DevicesResponse element.

**Table 2-9  DevicesResponse**

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device</td>
<td>—</td>
<td>list element</td>
<td>Element that contains the device information.</td>
</tr>
<tr>
<td>description</td>
<td>xs:string</td>
<td></td>
<td>Description of the device.</td>
</tr>
<tr>
<td>hostname</td>
<td>xs:string</td>
<td></td>
<td>Hostname of the device.</td>
</tr>
<tr>
<td>ipAddress</td>
<td>xs:string</td>
<td></td>
<td>IP address of the device.</td>
</tr>
<tr>
<td>type</td>
<td>xs:string</td>
<td></td>
<td>Device type (Streamer, Vault, or ISV¹).</td>
</tr>
</tbody>
</table>

¹. ISV = Integrated Streamer-Vault.

**Response Example**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<DevicesResponse
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="/.../src/xsd/Devices.xsd"
>
<Device description="Vault" hostname="vault316" ipAddress="209.165.201.2" type="Vault" />
<Device description="Vault" hostname="vault318" ipAddress="209.165.201.4" type="Vault" />
<Device description="Vault" hostname="vault319" ipAddress="209.165.201.5" type="Vault" />
<Device description="Streamer" hostname="stm44" ipAddress="209.165.201.6" type="Streamer" />
<Device description="Streamer" hostname="stm74" ipAddress="209.165.201.8" type="Streamer" />
<Device description="Streamer" hostname="stm75" ipAddress="209.165.201.9" type="Streamer" />
<Device description="Streamer" hostname="stm76" ipAddress="209.165.201.11" type="Streamer" />
<Device description="Streamer" hostname="stm77" ipAddress="209.165.201.12" type="Streamer" />
<Device description="Streamer" hostname="stm78" ipAddress="209.165.201.14" type="Streamer" />
<Device description="Streamer" hostname="stm79" ipAddress="209.165.201.16" type="Streamer" />
<Device description="Streamer" hostname="stm81" ipAddress="209.165.201.19" type="Streamer" />
<Device description="Streamer" hostname="stm82" ipAddress="209.165.201.20" type="Streamer" />
<Device description="Streamer" hostname="stm83" ipAddress="209.165.201.21" type="Streamer" />
</DevicesResponse>
```
ServiceGroups

The ServiceGroups request-response message returns a list of the service groups configured in the VDS.

Request

Required: ServiceGroups

Request Example

http://<VVIM/CDSM_HostIP>/everstream/api.php?messageType=ServiceGroups

Response

One of the following HTTP status codes is returned:

- 200 Ok—Request was successful.
- 400 Bad Request—Request parameters were incomplete or invalid.
- 404 Not Found—MessageType was invalid.

The ServiceGroupsResponse element is returned in the XML body response. Table 2-10 describes the XML body elements and attributes returned in the ServiceGroupsResponse element.

Table 2-10 ServiceGroupsResponse

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ServiceGroup</td>
<td>—</td>
<td>list element</td>
<td>Element that contains the service group information.</td>
</tr>
<tr>
<td>serviceGroup</td>
<td>xs:integer</td>
<td>Service group configured in the VDS.</td>
<td></td>
</tr>
<tr>
<td>TSIDOutput</td>
<td>xs:integer</td>
<td>The associated TSID out port on the QAM¹ device.²</td>
<td></td>
</tr>
<tr>
<td>bandwidth</td>
<td>xs:integer</td>
<td>The bandwidth usage of this service group.²</td>
<td></td>
</tr>
<tr>
<td>EdgeDeviceIP</td>
<td>xs:string</td>
<td>The IP address of the QAM device.²</td>
<td></td>
</tr>
</tbody>
</table>

¹ QAM = Quadrature Amplitude Modulation.
² Only for Asynchronous Serial Interface (ASI) streaming.

Response Example

The response example is for a VDS configured with Gigabit Ethernet streaming.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ServiceGroupsResponse
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="../../src/xsd/ServiceGroups.xsd">
  <ServiceGroup serviceGroup="21"/>
  <ServiceGroup serviceGroup="22"/>
  <ServiceGroup serviceGroup="73"/>
</ServiceGroupsResponse>
```
ContentState

The ContentState request-response message returns a list of the content in the VDS.

Request

Required: ContentState

Request Example

For ISA environment

http://<VVIM/CDSM_HostIP>/everstream/api.php?messageType=ContentState&asmConfig=1

For RTSP environment

http://<VVIM/CDSM_HostIP>/everstream/api.php?messageType=ContentState&asmConfig=2

Response

One of the following HTTP status codes is returned:

- 200 Ok—Request was successful.
- 400 Bad Request—Request parameters were incomplete or invalid.
- 404 Not Found—MessageType was invalid.

The ContentStateResponse element is returned in the XML body response. Table 2-11 describes the XML body elements and attributes returned in the ContentStateResponse element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ContentState</td>
<td>—</td>
<td>list element</td>
<td>Element that contains the content information.</td>
</tr>
<tr>
<td>contentName</td>
<td>xs:string</td>
<td></td>
<td>Name of the content.</td>
</tr>
<tr>
<td>dateIngested</td>
<td>xs:integer</td>
<td></td>
<td>The time (since start of UNIX epoch time) that the content was ingested into</td>
</tr>
<tr>
<td>digestIpAddress</td>
<td>xs:string</td>
<td></td>
<td>The IP address of the Vault that ingested the content.</td>
</tr>
<tr>
<td>filesize</td>
<td>xs:string</td>
<td></td>
<td>The size, in bytes, of the content file.</td>
</tr>
<tr>
<td>isaAdminState</td>
<td>xs:string</td>
<td></td>
<td>The administrative state of this content. The possible administrative states are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Unprovisioned—Content is loading.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- In Service—Content is available for streaming.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Out of Service—Content is not available for streaming.</td>
</tr>
<tr>
<td>isaOpState</td>
<td>xs:string</td>
<td></td>
<td>The operational state of this content. The possible operational states are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Created—Content is loading.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- In Service—Content is available for streaming.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Out of Service—Content is not available for streaming.</td>
</tr>
<tr>
<td>bitrate</td>
<td>xs:integer</td>
<td></td>
<td>The rate of ingest in bits per second (3750000 = 3.75 Mbps).</td>
</tr>
<tr>
<td>assetName</td>
<td>xs:string</td>
<td></td>
<td>Name used in the ADI metadata for a group of content objects that make up one asset.</td>
</tr>
</tbody>
</table>
The Version request-response message returns the version of the API running on the VDS.

Request
Required: Version

Request Example
http://<VVIM/CDSM_HostIP>/everstream/api.php?messageType=Version

Response
One of the following HTTP status codes is returned.
- 200 Ok—Request was successful.
- 400 Bad Request—Request parameters were incomplete or invalid.
- 404 Not Found—MessageType was invalid.

The APIVersionResponse element is returned in the XML body response. Table 2-12 describes the XML body attributes returned in the APIVersionResponse.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>major</td>
<td>xs:integer</td>
<td>The integer portion of the API version.</td>
</tr>
<tr>
<td>minor</td>
<td>xs:integer</td>
<td>The decimal number portion of the API version.</td>
</tr>
</tbody>
</table>

Response Example
<?xml version="1.0" encoding="UTF-8"?>
<APIVersionResponse
  major="1"
  minor="3"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
ContentRange

The Content Range request-response message returns a list of content objects contained within the VDS network within a specified range.

Request
Required: ContentRange
Required: maxRows
Optionally, you can include the fromOffset parameter. The request example uses the maxRows parameter to request the first 500 content objects.

Request Example
For ISA environment
http://<VVIM/CDSM_HostIP>/everstream/api.php?fromDate=1459745558&dateFormat=sec&toDate=1459746457&messageType=ContentRange&asmConfig=1&fromOffset=0&maxRows=10

For RTSP environment
http://<VVIM/CDSM_HostIP>/everstream/api.php?fromDate=1459745558&dateFormat=sec&toDate=1459746457&messageType=ContentRange&asmConfig=2&fromOffset=0&maxRows=10

Response
One of the following HTTP status codes is returned.
• 200 Ok—Request was successful.
• 400 Bad Request—Request parameters were incomplete or invalid.
• 404 Not Found—MessageType was invalid.

The ContentRangeResponse element is returned in the XML body response.
Table 2-13 describes the XML body attributes returned in the ContentRangeResponse element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content</td>
<td>—</td>
<td>list element</td>
<td>Element that contains the content information.</td>
</tr>
</tbody>
</table>
The list of content objects returned is ordered alphabetically by content name.

**Response Example**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ContentRangeResponse
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="../../src/xsd/ContentRangeResponse.xsd">
 <Content Name="example.com::15825::DHW5_512_A1_01_11.mpg"
  isaAdminState="InService"
  isaOpState="InService"/>
 <Content Name="example.com::15828::DHW5_512_HD_A1_01_11.mpg"
  isaAdminState="InService"
  isaOpState="InService"/>
 <Content Name="example.com::15835::SCR8_803_A1_01_13.mpg"
  isaAdminState="InService"
  isaOpState="InService"/>

... 

<Content Name="example.com::16134::SCR8_803_HD_A1_01_13.mpg"
  isaAdminState="InService"
  isaOpState="InService"/>
</ContentRangeResponse>
```

In the following example, the request messages uses the name of the last content object returned from the previous request as the value of the fromOffset parameter and specifies that a maximum of 500 content objects be returned using the maxRows parameter:

http://209.165.201.1/everstream/api.php?messageType=ContentRange&fromOffset=example.com::16134::SCR8_803_HD_A1_01_13.maxRows=500

The response provides the next set of content objects, starting alphabetically from the next content object after example.com::16134::SCR8_803_HD_A1_01_13, up to a maximum of 500 content objects. For example:

```xml
<?xml version="1.0" encoding="UTF-8"?>
```
CapacityPlanning

The Capacity Planning Report request-response message returns capacity-related data from Streamers and ISVs (SSVs) to assist in monitoring the capacity of the VDS.

Request

Required: CapacityPlanning

Required: fromDate

Required: toDate

The length of time between fromDate and toDate must not exceed one hour. Optional parameters include the action and timeType parameters. If action is set to SERVICEGROUP, the serviceGroup parameter must be provided. If action is set to SERVER, the serverID parameter must be provided.

Request Examples

The following request example uses the action and timeType parameters to request peak bandwdith and peak stream counts per day for the specified time range:

For ISA environment

http://<VVIM/CDSM_HostIP>/everstream/api.php?fromDate=1459745558&dateFormat=sec&toDate=1459746457&messageType=CapacityPlanning&asmConfig=1&action=OVERALL&timeType=DAY

For RTSP environment

http://<VVIM/CDSM_HostIP>/everstream/api.php?fromDate=1459745558&dateFormat=sec&toDate=1459746457&messageType=CapacityPlanning&asmConfig=2&action=OVERALL&timeType=DAY

The following request example uses the action, timeType, and serviceGroup parameters to request peak bandwdith and peak stream counts per hour for the specified time range for all Streamers and ISVs in Service Group 5120:

For ISA environment
Chapter 2  Monitoring APIs

CapacityPlanning

http://<VVIM/CDSM_HostIP>/everstream/api.php?fromDate=1459745558&dateFormat=sec&toDate=1459746457&messageType=CapacityPlanning&asmConfig=1&action=SERVICEGROUP&timeType=HOUR&serviceGroup=5120

For RTSP environment
http://<VVIM/CDSM_HostIP>/everstream/api.php?fromDate=1459745558&dateFormat=sec&toDate=1459746457&messageType=CapacityPlanning&asmConfig=2&action=SERVICEGROUP&timeType=HOUR&serviceGroup=5120

The following request example uses the action, timeType, and Server ID parameters to request peak bandwidth and peak stream counts per week for the specified time range for Streamer 70:

For ISA environment
http://<VVIM/CDSM_HostIP>/everstream/api.php?fromDate=1459745558&dateFormat=sec&toDate=1459746457&messageType=CapacityPlanning&asmConfig=1&action=SERVERID&timeType=WEEK&serverID=70

For RTSP environment
http://<VVIM/CDSM_HostIP>/everstream/api.php?fromDate=1459745558&dateFormat=sec&toDate=1459746457&messageType=CapacityPlanning&asmConfig=2&action=SERVERID&timeType=WEEK&serverID=70

Response
One of the following HTTP status codes is returned.

- 200 Ok—Request was successful.
- 400 Bad Request—Request parameters were incomplete or invalid.
- 404 Not Found—MessageType was invalid.

The CapacityPlanning element is returned in the XML body response.

Table 2-13 describes the XML body attributes returned in the CapacityPlanning element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interval</td>
<td>—</td>
<td>list element</td>
<td>Element that contains the interval information.</td>
</tr>
<tr>
<td>starttime</td>
<td>xs:integer</td>
<td></td>
<td>Stream start time (since start of UNIX epoch time).¹</td>
</tr>
<tr>
<td>endtime</td>
<td>xs:integer</td>
<td></td>
<td>Stream end time (since start of UNIX epoch time).¹</td>
</tr>
<tr>
<td>hdstremas</td>
<td>xs:integer</td>
<td></td>
<td>Peek number of high-definition stream objects streamed during the specified time interval.</td>
</tr>
<tr>
<td>sdstreams</td>
<td>xs:integer</td>
<td></td>
<td>Peek number of standard-definition stream objects streamed during the specified time interval.</td>
</tr>
<tr>
<td>bandwidth</td>
<td>xs:integer</td>
<td></td>
<td>Peak bandwidth, in megabits, used for active streams during the specified time interval.</td>
</tr>
<tr>
<td>counter</td>
<td>xs:integer</td>
<td></td>
<td>Peak number of streams during the specified time interval.</td>
</tr>
</tbody>
</table>

¹ UNIX epoch time is 1970-01-01T00:00:00Z.

Response Example

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<CapacityPlanning
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="../src/xsd/CapacityPlanning.xsd">
```

Cisco VDS-TV API Guide
Trick-mode Event Reconciliation

The Trick-mode Event Reconciliation request-response message returns the list of trick-mode actions for the specified stream.

Note

The Trick-mode Event Reconciliation API can only be used on completed streams; that is streams that have been destroyed.
Trick-mode Event Reconciliation

Chapter 2  Monitoring APIs

Request
Required: GetTrickmodeEventList
Required: StreamID

StreamID is the ID of the stream for which a list of trick-modes events is being requested.

Request Example
http://<VVIM/CDSM_HostIP>/apis/TrickmodeEvents.php?messageType=GetTrickmodeEventsList&StreamID=0021be9ce7ba000008ed

Response
If the request is successful, the HTTP status code 200 OK is returned in the header and the TrickmodeEventResponse element is returned in the XML body response.

If no streamID was provided in the request message, the HTTP status code 400 Bad Request is returned in the header and the Error element is returned in the XML body element. The Error element contains the error code 101 and the error message ‘Missing Stream ID in request’. For example:

<Error code="101" message="Missing Stream ID in request"/>

If no messageType is provided in the request message, the HTTP status code 400 Bad Request is returned in the header and the Error element is returned in the XML body element. The Error element contains the error code 102 and the error message ‘Invalid messageType in request’. For example:

<Error code="102" message="Invalid messageType in request"/>

Table 2-1 describes the XML body elements and attributes returned in the TrickmodeEventResponse element.

Table 2-15  TrickmodeEventResponse

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>TrickmodeEventResponse</td>
<td>—</td>
<td>list element</td>
<td>Element that contains the trick-mode event reconciliation response.</td>
</tr>
<tr>
<td>TrickmodeEventList</td>
<td>Subelement</td>
<td></td>
<td>Element that contains the list of trick-mode actions.</td>
</tr>
<tr>
<td>TrickmodeEventList</td>
<td>—</td>
<td>list element</td>
<td>Element that contains the list of trick-mode actions.</td>
</tr>
<tr>
<td>StreamID</td>
<td>xs:string</td>
<td></td>
<td>ID of the stream.</td>
</tr>
<tr>
<td>Size</td>
<td>xs:nonNegative Integer</td>
<td></td>
<td>Size of the stream.</td>
</tr>
<tr>
<td>TrickmodeEvent</td>
<td>Subelement</td>
<td></td>
<td>Element that contains information about a trick-mode action.</td>
</tr>
</tbody>
</table>
Table 2-15  TrickmodeEventResponse (continued)

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>TrickmodeEvent</td>
<td>—</td>
<td>list element</td>
<td>Element that contains information about a trick-mode action.</td>
</tr>
<tr>
<td>EventMode</td>
<td>xs:string</td>
<td></td>
<td>Type of trick-mode action. Valid values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• TV_Play</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• TM_PAUSE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• TM_STOP</td>
</tr>
<tr>
<td>ModeScaleNum</td>
<td>xs:string</td>
<td></td>
<td>Direction and speed of the trick-mode action.</td>
</tr>
<tr>
<td>ModeScaleDenom</td>
<td>xs:string</td>
<td></td>
<td>Divider which is always 1.</td>
</tr>
<tr>
<td>CurrentNPT</td>
<td>xs:string</td>
<td></td>
<td>The current normal play time. NPT&lt;sup&gt;1&lt;/sup&gt; starts at 0 at the start of the video, advances in real time in normal play mode, decrements in reverse mode, and is fixed when the video is paused.</td>
</tr>
<tr>
<td>EventTime</td>
<td>xs:string</td>
<td></td>
<td>Time (since start of UNIX epoch time) this trick-mode action was recorded in the database.</td>
</tr>
<tr>
<td>SegmentId</td>
<td>xs:nonNegative Integer</td>
<td></td>
<td>Index of the content segment starting from 1.</td>
</tr>
</tbody>
</table>

1. NPT = normal play time.
2. UNIX epoch time is 1970-01-01T00:00:00Z.

Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<TrickmodeEventResponse xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
  <TrickmodeEventList StreamID="looney.mpg::6/20/2011|22:59:47.996098::1" Size="7">
    <TrickmodeEvent EventMode="TM_PLAY" ModeScaleNum="1" ModeScaleDenom="1" CurrentNPT="0" EventTime="1308610790010" SegmentId="1"/>
    <TrickmodeEvent EventMode="TM_PAUSE" ModeScaleNum="0" ModeScaleDenom="1" CurrentNPT="53012" EventTime="1308610843021" SegmentId="1"/>
    <TrickmodeEvent EventMode="TM_PLAY" ModeScaleNum="10" ModeScaleDenom="1" CurrentNPT="53546" EventTime="1308610848414" SegmentId="1"/>
    <TrickmodeEvent EventMode="TM_PLAY" ModeScaleNum="1" ModeScaleDenom="1" CurrentNPT="107704" EventTime="1308610853834" SegmentId="1"/>
    <TrickmodeEvent EventMode="TM_PLAY" ModeScaleNum="-10" ModeScaleDenom="1" CurrentNPT="109976" EventTime="1308610857116" SegmentId="1"/>
    <TrickmodeEvent EventMode="TM_STOP" ModeScaleNum="1" ModeScaleDenom="1" CurrentNPT="50514" EventTime="1308610876434" SegmentId="1"/>
  </TrickmodeEventList>
</TrickmodeEventResponse>
```
The XML schema file for the Trick-mode Event Reconciliation message response is as follows:

```xml
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:ws="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"
            targetNamespace="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
    <xsd:element name="TrickmodeEventResponse">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="ws:TrickmodeEventList" minOccurs="0" maxOccurs="1" />
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="TrickmodeEventList">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="ws:TrickmodeEvent" minOccurs="0" maxOccurs="unbounded" />
            </xsd:sequence>
            <xsd:attribute name="StreamID" type="xsd:string" use="required" />
            <xsd:attribute name="Size" type="xsd:nonNegativeInteger" use="required" />
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="TrickmodeEvent">
        <xsd:complexType>
            <xsd:attribute name="EventMode" type="xsd:string" use="required" />
            <xsd:attribute name="ModeScaleNum" type="xsd:string" use="required" />
            <xsd:attribute name="ModeScaleDenom" type="xsd:string" use="required" />
            <xsd:attribute name="CurrentNPT" type="xsd:string" use="required" />
            <xsd:attribute name="EventTime" type="xsd:string" use="required" />
            <xsd:attribute name="SegmentID" type="xsd:nonNegativeInteger" use="required" />
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```
Maintenance API

This chapter describes the format and content of the VDS-TV Maintenance API messages. The Maintenance API messages consists of the following:

- **Service restart**, page 3-1

**Service restart**

The service restart request-response message restarts a service after a specified time.

**Note**
The service restart API can currently be used only for restarting AIM service.

**Request Type**
HTTP method supported: GET

**Request Format**

**Required value**
service - Name of the service to restart

**Optional values**
timeout - Time in seconds after which service restart is triggered
serverid - ServerID of the master vault server

**Request Example**

If the serverid is not specified, then the master vault serverid is used as default.
If the timeout is not specified, then 120 is used as default timeout.
Response

If the request succeeds, the following response is returned:

```xml
<?xml version="1.0" encoding="UTF-8"?><Status>Successfully sends a trigger to restart AIM</Status>
```

If the request fails, then one of the following responses is returned:

```xml
<?xml version="1.0" encoding="UTF-8"?><Status>Invalid Master Vault server id</Status>
```

```xml
<?xml version="1.0" encoding="UTF-8"?><Status>Timeout to Restart AIM must be in the range of 30 to 600 secs</Status>
```
RTSP APIs

This chapter describes the format and content of the RTSP API messages. The RTSP API messages are used for RTSP environments and consist of the following:

- StreamsBySmartcard, page 4-1
- GetStreamDetailsBySessionId, page 4-3
- GetStreamTrickmodesBySessionId, page 4-4

All HTTP request messages follow the format given in the “HTTP Headers” section on page 1-3.

Stream Diagnostic APIs

StreamsBySmartcard

The StreamsBySmartcard request-response message returns the streams associated with a specified smart card. A smart card is a unique identifier (ID) that represents the STB.

Request
Element: StreamsBySmartcard

Table 4-1 describes the XML attributes for the StreamsBySmartcard request.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smartcard</td>
<td>xs:string</td>
<td>Unique ID for the STB&lt;sup&gt;1&lt;/sup&gt;.</td>
</tr>
<tr>
<td>Starttime</td>
<td>xs:integer</td>
<td>Stream start time. The time and date is represented in seconds since the start of UNIX epoch time.&lt;sup&gt;2&lt;/sup&gt;</td>
</tr>
<tr>
<td>Endtime</td>
<td>xs:integer</td>
<td>Stream end time. The time and date is represented in seconds since the start of UNIX epoch time.&lt;sup&gt;2&lt;/sup&gt;</td>
</tr>
</tbody>
</table>

1. STB = set-top box.
2. UNIX epoch time is 1970-01-01T00:00:00Z.
Request Example

POST /rtsp/SDI?action=StreamsBySmartcard HTTP/1.0
User-Agent: HTTPTool/1.0
Content-Type: text/xml
Cseq: 123
Content-Length: 128
<?xml version="1.0" encoding="utf-8"?>
<StreamsBySmartcard
  Smartcard = "123456789"
  Starttime = "1193782335"
  Endtime   = "1193782395"
/>

Note
If you do not specify a Starttime and Endtime, all the streams that the STB has sent are listed. The Starttime and Endtime parameters represent the range to use for searching the start time of the streams. The response returns all the streams that have a start time that falls within Starttime and Endtime range.

Response

One of the following HTTP status codes is returned:

- 200 Ok—Request was successful.
- 400 Bad Request—Request parameters were incomplete or invalid.

The StreamsBySmartcardResponse element is returned in the XML body response.

Table 4-2 describes the XML body elements and attributes returned in the StreamsBySmartcardResponse element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>StreamObject</td>
<td>—</td>
<td>Element</td>
<td>Element that contains the stream information.</td>
</tr>
<tr>
<td>SessionId</td>
<td>xs:string</td>
<td>SessionID</td>
<td>Session ID associated with this stream.</td>
</tr>
<tr>
<td>Content</td>
<td>xs:string</td>
<td>Name</td>
<td>Name of content.</td>
</tr>
<tr>
<td>Starttime</td>
<td>xs:integer</td>
<td>Stream Time</td>
<td>Stream start time (since start of UNIX epoch time).¹</td>
</tr>
<tr>
<td>Endtime</td>
<td>xs:integer</td>
<td>Stream Time</td>
<td>Stream end time (since start of UNIX epoch time).¹</td>
</tr>
</tbody>
</table>

¹. UNIX epoch time is 1970-01-01T00:00:00Z.

Response Example

<?xml version="1.0" encoding="utf-8"?>
<StreamsBySmartcardResponse>
  <StreamObject
    SessionId = "123456789"
    Content   = "daybreak.mpg"
    Starttime = "1193782335"
    Endtime   = "1193782395"
  />
  <StreamObject
    SessionId = "123456790"
    Content   = "looney.mpg"
    Starttime = "1193782400"
    Endtime   = "1193782500"
  />
</StreamsBySmartcardResponse>
GetStreamDetailsBySessionId

The GetStreamDetailsBySessionId request-response message returns a list of the stream details for a specified session ID.

**Request**
Element: GetStreamDetailsBySessionId
Attribute: SessionId

The SessionId attribute is a string data type.

**Request Example**

```plaintext
POST /rtsp/SDI?action=GetStreamDetailsBySessionId HTTP/1.0
    User-Agent: HTTPTool/1.0
    Content-Type: text/xml
    Cseq: 123
    Content-Length: 103

<?xml version="1.0" encoding="utf-8"?>
<GetStreamsDetailsBySessionId SessionId = "84127424038679804" />
```

**Response**

One of the following HTTP status codes is returned:
- 200 Ok—Request was successful.
- 400 Bad Request—Request parameters were incomplete or invalid.

The GetStreamDetailsBySessionIdResponse element is returned in the XML body response.

**Table 4-4** describes the XML body elements and attributes returned in the GetStreamDetailsBySessionIdResponse element.

**Table 4-3**  
<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>StreamDetails</td>
<td>—</td>
<td>Element</td>
<td>Element that contains the stream details.</td>
</tr>
<tr>
<td>SessionId</td>
<td>xs:string</td>
<td></td>
<td>Session ID associated with this stream.</td>
</tr>
<tr>
<td>Content</td>
<td>xs:string</td>
<td></td>
<td>Name of content.</td>
</tr>
<tr>
<td>Starttime</td>
<td>xs:integer</td>
<td></td>
<td>Stream start time (since start of UNIX epoch time).¹</td>
</tr>
<tr>
<td>Endtime</td>
<td>xs:integer</td>
<td></td>
<td>Stream end time (since start of UNIX epoch time).¹</td>
</tr>
<tr>
<td>DestinationIP</td>
<td>xs:string</td>
<td></td>
<td>Destination IP address for this stream.</td>
</tr>
<tr>
<td>DestinationPort</td>
<td>xs:integer</td>
<td></td>
<td>Destination port for this stream.</td>
</tr>
<tr>
<td>ServiceGroup</td>
<td>xs:integer</td>
<td></td>
<td>Service Group used to send the stream.</td>
</tr>
<tr>
<td>AllocatedBandwidth</td>
<td>xs:integer</td>
<td></td>
<td>Bandwidth allocated, in bits per second, for this stream.</td>
</tr>
</tbody>
</table>

¹ UNIX epoch time is 1970-01-01T00:00:00Z.
Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<GetStreamDetailsBySessionIdResponse>
  <StreamDetails>
    <SessionId>84127424038679804</SessionId>
    <StartTime>1236382972</StartTime>
    <EndTime>1236383190</EndTime>
    <DestinationIP>192.168.2.1</DestinationIP>
    <DestinationPort>8198</DestinationPort>
    <ServiceGroup>1234</ServiceGroup>
    <AllocatedBandwidth>3750000</AllocatedBandwidth>
  </StreamDetails>
</GetStreamDetailsBySessionIdResponse>
```

**GetStreamTrickmodesBySessionId**

The GetStreamTrickmodesBySessionId request-response message returns a list of trick modes for a specified session ID.

**Request**

Element: **GetStreamTrickmodesBySessionId**

Attribute: **SessionId**

The SessionId attribute is a string data type.

**Request Example**

```xml
POST /rtsp/SDI?action=GetStreamTrickmodesBySessionId HTTP/1.0
User-Agent: HTTPTool/1.0
Content-Type: text/xml
Cseq: 123
Content-Length: 105

<?xml version="1.0" encoding="utf-8"?>
<GetStreamTrickmodesBySessionId SessionId="123456789"/>
```

**Response:**

One of the following HTTP status codes is returned:

- 200 OK—Request was successful.
- 400 Bad Request—Request parameters were incomplete or invalid.

The GetStreamTrickmodesBySessionIdResponse element is returned in the XML body response. Table 4-4 describes the XML body elements and attributes returned in the GetStreamTrickmodesBySessionIdResponse element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>TrickmodeAction</td>
<td>--</td>
<td>list element</td>
<td>Element that contains the trick-mode information.</td>
</tr>
<tr>
<td>ActionTime</td>
<td>xs:date-Time</td>
<td>UTC time when the trick mode occurred. Date and time are specified in ISO 8601:2000 format.</td>
<td></td>
</tr>
<tr>
<td>NPTOffset</td>
<td>xs:integer</td>
<td>Current NPT offset of content in milliseconds.</td>
<td></td>
</tr>
<tr>
<td>Speed</td>
<td>xs:integer</td>
<td>Play speed change that occurred at the ActionTime.</td>
<td></td>
</tr>
</tbody>
</table>
1. UTC = coordinated universal time.
2. NPT = normal play time.

**Response Example:**

```
<?xml version="1.0" encoding="UTF-8"?>
<GetStreamTrickmodesBySessionIdResponse>
  <TrickmodeList>
    <TrickmodeAction>
      <ActionTime>"2009-03-06T23:43:56Z"</ActionTime>
      <NPTOffset>0</NPTOffset>
      <Speed>1</Speed>
    </TrickmodeAction>
  </TrickmodeList>
</GetStreamTrickmodesBySessionIdResponse>
```
D5 Interface APIs

The Next Generation Session and Resource Management (NG-SRM) architecture is an evolution of the Next Generation On Demand (NGOD) architecture intended to improve scalability, performance, and service velocity.

NG-SRM architecture adopts web services as the standard communication mechanism between different components.

D5 is HTTP over XML based interface between On Demand Resource Manager (ODRM) and Streaming Server component of Next Generation On Demand (NGOD) system.

This two way interface is used by Streaming Server to notify the ODRM of streaming server configuration and streaming server statistics/current resource usage and is also used by ODRM to query the streaming server status and current resource usage.

This chapter describes the format and content of the following D5 Interface APIs:

- SetStreamingServerConfig, page 5-1
- SetStreamingServerStatus, page 5-4
- GetStreamingServerStatus, page 5-5

SetStreamingServerConfig

The Streaming Server invokes the HTTP POST method to notify the ODRM of the Streaming Server configuration information. Since the configuration does not change frequently, this message is not exchanged often.

**Note**

The operational status is sent in a different message. Thus a configured device that suddenly has an output port failure will not report a configuration change, but rather reports a status change.

**Request**

Element: StreamingServerconfig

**Request Format**

http://<device-url>/odrm/StreamingServerConfig/<StreamingServerName>

**Response**

One of the following HTTP status codes is returned:
- 200 Ok—Request was successful.
- 400 Bad Request—Request parameters were incomplete or invalid.

The StreamingServerConfig element is returned in the XML body response. Table 5-1 describes the XML body elements and attributes returned in the StreamingServerConfig element.

**Table 5-1**  
<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetStreamingServerConfig</td>
<td>—</td>
<td>List element</td>
<td>Streaming Server Config message.</td>
</tr>
<tr>
<td>xmlns</td>
<td>xs:string</td>
<td>Namespace for the XML which is always “urn:com:comcast:ngsrmd5”</td>
<td></td>
</tr>
<tr>
<td>protocolVersion</td>
<td>xs:integer</td>
<td>The protocol version number of the configuration. It is always “1” for the current version of software.</td>
<td></td>
</tr>
<tr>
<td>componentName</td>
<td>xs:string</td>
<td>Name of the Streaming Server defined in the configuration.</td>
<td></td>
</tr>
<tr>
<td>deviceStatus</td>
<td>xs:string</td>
<td>The operational status of the Streaming Server. Device Status across all ServerPorts in the Streaming Group is considered.</td>
<td></td>
</tr>
<tr>
<td>modelName</td>
<td>xs:string</td>
<td>User defined string to represent the Streaming Server model.</td>
<td></td>
</tr>
<tr>
<td>ManagementURLList</td>
<td>Subelement</td>
<td>List of control plane interface address on the streaming server.</td>
<td></td>
</tr>
<tr>
<td>ServerPortList</td>
<td>Subelement</td>
<td>A list of streaming output ports.</td>
<td></td>
</tr>
<tr>
<td>DeliveryProtocolList</td>
<td>Subelement</td>
<td>A list of supported content delivery protocols.</td>
<td></td>
</tr>
<tr>
<td>PolicyList</td>
<td>Subelement</td>
<td>A list of standard policy definitions.</td>
<td></td>
</tr>
<tr>
<td>ManagementURLList</td>
<td>—</td>
<td>List element</td>
<td>List of control plane interface address on the streaming server.</td>
</tr>
<tr>
<td>ServerPortList</td>
<td>—</td>
<td>List element</td>
<td>List of streaming output ports.</td>
</tr>
<tr>
<td>ServerPort</td>
<td>—</td>
<td>List element</td>
<td>The server port address.</td>
</tr>
<tr>
<td>ServerPortId</td>
<td>xs:string</td>
<td>Name or ID of the server port. It is composed of server name and port number.</td>
<td></td>
</tr>
<tr>
<td>ServerPortStatus</td>
<td>xs:string</td>
<td>The operational status of the server port.</td>
<td></td>
</tr>
<tr>
<td>maxBW</td>
<td>xs:integer</td>
<td>The Maximum aggregate bandwidth that can be delivered by a particular server port in kilobits per second.</td>
<td></td>
</tr>
<tr>
<td>routerLoopBackAddress</td>
<td>xs:string</td>
<td>The loopback address of the router to which the server port is connected.</td>
<td></td>
</tr>
</tbody>
</table>
Table 5-1  StreamingServerConfig

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeliveryProtocolList</td>
<td>—</td>
<td>List element</td>
<td>A list of supported content delivery protocols.</td>
</tr>
<tr>
<td></td>
<td>DeliveryProtocol</td>
<td>Sub element</td>
<td></td>
</tr>
<tr>
<td>PolicyList</td>
<td>—</td>
<td>List element</td>
<td>A list of standard policy definitions.</td>
</tr>
<tr>
<td></td>
<td>Policy</td>
<td>Sub element</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Name</td>
<td>xs:string</td>
<td>Name of the policy to be considered.</td>
</tr>
<tr>
<td></td>
<td>Value</td>
<td>xs:integer</td>
<td>Associated integer value of the policy.</td>
</tr>
</tbody>
</table>

Response Example:

```xml
<StreamingServerConfig
  xmlns="urn:com:comcast:ngsrm:d5"
  protocolVersion="1"
  componentName="VIP"
  deviceStatus="OPERATIONAL"
  modelName="HCL!!!CISCO!!!">
  <ManagementURLList>
    <ManagementURL>10.78.178.116:554</ManagementURL>
  </ManagementURLList>
  <ServerPortList>
    <ServerPort
      serverPortId="SOP1"
      serverPortStatus="OPERATIONAL"
      routerLoopBackAddress="192.169.102.212"
      maxBW="10208000"/>
    <ServerPort
      serverPortId="SOP2"
      serverPortStatus="OPERATIONAL"
      routerLoopBackAddress="192.169.103.213"
      maxBW="4640000"/>
    <ServerPort
      serverPortId="SOP3"
      serverPortStatus="OPERATIONAL"
      routerLoopBackAddress="192.169.104.5"
      maxBW="5568000"/>
    <ServerPort
      serverPortId="SOP5"
      serverPortStatus="OPERATIONAL"
      routerLoopBackAddress="192.169.105.5"
      maxBW="4640000"/>
    <ServerPort
      serverPortId="SOP6"
      serverPortStatus="OFFLINE"
      routerLoopBackAddress="192.169.106.200"
      maxBW="1856000"/>
    <ServerPort
      serverPortId="SOP7"
      serverPortStatus="OPERATIONAL"
      routerLoopBackAddress="192.169.107.200"
      maxBW="928000"/>
  </ServerPortList>
  <DeliveryProtocolList>
    <DeliveryProtocol>UDP</DeliveryProtocol>
  </DeliveryProtocolList>
</StreamingServerConfig>
```
SetStreamingServerStatus

The Streaming Server calls the HTTP POST method to notify the ODRM of the Streaming Server status and current resource usage details.

Request
Element: StreamingServerStatus

Request Format
http://<device-url>/odrm/StreamingServerStatus/<StreamingServerName>

Response
One of the following HTTP status codes is returned:
- 200 Ok—Request was successful.
- 400 Bad Request—Request parameters were incomplete or invalid.

The StreamingServerconfig element is returned in the XML body response.

Table 5-2 describes the XML body elements and attributes returned in the StreamingServerconfig element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>StreamingServerStatus</td>
<td>—</td>
<td>List element</td>
<td>The status of the streaming server.</td>
</tr>
<tr>
<td>component Name</td>
<td>xs:string</td>
<td></td>
<td>Name of the streaming server.</td>
</tr>
<tr>
<td>deviceStatus</td>
<td>xs:string</td>
<td></td>
<td>The operational status of the streaming server.</td>
</tr>
<tr>
<td>ServerPortStatusList</td>
<td>—</td>
<td>List element</td>
<td>The status of all the server ports reported in the configuration.</td>
</tr>
<tr>
<td>serverPortId</td>
<td>xs:string</td>
<td></td>
<td>The name or Id of the server port.</td>
</tr>
<tr>
<td>serverPortStatus</td>
<td>xs:string</td>
<td></td>
<td>The operational status of the server port is displayed.</td>
</tr>
<tr>
<td>streamCurrBW</td>
<td>xs:integer</td>
<td></td>
<td>Current Bandwidth usage on the server port for content delivery in kbps.</td>
</tr>
<tr>
<td>streamAvailBW</td>
<td>xs:integer</td>
<td></td>
<td>Available Bandwidth usage on the server port for content delivery in kbps.</td>
</tr>
</tbody>
</table>

Response Example:
```xml
<StreamingServerStatus
  xmlns="urn:com:comcast:ngsrm:d5"
  componentName="VIP"
>
GetStreamingServerStatus

The ODRM invokes the HTTP GET method to query the Streaming Server of its status and current resource usage information.

Request
Element: StreamingServerStatus

Request Format
http://<device-url>/ss/StreamingServerStatus/<StreamingServerName>

Response
One of the following HTTP status codes is returned:
- 200 Ok—Request was successful.
- 400 Bad Request—Request parameters were incomplete or invalid.
- 401 Unauthorized
- 403 Forbidden
- 404 Not Found
- 500 Internal Server Error
- 503 Service Unavailable

The StreamingServerStatus element is returned in the XML body response.
Table 5-3 describes the XML body elements and attributes returned in the StreamingServerStatus element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>StreamingServerStatus</td>
<td>—</td>
<td>List element</td>
<td>The status of the streaming server.</td>
</tr>
<tr>
<td>ServerPortStatus</td>
<td>Subelement</td>
<td></td>
<td>The operational status of the server port.</td>
</tr>
<tr>
<td>Component Name</td>
<td>—</td>
<td>xs:string</td>
<td>Name of the Streaming Server.</td>
</tr>
<tr>
<td>deviceStatus</td>
<td>Content</td>
<td>xs:string</td>
<td>The operational status of streaming server device.</td>
</tr>
<tr>
<td>ServerPortStatus</td>
<td>—</td>
<td>List element</td>
<td>The operational status of the server port.</td>
</tr>
<tr>
<td>serverPortId</td>
<td>xs:string</td>
<td></td>
<td>The name or Id of the server port.</td>
</tr>
<tr>
<td>serverPortStatus</td>
<td>xs:string</td>
<td></td>
<td>The operational status of the server port is displayed.</td>
</tr>
<tr>
<td>streamCurrBW</td>
<td>xs:integer</td>
<td></td>
<td>Current Bandwidth usage on the server port for content delivery in kbps.</td>
</tr>
<tr>
<td>streamAvailBW</td>
<td>xs:integer</td>
<td></td>
<td>Available Bandwidth usage on the server port for content delivery in kbps.</td>
</tr>
</tbody>
</table>

Response Example:

```xml
<StreamingServerStatus
    xmlns="urn:com:comcast:ngsrm:d5"
    componentName="VIP"
    deviceStatus="OPERATIONAL">
    <ServerPortStatusList>
        <ServerPortStatus serverPortId="SOP1"
            serverPortStatus="OPERATIONAL"
            streamCurrBW="3751"
            streamAvailBW="7420249"/>
        <ServerPortStatus serverPortId="SOP2"
            serverPortStatus="OPERATIONAL"
            streamCurrBW="0"
            streamAvailBW="3712000"/>
        <ServerPortStatus serverPortId="SOP3"
            serverPortStatus="OPERATIONAL"
            streamCurrBW="0"
            streamAvailBW="3712000"/>
        <ServerPortStatus serverPortId="SOP5"
            serverPortStatus="OPERATIONAL"
            streamCurrBW="0"
            streamAvailBW="3712000"/>
        <ServerPortStatus serverPortId="SOP6"
            serverPortStatus="OFFLINE"
            streamCurrBW="0"
            streamAvailBW="0"/>
    </ServerPortStatusList>
</StreamingServerStatus>
```
Package APIs

This chapter describes the format and content of the following package details APIs.

- Package List, page 6-1
- Package Status List, page 6-2
- Package History List, page 6-6
- SOAP Exports, page 6-12
- SOAP Imports, page 6-17
- SOAP Faults, page 6-18
- WSDL Definitions, page 6-19
- GET Publishing Queue Management List, page 6-25
- POST Packages in Publishing Queue, page 6-27

All HTTP request messages follow the format given in the “HTTP Headers” section on page 1-3.

Package List

The package list request-response message returns a list of all packages in the system.

**Request Type**

HTTP method supported: GET.

**Request Example**


**Request Examples for different filter criteria**

```xml
<?xml version="1.0" encoding="UTF-8"?><PackageList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><PackageDetail
PackageName='TVN-TVN06310800100011'/></PackageDetail><PackageDetail
PackageName='TVN-TVN06464600100011'/></PackageDetail></PackageList>
```

If the request fails, the following response is returned:

```xml
<?xml version="1.0" encoding="UTF-8"?><Error Message="No Package(s) found"/>
```

**Response**

If the request succeeds, the List element is returned in the XML body response.
Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<PackageList
 xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><PackageDetail
 PackageName='TVN-TVN06310800100011'></PackageDetail>
</PackageList>
```

If the request fails, the following response is returned:

```xml
<?xml version="1.0" encoding="UTF-8"?> <Error Message="No Package(s) found"/>
```

## Package Status List

The package status list request-response message returns status list of all packages in the system based on different filter criteria.

### Request Type

HTTP method supported: GET.

### Request Examples for different filter criteria

**Filter By: get all**


**Filter By: provider name**


**Filter By: product name**


**Filter By: package name**


**Filter By: market name**


**Filter By: publish status**


**Filter By: content name**

curl -H 'Accept: application/xml' -H 'Content-Type: application/xml' -X GET http://<cdsm_host>/api/services/monitor/system/id/packagestatus/content/movie

**Filter By: title**


**Filter By: titlebrief**


**Filter By: Asset ID**

curl -H 'Accept: application/xml' -H 'Content-Type: application/xml' -X GET
Package Status List

http://<cdsm_host>/api/services/monitor/system/id/packagestatus/assetid/AFRPO

```
curl -H "Accept: application/xml" -H "Content-Type: application/xml" -X GET
http://<cdsm_host>/api/services/monitor/system/id/packagestatus/assetid/143741
```

Filter By: package creation date
```
curl -H "Accept: application/xml" -H "Content-Type: application/xml" -X GET
```

Filter By: package deletion date
```
curl -H "Accept: application/xml" -H "Content-Type: application/xml" -X GET
```

Filter By: content creation date
```
curl -H "Accept: application/xml" -H "Content-Type: application/xml" -X GET
```

Filter By: content last modified date
```
curl -H "Accept: application/xml" -H "Content-Type: application/xml" -X GET
```

Filter By: license expiration date
```
curl -H "Accept: application/xml" -H "Content-Type: application/xml" -X GET
```

Filter By: Asset name
```
curl -H "Accept: application/xml" -H "Content-Type: application/xml" -X GET
http://<cdsm_host>/api/services/monitor/system/id/packagestatus/assetname/pricetest_3
```

Response

If the request fails, the following response is returned:

```xml
    <?xml version="1.0" encoding="UTF-8"?><Error ErrorCode="No Package(s) matching your query were found"/>
```

If the request succeeds, the List element is returned in the XML body response.

Table 6-1 describes the XML body elements, sub elements, and attributes returned in the List element.

<table>
<thead>
<tr>
<th>Table 6-1 PackageStatus List Element</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Element</strong></td>
</tr>
<tr>
<td>--------------</td>
</tr>
<tr>
<td>PackageStatusList</td>
</tr>
<tr>
<td>PackageDetail</td>
</tr>
<tr>
<td>PackageName</td>
</tr>
<tr>
<td>PackageAssetName</td>
</tr>
<tr>
<td>PackageAssetID</td>
</tr>
<tr>
<td>TitleAssetID</td>
</tr>
<tr>
<td>Element</td>
</tr>
<tr>
<td>--------------------------</td>
</tr>
<tr>
<td>Title</td>
</tr>
<tr>
<td>TitleBrief</td>
</tr>
<tr>
<td>LicenseExpire Date</td>
</tr>
<tr>
<td>PackageDelete Date</td>
</tr>
<tr>
<td>PackageCreateDate</td>
</tr>
<tr>
<td>PackageVersionMinor</td>
</tr>
<tr>
<td>MarketStatus</td>
</tr>
<tr>
<td>MarketName</td>
</tr>
<tr>
<td>StatusTime</td>
</tr>
<tr>
<td>Status</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>ErrCode</td>
</tr>
<tr>
<td>FTPStatus</td>
</tr>
<tr>
<td>FTPStatus</td>
</tr>
<tr>
<td>StatusTime</td>
</tr>
<tr>
<td>Status</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>ErrCode</td>
</tr>
</tbody>
</table>
Table 6-1  PackageStatus List Element (continued)

<table>
<thead>
<tr>
<th>Subelement</th>
<th>Element that represents an asset.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asset</td>
<td>—</td>
</tr>
<tr>
<td>PackageDetail</td>
<td>—</td>
</tr>
<tr>
<td>PackageAssetName</td>
<td>Name of the asset</td>
</tr>
<tr>
<td>PackageAssetID</td>
<td></td>
</tr>
<tr>
<td>TitleAssetID</td>
<td></td>
</tr>
<tr>
<td>Title</td>
<td></td>
</tr>
<tr>
<td>TitleBrief</td>
<td></td>
</tr>
<tr>
<td>LicenseExpireDate</td>
<td></td>
</tr>
<tr>
<td>PackageDeleteDate</td>
<td></td>
</tr>
<tr>
<td>PackageCreateDate</td>
<td></td>
</tr>
<tr>
<td>PackageVersionMajor</td>
<td></td>
</tr>
<tr>
<td>PackageVersionMinor</td>
<td></td>
</tr>
</tbody>
</table>

Response Example

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<PackageStatusList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
  <PackageDetail
    PackageName="Package1006"
    PackageAssetName="pricetest_3"
    PackageAssetID='AFRO0100001439794800'
    TitleAssetID='AFRO0200001439794800'
    Title='Movie'
    TitleBrief='Storytell'
    LicenseExpireDate='08-27-2015'
    PackageDeleteDate='08-28-2015'
    PackageCreateDate='08-16-2015'
    PackageVersionMajor='1'
    PackageVersionMinor='0'
  >
    <MarketStatus
      MarketName="BMS1"
      StatusTime="01-07-2015, 1:29:36 am"
      Status="Complete"
      ErrCode="None"></MarketStatus>
    <MarketStatus
      MarketName="HerMarket"
      StatusTime="12-23-2014, 9:40:52 am"
      Status="Failed"
      ErrCode="Corba exception raised while publishing the package"></MarketStatus>
    <MarketStatus
      MarketName="MyMarket"
      StatusTime="12-23-2014, 9:41:13 am"
      Status="Failed"
      ErrCode="Corba exception raised while publishing the package"></MarketStatus>
    <FTPStatus
      StatusTime="12-23-2014, 9:39:58 am"
      Status="Scheduled"
      ErrCode="None"></FTPStatus>
    <Asset AssetName="movie1006.mpg"></Asset>
    <Asset AssetName="poster1006.bmp"></Asset>
    <Asset AssetName="preview1006.mpg"></Asset>
  </PackageDetail>
</PackageStatusList>
```
Package History List

The package history list request-response message provides a complete history of the package flow in the system.

**Request Type**
HTTP method supported: GET.

**Request Example**

**Response**
If the request fails, the following response is returned:

```xml
<?xml version="1.0" encoding="UTF-8"?><PackageHistory><Failed status='Package does not have any history'></Failed></PackageHistory>
```

If the request succeeds, the List element is returned in the XML body response.

**Table 6-2** describes the XML body elements, subelements, and attributes returned in the List element.

**Table 6-2  PackageHistory List Element**

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes/Sub Elements</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>PackageHistory</td>
<td>—</td>
<td>List Element</td>
<td>Element that represents the complete package history in the system.</td>
</tr>
<tr>
<td>Package</td>
<td>Subelement</td>
<td></td>
<td>Element that represents a package.</td>
</tr>
<tr>
<td>Name</td>
<td></td>
<td>xs:string</td>
<td>Name of the package.</td>
</tr>
<tr>
<td>history</td>
<td></td>
<td>xs:string</td>
<td>The history version of the package.</td>
</tr>
<tr>
<td>ingest</td>
<td></td>
<td>subelement</td>
<td>Element that represents the ingestion start time.</td>
</tr>
<tr>
<td>starttime</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>market</td>
<td>Subelement</td>
<td></td>
<td>Element that represents the market to publish the package.</td>
</tr>
<tr>
<td>name</td>
<td>Subelement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ip</td>
<td></td>
<td>xs:string</td>
<td>IP Address of the VOD market site.</td>
</tr>
<tr>
<td>publish</td>
<td>Subelement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>start_time</td>
<td></td>
<td>xs:string</td>
<td>Element that represents the start time of package publishing.</td>
</tr>
</tbody>
</table>
Table 6-2  PackageHistory List Element (continued)

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>publish end_time</td>
<td>xs:string</td>
<td>Element that represents the end time of package publishing.</td>
</tr>
<tr>
<td>publish protocol</td>
<td>xs:string</td>
<td>Element that represents the package publishing protocol.</td>
</tr>
<tr>
<td>publish status</td>
<td>xs:string</td>
<td>Element that represents the package publish status.</td>
</tr>
<tr>
<td>xml olddiff</td>
<td>Subelement</td>
<td>Element that represents the old ADI metadata of a package.</td>
</tr>
<tr>
<td>xmlolddiff</td>
<td>Package Subelement</td>
<td>Element that represents the old ADI metadata of a package.</td>
</tr>
<tr>
<td>AMS Asset_Name</td>
<td>xs:string</td>
<td>A string containing the identifying name of the asset. Asset names must be unique within a product.</td>
</tr>
<tr>
<td>Provider</td>
<td>xs:string</td>
<td>Name of the asset’s provider.</td>
</tr>
<tr>
<td>Product</td>
<td>xs:string</td>
<td>A unique identifier for the product (within the provider’s namespace).</td>
</tr>
<tr>
<td>Version_Major</td>
<td>xs:string</td>
<td>An integer representing the major version number (usually displayed before the decimal point: in Version 7.8, 7 is the major version number). “*” represents all versions.</td>
</tr>
<tr>
<td>Version_Minor</td>
<td>xs:string</td>
<td>An integer representing the minor version number (usually displayed after the decimal point: in Version 7.8, 8 is the minor version number). “*” represents all versions.</td>
</tr>
<tr>
<td>Description</td>
<td>xs:string</td>
<td>Description of the package.</td>
</tr>
<tr>
<td>Creation_Date</td>
<td>xs:string</td>
<td>The creation date of the package.</td>
</tr>
<tr>
<td>Provider_ID</td>
<td>xs:string</td>
<td>A unique identifier for the asset’s provider.</td>
</tr>
<tr>
<td>Asset_ID</td>
<td>xs:string</td>
<td>Asset_ID A string containing the identifying name of the asset. An Asset_ID shall uniquely identify an asset within a provider’s namespace defined by the Provider_ID attribute. All Asset_IDs will have a fixed length of 20, with the first 4 characters alpha and the last 16 characters numeric.</td>
</tr>
<tr>
<td>Asset_Class</td>
<td>xs:string</td>
<td>The category the asset belongs to.</td>
</tr>
<tr>
<td>Verb</td>
<td></td>
<td>A string containing an action to be performed on the asset. The only valid values for the Verb are the empty string (&quot;&quot;), and &quot;DELETE&quot;.</td>
</tr>
<tr>
<td>App_Data</td>
<td>Subelement</td>
<td>The application metadata.</td>
</tr>
<tr>
<td>App_Data</td>
<td>xs:string</td>
<td>The possible value for App are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• MOD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SVOD</td>
</tr>
<tr>
<td>Name</td>
<td>xs:string</td>
<td>Name of the application metadata.</td>
</tr>
</tbody>
</table>
### Table 6-2  PackageHistory List Element (continued)

<table>
<thead>
<tr>
<th>Value</th>
<th>xs:string</th>
<th>The corresponding value of the application metadata.</th>
</tr>
</thead>
<tbody>
<tr>
<td>xmlnewdiff Subelement</td>
<td>Element that represents the new ADI metadata of a package.</td>
<td></td>
</tr>
<tr>
<td>xmlnewdiff</td>
<td>Package Subelement</td>
<td>Element that represents the new ADI metadata of a package.</td>
</tr>
<tr>
<td>AMS</td>
<td>Asset_Name</td>
<td>A string containing the identifying name of the asset. Asset names must be unique within a product.</td>
</tr>
<tr>
<td>Provider</td>
<td>xx:string</td>
<td>Name of the asset’s provider.</td>
</tr>
<tr>
<td>Product</td>
<td>xx:string</td>
<td>A unique identifier for the product (within the provider’s namespace).</td>
</tr>
<tr>
<td>Version_Major</td>
<td>xx:string</td>
<td>An integer representing the major version number (usually displayed before the decimal point: in Version 7.8, 7 is the major version number). “*” represents all versions.</td>
</tr>
<tr>
<td>Version_Minor</td>
<td>xx:string</td>
<td>An integer representing the minor version number (usually displayed after the decimal point: in Version 7.8, 8 is the minor version number). “*” represents all versions.</td>
</tr>
<tr>
<td>Description</td>
<td>xx:string</td>
<td>Description of the package.</td>
</tr>
<tr>
<td>Creation_Date</td>
<td>xx:string</td>
<td>The creation date of the package.</td>
</tr>
<tr>
<td>Provider_ID</td>
<td>xx:string</td>
<td>A unique identifier for the asset’s provider.</td>
</tr>
<tr>
<td>Asset_ID</td>
<td>xx:string</td>
<td>Asset_ID A string containing the identifying name of the asset. An Asset_ID shall uniquely identify an asset within a provider’s namespace defined by the Provider_ID attribute. All Asset_IDs will have a fixed length of 20, with the first 4 characters alpha and the last 16 characters numeric.</td>
</tr>
<tr>
<td>Asset_Class</td>
<td>xx:string</td>
<td>A string containing the identifying name of the asset. Asset names must be unique within a product.</td>
</tr>
<tr>
<td>Verb</td>
<td>xx:string</td>
<td>A string containing an action to be performed on the asset. The only valid values for the Verb are the empty string (“”), and “DELETE”.</td>
</tr>
<tr>
<td>App_Data Subelement</td>
<td>The application metadata.</td>
<td></td>
</tr>
<tr>
<td>App_Data</td>
<td>xmlnewdiff Subelement</td>
<td>The application metadata.</td>
</tr>
<tr>
<td>App</td>
<td>xx:string</td>
<td>The possible value for App are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• MOD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SVOD</td>
</tr>
<tr>
<td>Name</td>
<td>xx:string</td>
<td>Name of the application metadata.</td>
</tr>
<tr>
<td>Value</td>
<td>xx:string</td>
<td>The corresponding value of the application metadata.</td>
</tr>
</tbody>
</table>

**Response Example**

```xml
<?xml version="1.0" encoding="UTF-8"?>
```
<PackageHistory>
   <Package Name="Test_8312" history="0">
      <ingest starttime="1433219401"/>
      <market name="BMS1">
         <market ip="172.22.99.80"/>
         <publish start_time="9:30:04 PM PDT"/>
         <publish end_time="9:30:54 PM PDT"/>
         <publish protocol="FTP"/>
         <publish status="Failed"/>
      </market>
      <xml olddiff=" <AMS Asset_Name="Package_8312" Provider="HBO" Product="MOD" Version_Major="1" Version_Minor="0" Description="Test package Package_8312" Creation_Date="2014-12-17" Provider_ID="hbo.com" Asset_ID="TTVV8312100000000001" Asset_Class="package" Verb=""/>"> <xml newdiff=" <AMS Asset_Name="Package_8312" Provider="HBO" Product="MOD" Version_Major="2" Version_Minor="0" Description="Test package Package_8312" Creation_Date="2014-12-17" Provider_ID="hbo.com" Asset_ID="TTVV8312100000000001" Asset_Class="package" Verb=""/>
   </xml>
   </xml>
</Package>

<Package Name="Test_8312" history="1">
   <ingest starttime="1433257862"/>
   <market name="BMS1">
      <market ip="172.22.99.80"/>
      <publish start_time="8:11:36 AM PDT"/>
      <publish end_time="8:12:48 AM PDT"/>
      <publish protocol="FTP"/>
      <publish status="Failed"/>
   </market>
   <xml olddiff=" <AMS Asset_Name="Package_8312" Provider="HBO" Product="MOD" Version_Major="2" Version_Minor="0" Description="Test package Package_8312" Creation_Date="2014-12-17" Provider_ID="hbo.com" Asset_ID="TTVV8312100000000001" Asset_Class="package" Verb=""/>
   </xml>
   <xml newdiff=" <AMS Asset_Name="Package_8312" Provider="HBO" Product="MOD" Version_Major="3" Version_Minor="0" Description="Test package Package_8312" Creation_Date="2014-12-17" Provider_ID="hbo.com" Asset_ID="TTVV8312100000000001" Asset_Class="package" Verb="" />
   <App_Data App="ENTOD" Name="Actors" Value="davis,bill"/>
   </xml>
</Package>

<Package Name="Test_8312" history="2">
   <ingest starttime="1433283961"/>
   <market name="BMS1">
      <market ip="172.22.99.80"/>
      <publish start_time="3:26:23 PM PDT"/>
      <publish end_time="3:28:25 PM PDT"/>
      <publish protocol="FTP"/>
      <publish status="Failed"/>
   </market>
   <xml olddiff=" <AMS Asset_Name="Package_8312" Provider="HBO" Product="MOD" Version_Major="2" Version_Minor="0" Description="Test package Package_8312" Creation_Date="2014-12-17" Provider_ID="hbo.com" Asset_ID="TTVV8312100000000001" Asset_Class="package" Verb=""/>
   </xml>
   <xml newdiff=" <AMS Asset_Name="Package_8312" Provider="HBO" Product="MOD" Version_Major="3" Version_Minor="0" Description="Test package Package_8312" Creation_Date="2014-12-17" Provider_ID="hbo.com" Asset_ID="TTVV8312100000000001" Asset_Class="package" Verb="" />
   <App_Data App="ENTOD" Name="Distributor_Royalty_Flat_Rate" Value="15.549"/>
   </xml>
</Package>

<Package Name="Test_8312" history="3">
   <ingest starttime="1433285222"/>
   <market name="BMS1">
      <market ip="172.22.99.80"/>
      <publish start_time="3:47:50 PM PDT"/>
      <publish end_time="3:49:47 PM PDT"/>
      <publish protocol="FTP"/>
      <publish status="Failed"/>
   </market>
</Package>
Package Stats List

The package stats list request-response message provides statistics of package ingestion and package publish of list of packages in the system.

Request Type
HTTP method supported: GET.

Request Example


Response
If the request fails, the following response is returned:

<?xml version="1.0" encoding="UTF-8"?><PackageStats><Failed status='Package does not have any stats'></Failed></PackageStats>

If the request succeeds, the List element is returned in the XML body response.

Table 6-3 describes the XML body elements, subelements, and attributes returned in the List element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes/Sub Elements</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>PackageStatuses</td>
<td>—</td>
<td>List Element</td>
<td>Element that represents the complete package status in the system.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Package</td>
<td>Subelement</td>
<td>PackageStatus Subelement</td>
<td>Element that represents a package.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>xs:string</td>
<td></td>
<td>Name of the package.</td>
</tr>
<tr>
<td>history</td>
<td>xs:string</td>
<td></td>
<td>The history version of the package.</td>
</tr>
<tr>
<td>ingest starttime</td>
<td>subelement</td>
<td></td>
<td>Element that represents the ingestion start time.</td>
</tr>
<tr>
<td>ingest endtime</td>
<td>xs:string</td>
<td></td>
<td>Element that represents the ingestion end time.</td>
</tr>
<tr>
<td>size</td>
<td>xs:string</td>
<td></td>
<td>Element that represents the size of the package ingested.</td>
</tr>
<tr>
<td>duration</td>
<td>xs:string</td>
<td></td>
<td>Elements that represents the duration for package ingest.</td>
</tr>
</tbody>
</table>
Table 6-3 PackageStats List Element (continued)

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ingest_rate</td>
<td>xs:string Element that represents the rate at which the package was ingested.</td>
</tr>
<tr>
<td>major_version</td>
<td>xs:string Element that represents the major version number for a package.</td>
</tr>
<tr>
<td>minor_version</td>
<td>xs:string Element that represents the minor version number for a package.</td>
</tr>
<tr>
<td>market</td>
<td>Subelement Element that represents the market to publish the package.</td>
</tr>
<tr>
<td>name</td>
<td>xs:string Domain name of VOD market site.</td>
</tr>
<tr>
<td>ip</td>
<td>xs:string IP Address of the VOD market site.</td>
</tr>
<tr>
<td>publish start_time</td>
<td>xs:string Element that represents the start time of package publishing.</td>
</tr>
<tr>
<td>publish end_time</td>
<td>xs:string Element that represents the end time of package publishing.</td>
</tr>
<tr>
<td>publish protocol</td>
<td>xs:string Element that represents the package publishing protocol.</td>
</tr>
<tr>
<td>publish status</td>
<td>xs:string Element that represents the package publish status.</td>
</tr>
<tr>
<td>publish size</td>
<td>xs:string Element that represents the size of the package published.</td>
</tr>
<tr>
<td>publish duration</td>
<td>xs:string Element that represents the duration for package publish.</td>
</tr>
<tr>
<td>publish rate</td>
<td>xs:string Element that represents the rate at which the package was published.</td>
</tr>
</tbody>
</table>

Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<PackageStats>
  <Package Name="Test_8312">
    <ingest starttime="1434289382"/>
    <ingest endtime="1434611945"/>
    <size="333107800"/>
    <duration="7"/>
    <ingest_rate="47586828"/>
    <major_version="1"/>
    <minor_version="0"/>
    <market name="MKT4">
      <market ip="172.22.97.190"/>
      <publish start_time="6:43:04 AM PDT"/>
      <publish end_time="6:43:04 AM PDT"/>
      <publish protocol="ATIS"/>
      <publish status="Complete"/>
      <publish size="2003134838"/>
      <publish duration="0"/>
      <publish rate="0"/>
    </market>
    <market name="MKT3">
      <market ip="172.22.98.120"/>
    </market>
  </Package>
</PackageStats>
```
SOAP Exports

SOAP Exports APIs are used by third party vendors to interact with Cisco’s AIM to ingest/update/delete packages in VDS-TV RTSP deployment. This section describes the format and content of the following SOAP Exports.

- **IngestPackage**, page 6-12
- **DeletePackage**, page 6-13
- **UpdatePackage**, page 6-14
- **GetPackageStatus**, page 6-15
- **GetAllPackages**, page 6-16
- **GetAvailableDiskSpace**, page 6-17

**IngestPackage**

This function is used to ingest a package to AIM.

**Signature**

```c
SOAP_FMAC5 int SOAP_FMAC6 CISCOAIM__IngestPackage(
    struct soap *pSOAP,
    struct _CISCOAIM__IngestPackage *pIn,
    struct _CISCOAIM__IngestPackageResponse *pOut)
```

**Structs**

```c
struct _CISCOAIM__IngestPackage
{
    char *ADIURL;        /* optional element of type xsd:string */
    char *PackageName;   /* optional element of type xsd:string */
    int *MetaDataOnly;   /* optional element of type xsd:int */
    int *DoAsync;        /* optional element of type xsd:int */
    struct CISCOAIM__EndpointList *AsyncTargets; /* opt element CISCOAIM:EndpointList */
};

struct _CISCOAIM__IngestPackageResponse
{
    char *IngestResult;
};
```
struct CISCOAIM__EndpointList
{
    int __sizeServerIp; /* sequence of elements <ServerIp> */
    char **ServerIp;    /* optional element of type xsd:string */
    int __sizePort;     /* sequence of elements <Port> */
    int *Port;          /* optional element of type xsd:int */
    int __sizePath;     /* sequence of elements <Path> */
    char **Path;        /* optional element of type xsd:string */
};

Parameter Description

REQUEST:
ADIURL: The location of the package XML. ie. 'ftp://Im:Pass@my.box.info/here/ADI.XML'
PackageName: Name of the package.
MetaDataOnly: [0-NO|1-YES] Setting it to 1 enables the user to ingest only XML metadata.

Note: Currently VDS-TV does not support ingesting only XML metadata.

DoAsync: [0-NO|1-YES] By setting it to 1, AIM Immediately returns from the IngestPackage call and does not wait until ingestion is complete. This setting is useful for clients that are single threaded and processes one package at a time. AIM does a callback to the application using imported AIMPackageNotification interface. If AIMPackageNotification interface is not available then GetPackageStatus is used to obtain the status of package ingestion.
AsyncTargets: List of notification targets for an async action.

RESPONSE:
IngestResult: If ingestion is successful then SUCCESS is returned else an error string is returned. For detailed information on SOAP faults, refer SOAP Faults, page 6-18

DeletePackage

This function is used to delete a package from AIM.

Signature

SOAP_FMAC5 int SOAP_FMAC6 __CISCOAIM__DeletePackage
{
    struct soap *pSOAP,
    struct _CISCOAIM__DeletePackage *pIn,
    struct _CISCOAIM__DeletePackageResponse *pOut)

Structs

struct _CISCOAIM__DeletePackage
{
    char *PackageName;
    int MetaDataOnly; // Optional, default 0=NO
### UpdatePackage

This function is used to update the package details. The ADI XML is used to determine the updated information. For example, if new content is added to a package only the new content and it's corresponding metadata is ingested using this function.

#### Signature

```c
SOAP_FMAC5 int SOAP_FMAC6 __CISCOAIM__UpdatePackage(
    struct soap *pSOAP,
    struct _CISCOAIM__UpdatePackage *pIn,
    struct _CISCOAIM__UpdatePackageResponse *pOut)
```

#### Structs

```c
struct _CISCOAIM__UpdatePackage
{
    char *ADIURL;
    char *PackageName;
    int   MetaDataOnly;
    int   DoAsync;
    struct CISCOAIM__EndpointList *AsyncTargets; /* opt element CISCOAIM:EndpointList */
};

struct _CISCOAIM__UpdatePackageResponse
{
    char *UpdateResult;
}
```
struct CISCOAIM__EndpointList
{
  int __sizeServerIp; /* sequence of elements <ServerIp> */
  char **ServerIp;    /* optional element of type xsd:string */
  int __sizePort;     /* sequence of elements <Port> */
  int *Port;          /* optional element of type xsd:int */
  int __sizePath;     /* sequence of elements <Path> */
  char **Path;        /* optional element of type xsd:string */
};

Parameter Description

REQUEST:
ADIURL: The location of the package XML. ie. 'ftp://Im:Pass@my.box.info/here/ADI.XML'
PackageName: Name of the package.
MetaDataOnly: [0-NO|1-YES] Setting it to 1 enables the user to ingest only XML metadata.

Note Currently VDS-TV does not support ingesting only XML metadata. If the user wants to update only the metadata then the ADI XML should be modified as defined by ADI specification.

DoAsync: [0-NO|1-YES] By setting it to 1, AIM Immediately returns from the IngestPackage call and does not wait until ingestion is complete. This setting is useful for clients that are single threaded and processes one package at a time. AIM does a callback to the application using imported AIMPackageNotification interface. If AIMPackageNotification interface is not available then GetPackageStatus is used to obtain the status of package ingestion.
AsyncTargets: List of notification targets for an async action.

RESPONSE:
UpdateResult: If update is successful then SUCCESS is returned else an error string is returned. For detailed information on SOAP faults, refer SOAP Faults, page 6-18

GetPackageStatus

This function is used to retrieve the status of ingest of a specified package.

Signature

SOAP_FMAC5 int SOAP_FMAC6 __CISCOAIM__GetPackageStatus(
  struct soap *pSOAP,
  struct _CISCOAIM__GetPackageStatus *pIn,
  struct _CISCOAIM__GetPackageStatusResponse *pOut)

Structs

struct _CISCOAIM__GetPackageStatus
{
  char *PackageName;
};

struct _CISCOAIM__GetPackageStatusResponse
Parameter Description

REQUEST:
PackageName: Name of the package

RESPONSE:
StatusResult: Returns the status of the package in the system

• 0 Complete - Package ingest is successful and the package exists in the system
• -1 Failed - Package ingest failed. Error occurred, with more information
• 1 Incomplete - Package ingest is incomplete
• 2 Pending callback - Package ingest is successful and waiting for available resources for callback

GetAllPackages

This function is used to obtain a list of all packages in the system.

Signature

SOAP_FMAC5 int SOAP_FMAC6 __CISCOAIM__GetAllPackages(
    struct soap *pSOAP,
    char *pLocation,
    struct __CISCOAIM__GetAllPackagesResponse *pOut)

Structs

struct __CISCOAIM__GetAllPackagesResponse
{
    struct CISCOAIM__List *PackageList;
};

Parameter Description

REQUEST:
No parameters

RESPONSE:
PackageList: A list of all packages and the structure for package list is defined below:

struct CISCOAIM__List
{
    int __sizestring;
    char **string;    // Null pointer, when list is size 0
};
**GetAvailableDiskSpace**

This function is used to obtain the total space and available disk space in the vault group.

**Signature**

```c
SOAP_FM5 int SOAP_FM6 __CISCOAIM__GetAvailableDiskSpace(
    struct soap *pSOAP,
    struct __CISCOAIM__GetGetAvailableDiskSpace *pIn, //optional
    struct __CISCOAIM__GetAvailableDiskSpaceResponse *pOut);
```

**Structs**

```c
struct __CISCOAIM__GetGetAvailableDiskSpace
{
    char *ID; //optional
};
```

```c
struct __CISCOAIM__GetAvailableDiskSpaceResponse
{
    char *TotalSpace;
    char *AvailableSpace;
};
```

**Parameter Description**

**REQUEST:**

ID: Provider ID mapped to the vault group, Optional Field

**Note**

Used only in OCN content steering feature to obtain the specific disk space for each “Provider”.

**RESPONSE:**

TotalSpace: The total number of bytes that can be stored in the vault group.

AvailableSpace: The number of available bytes in the vault group.

**SOAP Imports**

This interface is OPTIONAL. If the async flag is set, callbacks are sent to the client and the AIMPackageNotification SOAP call is invoked. The clients sends a notification result as a 32 bit signed integer. This notification result is mainly used for debugging purposes.
AIMPackageNotification

Signature

```c
SOAP_FMAC5 int SOAP_FMAC6 soap_call___IMPORT__AIMPackageNotification(
    struct soap *soap,
    const char  *soap_endpoint,
    const char  *soap_action,
    struct _IMPORT__AIMPackageNotification *IMPORT__AIMPckgN,
    struct _IMPORT__AIMPackageNotificationResponse *IMPORT__AIMPckgNResponse
);
```

Structs

```c
struct _IMPORT__AIMPackageNotification
{
    char *ADIURL;
    char *PackageName;
    char *Result
};
struct _IMPORT__AIMPackageNotificationResponse
{
    int NotificationResult;
};
```

Parameter Description

REQUEST:
ADIURL: The location of the package XML. ie. 'ftp://Im:Pass@my.box.info/here/ADI.XML'
PackageName: Name of the package
Result: Same format as the other results(i.e., IngestPackageResult)

RESPONSE:
NotificationResult: AIM accepts any valid 32bit signed integer.

SOAP Faults

This section describes the SOAP fault structures.

Structs

```c
/* SOAP-ENV:Detail */
struct SOAP_ENV__Detail
{
    int __type;/* any type of element <fault> (defined below) */
    void *fault;/* transient */
    char *__any;
};
```
Parameter Description:

SOAP_ENV__Fault::faultstring - ""
SOAP_ENV__Fault::faultactor - unused
SOAP_ENV__Fault::detail - SOAP_ENV__Detail
SOAP_ENV__Detail::__type - SOAP_TYPE_string
SOAP_ENV__Detail::fault - Specific message about the error
SOAP_ENV__Detail::__any - unused
SOAP_ENV__Fault::SOAP_ENV__Code - unused
SOAP_ENV__Fault::SOAP_ENV__Reason - unused
SOAP_ENV__Fault::SOAP_ENV__Node - unused
SOAP_ENV__Fault::SOAP_ENV__Role - unused
SOAP_ENV__Fault::SOAP_ENV__Detail - unused

WSDL Definitions

CiscoAIM.wsdl

```xml
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:tns="http://cisco.aim.ns/CiscoAIM"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/
targetNamespace="http://cisco.aim.ns/CiscoAIM"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
<wsdl:types>
<s:schema elementFormDefault="qualified"
targetNamespace="http://cisco.aim.ns/CiscoAIM">
<s:element name="IngestPackage">
    <s:complexType>
        <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="ADIURL" type="s:string"/>
        </s:sequence>
    </s:complexType>
</s:element>
</s:schema>
</wsdl:types>
</wsdl:definitions>
```
<s:element minOccurs="0" maxOccurs="1" name="PackageName" type="s:string"/>
<s:element minOccurs="0" maxOccurs="1" name="MetaDataOnly" type="s:int"/>
<s:element minOccurs="0" maxOccurs="1" name="DoAsync" type="s:int"/>
<s:element minOccurs="0" maxOccurs="1" name="AsyncTargets" type="tns:EndpointList"/>
</s:sequence>
</s:complexType>
</s:element>
<s:element name="IngestPackageResponse">
<s:complexType>
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="IngestResult" type="s:string"/>
</s:sequence>
</s:complexType>
</s:element>
<s:element name="DeletePackage">
<s:complexType>
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="PackageName" type="s:string"/>
<s:element minOccurs="0" maxOccurs="1" name="MetaDataOnly" type="s:int"/>
</s:sequence>
</s:complexType>
</s:element>
<s:element name="DeletePackageResponse">
<s:complexType>
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="DeleteResult" type="s:string"/>
</s:sequence>
</s:complexType>
</s:element>
<s:element name="UpdatePackage">
<s:complexType>
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="ADIURL" type="s:string"/>
<s:element minOccurs="0" maxOccurs="1" name="PackageName" type="s:string"/>
<s:element minOccurs="0" maxOccurs="1" name="MetaDataOnly" type="s:int"/>
<s:element minOccurs="0" maxOccurs="1" name="DoAsync" type="s:int"/>
<s:element minOccurs="0" maxOccurs="1" name="AsyncTargets" type="tns:EndpointList"/>
</s:sequence>
</s:complexType>
</s:element>
<s:element name="UpdatePackageResponse">
<s:complexType>
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="UpdateResult" type="s:string"/>
</s:sequence>
</s:complexType>
</s:element>
<s:element name="GetPackageStatus">
<s:complexType>
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="PackageName" type="s:string"/>
</s:sequence>
</s:complexType>
</s:element>
<s:element name="GetPackageStatusResponse">
<s:complexType>
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="StatusResult" type="s:string"/>
</s:sequence>
</s:complexType>
</s:element>
<s:element name="GetAllPackages">
<wsdl:message name="GetAllPackagesSoapIn">
  <wsdl:part name="parameters" element="tns:GetAllPackages"/>
</wsdl:message>

<wsdl:message name="GetAllPackagesSoapOut">
  <wsdl:part name="parameters" element="tns:GetAllPackagesResponse"/>
</wsdl:message>

<wsdl:message name="GetAvailableDiskSpaceSoapIn">
  <wsdl:part name="parameters" element="tns:GetAvailableDiskSpace"/>
</wsdl:message>

<wsdl:message name="GetAvailableDiskSpaceSoapOut">
  <wsdl:part name="parameters" element="tns:GetAvailableDiskSpaceResponse"/>
</wsdl:message>

<wsdl:portType name="CiscoAIMSoap11">
  <wsdl:operation name="IngestPackage">
    <wsdl:input message="tns:IngestPackageSoapIn"/>
    <wsdl:output message="tns:IngestPackageSoapOut"/>
  </wsdl:operation>
  <wsdl:operation name="UpdatePackage">
    <wsdl:input message="tns:UpdatePackageSoapIn"/>
    <wsdl:output message="tns:UpdatePackageSoapOut"/>
  </wsdl:operation>
  <wsdl:operation name="DeletePackage">
    <wsdl:input message="tns:DeletePackageSoapIn"/>
    <wsdl:output message="tns:DeletePackageSoapOut"/>
  </wsdl:operation>
  <wsdl:operation name="GetPackageStatus">
    <wsdl:input message="tns:GetPackageStatusSoapIn"/>
    <wsdl:output message="tns:GetPackageStatusSoapOut"/>
  </wsdl:operation>
  <wsdl:operation name="GetAllPackages">
    <wsdl:input message="tns:GetAllPackagesSoapIn"/>
    <wsdl:output message="tns:GetAllPackagesSoapOut"/>
  </wsdl:operation>
  <wsdl:operation name="GetAvailableDiskSpace">
    <wsdl:input message="tns:GetAvailableDiskSpaceSoapIn"/>
    <wsdl:output message="tns:GetAvailableDiskSpaceSoapOut"/>
  </wsdl:operation>
</wsdl:portType>

<wsdl:binding name="CiscoAIMSoap11" type="tns:CiscoAIMSoap11">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http"/>
  <soap:operation name="IngestPackage" style="document"/>
  <soap:input message="tns:IngestPackageSoapIn"/>
  <soap:output message="tns:IngestPackageSoapOut"/>
  <soap:operation name="UpdatePackage" style="document"/>
  <soap:input message="tns:UpdatePackageSoapIn"/>
  <soap:output message="tns:UpdatePackageSoapOut"/>
  <soap:operation name="DeletePackage" style="document"/>
  <soap:input message="tns:DeletePackageSoapIn"/>
  <soap:output message="tns:DeletePackageSoapOut"/>
  <soap:operation name="GetPackageStatus" style="document"/>
  <soap:input message="tns:GetPackageStatusSoapIn"/>
  <soap:output message="tns:GetPackageStatusSoapOut"/>
  <soap:operation name="GetAllPackages" style="document"/>
  <soap:input message="tns:GetAllPackagesSoapIn"/>
  <soap:output message="tns:GetAllPackagesSoapOut"/>
  <soap:operation name="GetAvailableDiskSpace" style="document"/>
  <soap:input message="tns:GetAvailableDiskSpaceSoapIn"/>
  <soap:output message="tns:GetAvailableDiskSpaceSoapOut"/>
</wsdl:binding>
<wsdl:output>
  <soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>

<wsdl:operation name="GetPackageStatus">
  <soap:operation soapAction="CISCOAIM#GetPackageStatus" style="document"/>
  <wsdl:input>
    <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
    <soap:body use="literal"/>
  </wsdl:output>
</wsdl:operation>

<wsdl:operation name="GetAllPackages">
  <soap:operation soapAction="CISCOAIM#GetAllPackages" style="document"/>
  <wsdl:input>
    <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
    <soap:body use="literal"/>
  </wsdl:output>
</wsdl:operation>

<wsdl:operation name="GetAvailableDiskSpace">
  <soap:operation soapAction="CISCOAIM#GetAvailableDiskSpace" style="document"/>
  <wsdl:input>
    <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
    <soap:body use="literal"/>
  </wsdl:output>
</wsdl:operation>

<wsdl:operation name="IngestPackage">
  <soap:operation soapAction="CISCOAIM#IngestPackage" style="document"/>
  <wsdl:input>
    <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
    <soap:body use="literal"/>
  </wsdl:output>
</wsdl:operation>

<wsdl:operation name="UpdatePackage">
  <soap:operation soapAction="CISCOAIM#UpdatePackage" style="document"/>
  <wsdl:input>
    <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
    <soap:body use="literal"/>
  </wsdl:output>
</wsdl:operation>

<wsdl:operation name="DeletePackage">
  <soap:operation soapAction="CISCOAIM#DeletePackage" style="document"/>
  <wsdl:input>
    <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
    <soap:body use="literal"/>
  </wsdl:output>
</wsdl:operation>

<wsdl:operation name="GetPackageStatus">
  <soap:operation soapAction="CISCOAIM#GetPackageStatus" style="document"/>
  <wsdl:input>
    <soap:body use="literal"/>
  </wsdl:input>
</wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>

<wsdl:operation name="GetAllPackages">
<soap:operation soapAction="CISCOAIM#GetAllPackages" style="document"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>

<wsdl:operation name="GetAvailableDiskSpace">
<soap:operation soapAction="CISCOAIM#GetAvailableDiskSpace" style="document"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>

</wsdl:binding>
<wsdl:service name="CiscoAIM">
<wsdl:port name="CiscoAIMSoap11" binding="tns:CiscoAIMSoap11">
<soap:address location="http://localhost:8792/CiscoAIM"/>
</wsdl:port>
<wsdl:port name="CiscoAIMSoap12" binding="tns:CiscoAIMSoap11">
<soap:address location="http://localhost:8793/CiscoAIM"/>
</wsdl:port>
</wsdl:service>
</wsdl:definitions>

'CiscoAIMNotification.wsdl'

<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
xmlns:soap="http://schemas.xmlsoap.org/wsd1/soap/
xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/
xmlns:tns="http://cisco.aim.ns" 
xmlns:s="http://www.w3.org/2001/XMLSchema" 
xmlns:soap12="http://schemas.xmlsoap.org/wsd1/soap12/
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/
tARGETNamespace="http://cisco.aim.ns" 
xmlns:wsdl="http://schemas.xmlsoap.org/wsd1/">
<wsdl:types>
<s:complexType>
<s:sequence>
<s:element minOccurs="1" maxOccurs="1" name="ADIURL" type="s:string"/>
<s:element minOccurs="1" maxOccurs="1" name="PackageName" type="s:string"/>
<s:element minOccurs="1" maxOccurs="1" name="Result" type="s:string"/>
</s:sequence>
</s:complexType>
<wsdl:element name="AIMPackageNotificationResponse">
<s:complexType>
GET Publishing Queue Management List

The Publishing Queue Management list request-response message returns a list of all packages in a queue to be published to VOD market sites.
GET Publishing Queue Management List

Request Type
HTTP method supported: GET

Request Format
curl -H "Accept: application/xml" -H "Content-Type: application/xml" -X GET
http://<cdsm_host>/api/services/monitor/system/id/publishingqueuemanagement/

Response
If the request succeeds, a list element is returned in the XML body response containing a list of all packages in a queue to be published to VOD market sites.

Table 6-4 describes the XML body elements, sub elements, and attributes returned in the list element.

Table 6-4  PublishingQueueManagement List Element

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes/Sub Elements</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>PublishingQueueManagemen</td>
<td>—</td>
<td>List Element</td>
<td>Element that represents a list of packages to be published.</td>
</tr>
<tr>
<td>PackageQueue</td>
<td>Sub Element</td>
<td></td>
<td>Element that represents a package in a publishing queue.</td>
</tr>
<tr>
<td>Priority</td>
<td>xs:string</td>
<td></td>
<td>Priority of the package in the queue.</td>
</tr>
<tr>
<td>PackageName</td>
<td>xs:string</td>
<td></td>
<td>Name of the package</td>
</tr>
<tr>
<td>AssetName</td>
<td>xs:string</td>
<td></td>
<td>Name of the asset</td>
</tr>
<tr>
<td>Market</td>
<td>xs:string</td>
<td></td>
<td>Domain name of the VOD Market Site</td>
</tr>
<tr>
<td>CurrentStatus</td>
<td>xs:string</td>
<td></td>
<td>Current publishing status of the package in the queue.</td>
</tr>
<tr>
<td>TimeInserted</td>
<td>xs:string</td>
<td></td>
<td>Date and time the package was moved to the publishing queue.</td>
</tr>
<tr>
<td>UserAction</td>
<td>xs:string</td>
<td></td>
<td>The action set by the user on a package in a queue.</td>
</tr>
</tbody>
</table>

Response Example
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<PublishingQueueManagement>
</PublishingQueueManagement>
POST Packages in Publishing Queue

Following are the request components:

**Request Type**
HTTP method supported: POST.

**Request Format**
http://<cdsm_host>/api/services/monitor/system/id/publishingqueueManagement/

**Request Body XML**
Following is the XML body for the POST request to update/delete packages in a publishing queue.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<PublishingQueueManagement>
  <PackageQueue Priority="193" PackageName="Test_1236" AssetName="Package_1236"
  Market="MKT1" CurrentStatus="Hold" TimeInserted="04-26-2016 04:08:53"
  UserAction="Resume"/>
  <PackageQueue Priority="193" PackageName="Test_1237" AssetName="Package_1237"
  Market="MKT1" CurrentStatus="Waiting" TimeInserted="04-26-2016 04:08:53"
  UserAction="Hold"/>
  <PackageQueue Priority="193" PackageName="Test_1238" AssetName="Package_1238"
  Market="MKT1" CurrentStatus="Waiting" TimeInserted="04-26-2016 04:08:53"
  UserAction="Delete"/>
  <PackageQueue Priority="193" PackageName="Test_1239" AssetName="Package_1239"
  Market="MKT1" CurrentStatus="Waiting" TimeInserted="04-26-2016 04:08:53"
  UserAction="Hold"/>
  <PackageQueue Priority="193" PackageName="Test_1240" AssetName="Package_1240"
  Market="MKT1" CurrentStatus="Hold" TimeInserted="04-26-2016 04:08:52"
  UserAction="Delete"/>
  <PackageQueue Priority="193" PackageName="Test_1337" AssetName="Package1337" Market="MKT1"
  CurrentStatus="On Hold" TimeInserted="04-26-2016 02:41:47"
  UserAction="Hold"/>
</PublishingQueueManagement>
```
Replication Group APIs

This chapter describes the format and content of the following Replication Group APIs.

- Add Replication Group, page 7-1
- Delete Replication Group, page 7-3
- Assign members to Replication Group, page 7-4
- Un-assigning members from Replication Group, page 7-5
- Replication Group Configuration List, page 7-6
- Unassigned Replication Servers List, page 7-7

All HTTP request messages follow the format given in the “HTTP Headers” section on page 1-3.

Add Replication Group

Following are the request components:

Request Type
HTTP method supported: POST.

Request Format

curl --form upload=@/root/replicateGroup.xml --form press=OK
'Content-Type:text/xml;charset=UTF-8'; 'enctype=multipart/form-data'
http://<VIM/CDSM_HostIP>/api/services/configure/array/id/addReplicationGroup

Request Body XML

The addReplicationGroup POST REST API is used to upload one or more replication groups to the system.

Following is the XML body for the POST request to add the replication group configuration settings:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationGroupConfigList
 xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
  <ReplicationGroupConfig GroupName='PCG1' IPAddress='10.178.78.81' GroupID='2425'
     SubnetMask='255.255.255.0' Type='Cache'>
      <ReplicationGroupMembers>
        <Server IPAddress='10.178.78.81' ServerID='81' GroupID='2425'/>
        <Server IPAddress='10.178.78.82' ServerID='82' GroupID='2425'/>
      </ReplicationGroupMembers>
  </ReplicationGroupConfig>
</ReplicationGroupConfigList>
```
Table 7-2 describes the XML body elements, sub elements, and attributes of the request.

Response
If the request fails, the status message indicates that an error occurred while adding the replication group and one of the following error codes is returned:

- 1623
- 1624
- 1628
- 1629
- 1630
- 1631
- 1632
- 1639

For additional information on replication error codes, see Table 7-1.

If the request succeeds, the status message indicates that the “Replication group setup configured successfully” and the following response is returned:

```xml
<status>Replication group setup configured successfully</status>
```

Replication Error Codes

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1623</td>
<td>Replicate Group names must be unique among all Replicate Groups name.</td>
</tr>
<tr>
<td>1624</td>
<td>Replicate Group IP must be unique among all Replicate Groups name.</td>
</tr>
<tr>
<td>1628</td>
<td>Replicate member IP not configured.</td>
</tr>
<tr>
<td>1629</td>
<td>Invalid Server ID and Group ID of Replicate Member IP</td>
</tr>
<tr>
<td>1630</td>
<td>Invalid group type of Replicate Member IP</td>
</tr>
<tr>
<td>1631</td>
<td>Invalid Replicate Group Name.</td>
</tr>
<tr>
<td>1632</td>
<td>Error: Failed to import Replication group setup configurations from XML</td>
</tr>
<tr>
<td>1639</td>
<td>Please un-assign group members prior to deleting replicate group.</td>
</tr>
</tbody>
</table>
Delete Replication Group

Following are the request components:

**Request Type**
HTTP method supported: POST.

**Request Format**
curl --form upload=@/root/replicateGroup.xml --form press=OK
'Content-Type:text/xml;charset=UTF-8' 'enctype=multipart/form-data'
http://<VIM/CDSM_HostIP>/api/services/configure/array/id/deleteReplicationGroup

**Request Body XML**
The deleteReplicationGroup POST REST API is used to delete one or more replication groups from the system.

Following is the XML body for the POST request to delete the replication group configuration settings:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationGroupConfigList
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
  <ReplicationGroupConfig GroupName='PCG1' IPAddress='10.197.105.103' SubnetMask='255.255.255.0' Type='Cache'>
    <ReplicationGroupMembers>
      <Server IPAddress='10.178.78.81' ServerID='81' GroupID='2425'/>
      <Server IPAddress='10.178.78.82' ServerID='82' GroupID='2425'/>
    </ReplicationGroupMembers>
  </ReplicationGroupConfig>
</ReplicationGroupConfigList>
```

Table 7-2 describes the XML body elements, sub elements, and attributes of the request.

**Response**
If the request fails, the status message indicates that an error occurred while request to delete replication group and one of the following error codes is returned:

- 1631
- 1639

For additional information on replication error codes, see Table 7-1

If the request succeeds, the status message indicates that the “Replication group(s) deleted successfully” and the following response is returned:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<status>Replication group(s) deleted successfully</status>
```
Assign members to Replication Group

Following are the request components:

**Request Type**
HTTP method supported: POST.

**Request Format**

```
curl --form upload=@/root/replicateGroup.xml --form press=OK
'Content-Type:text/xml;charset=UTF-8' 'enctype=multipart/form-data'
http://<VVIM/CDSM_HostIP>/api/services/configure/array/id/assignReplicationGroup
```

**Request Body XML**
The assignReplicationGroup POST REST API is used to assign one or more servers to a replication group.

Following is the XML body for the POST request to assign one or more servers to a replication group:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationGroupConfigList
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
  <ReplicationGroupConfig GroupName='PVG1' IPAddress='10.197.105.103'
  SubnetMask='255.255.255.0' Type='Vault'>
    <ReplicationGroupMembers>
      <Server IPAddress='10.178.78.81' ServerID='81' GroupID='2425'/>
      <Server IPAddress='10.178.78.82' ServerID='82' GroupID='2425'/>
      <Server IPAddress='10.178.78.83' ServerID='83' GroupID='2425'/>
    </ReplicationGroupMembers>
  </ReplicationGroupConfig>
</ReplicationGroupConfigList>
```

**Table 7-2** describes the XML body elements, sub elements, and attributes of the request.

**Response**

If the request fails, the status message indicates that an error occurred while assigning members to the replication group and one of the following error codes is returned:

- 1628
- 1629
- 1630
- 1631
- 1632

For additional information on replication error codes, see **Table 7-1**.

If the request succeeds, the status message indicates that the “Replication group setup configured successfully” and the following response is returned:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<status> Replication group setup configured successfully</status>
```
Un-assigning members from Replication Group

Following are the request components:

**Request Type**
HTTP method supported: POST.

**Request Format**
curl --form upload=@/root/replicateGroup.xml --form press=OK
'Content-Type:text/xml;charset=UTF-8' 'enctype=multipart/form-data'
http://<VVIM/CDSM_HostIP>/api/services/configure/array/id/unassignReplicationGroup

**Request Body XML**
The unassignReplicationGroup POST REST API is used to un-assign one or more servers from a replication group.

Following is the XML body for the POST request to un-assign members from a replication group:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationGroupConfigList
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
  <ReplicationGroupConfig GroupName='' IPAddress='' SubnetMask='' Type='Streamer'>
    <ReplicationGroupMembers>
      <Server IPAddress='10.178.78.81' ServerID='81' GroupID='2425'/>
      <Server IPAddress='10.178.78.82' ServerID='82' GroupID='2425'/>
      <Server IPAddress='10.178.78.83' ServerID='83' GroupID='2425'/>
    </ReplicationGroupMembers>
  </ReplicationGroupConfig>
</ReplicationGroupConfigList>
```

Table 7-2 describes the XML body elements, sub elements, and attributes of the request.

**Response**
If the request fails, the status message indicates that an error occurred while un-assigning member from the replication group and one of the following error codes is returned:
- 1628
- 1629
- 1630
- 1632

For additional information on replication error codes, see Table 7-1.

If the request succeeds, the status message indicates that the “Replication group setup configured successfully” and the following response is returned:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<status> Replication group setup configured successfully</status>
```
Replication Group Configuration List

The replication group configuration list request-response message provides a list of all replication groups configured in the system. It also retrieves a list of replication groups based on specific filter criteria.

**Request Type**
HTTP method supported: GET.

**Request Examples for different filter criteria**

Filter By: getall

curl "http://<cdsm_host>/api/services/configure/array/id/replicationGroupSetup"

Filter By: Group Name

curl "http://10.197.101.146/api/services/configure/array/id/replicationGroupSetup/groupName/PSG1"

Filter By: Group Type

curl "http://10.197.101.146/api/services/configure/array/id/replicationGroupSetup/groupType/Cache"

Filter By: Server ID

curl "http://10.197.101.146/api/services/configure/array/id/replicationGroupSetup/serverID/81"

**Response**

If the request fails, the following response is returned:

```xml
<?xml version="1.0" encoding="UTF-8"?><ReplicationGroupConfigList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><error>No Records Found</error></ReplicationGroupConfigList>
```

If the request succeeds, the list element is returned in the XML body response.

**Table 7-2** describes the XML body elements, sub elements, and attributes returned in the List element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes/Sub Elements</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReplicationGroupConfigList</td>
<td></td>
<td>List Element</td>
<td>Element that represents a list of replication groups configured in the system.</td>
</tr>
<tr>
<td></td>
<td>ReplicationGroupConfig</td>
<td>Subelement</td>
<td>Element that represents a replication group.</td>
</tr>
<tr>
<td>GroupConfig</td>
<td></td>
<td>ReplicationGroup ConfigList Subelement</td>
<td>Element that represents a replication group.</td>
</tr>
<tr>
<td>GroupName</td>
<td>xs:string</td>
<td></td>
<td>Replication Group Name.</td>
</tr>
<tr>
<td>IPAddress</td>
<td>xs:string</td>
<td></td>
<td>IP address of the replication group.</td>
</tr>
<tr>
<td>SubnetMask</td>
<td>xs:string</td>
<td></td>
<td>Subnet mask of the replication group.</td>
</tr>
</tbody>
</table>
Chapter 7 Replication Group APIs

Unassigned Replication Servers List

The unassigned replication servers list request-response message provides a list of all servers unassigned from a replication group.

**Request Type**
HTTP method supported: GET.

**Request Format**
curl "http://<cdsm_host>/api/services/configure/array/id/replicationGroupSetup"

**Request Examples for different filter criteria**
Filter By: getall
curl "http://<cdsm_host>/api/services/configure/array/id/getUnassignedReplicationServers"
Filter By: Group Type

curl
"http://<cdsm_host>/api/services/configure/array/id/getUnassignedReplicationServers/groupType/Cache"

**Response**

If the request fails, the following response is returned:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationGroupConfigList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><error>No Records Found</error></ReplicationGroupConfigList>
```

If the request succeeds, the list element is returned in the XML body response.

Table 7-2 describes the XML body elements, sub elements, and attributes returned in the List element.

**Response Example**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationGroupConfigList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
  <ReplicationGroupConfig GroupName='' IPAddress='' SubnetMask='' Type='Cache'>
    <ReplicationGroupMembers>
      <Server IPAddress='10.178.78.81' ServerID='81' GroupID='2425'/>
      <Server IPAddress='10.178.78.82' ServerID='82' GroupID='2425'/>
    </ReplicationGroupMembers>
  </ReplicationGroupConfig>
  <ReplicationGroupConfig GroupName='' IPAddress='' SubnetMask='' Type='Vault'>
    <ReplicationGroupMembers>
      <Server IPAddress='10.178.78.91' ServerID='91' GroupID='2425'/>
      <Server IPAddress='10.178.78.92' ServerID='92' GroupID='2425'/>
    </ReplicationGroupMembers>
  </ReplicationGroupConfig>
</ReplicationGroupConfigList>
```
This chapter describes the format and content of APIs necessary to implement an interface between Personal Video Recorder (PVR) Scheduler and PVR Server. The PVR Scheduler Interface uses XML over HTTP to exchange control messages between PVR Scheduler and PVR Server.

This chapter describes the format and content of the following PVR Scheduler APIs.

- Provisioning a Home, page 8-1
- Adding an STB, page 8-4
- Updating an STB, page 8-5
- Deprovisioning the Home, page 8-7
- Removing an STB, page 8-8
- Get Home Profile Details, page 8-9
- Get STB Profile Details, page 8-10
- Scheduling a Recording, page 8-11
- Deleting a Recording, page 8-14
- Notifying the Recording Completion, page 8-17

Provisioning a Home

Following are the request components:

**Request Direction**
PVR Scheduler to PVR Server

**Description**
This API directs the PVR server to provision a new Home ID with an allocation size and provision a list of set-tops associated to the Home ID. If set-tops provisioning fails then a list of failed set-tops is returned as part of the response.
Request Type
HTTP method supported: POST.

Request Example
POST /RSDVR?action=ProvisionHome HTTP/1.1
Content-Type: text/xml
Connection: Close
cseq: 2
Content-Length: 257

<?xml version="1.0" encoding="utf-8"?>
<ProvisionHome
HomeID = "987654321234567"
><STBIDList>
<STBID
MACAddress = "9B8A84EF0101"
AllocationSize = "0"
ServiceGroup = "1234"
/>
<STBID
MACAddress = "9B8A84EF0102"
AllocationSize = "0"
ServiceGroup = "1234"
/>
</STBIDList>
</ProvisionHome>

Table 8-1 describes the XML body elements, sub elements, and attributes of the ProvisionHome request XML.

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes/Sub Elements</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProvisionHome</td>
<td>—</td>
<td>Element</td>
<td>Element that represents a Home ID to be provisioned.</td>
</tr>
<tr>
<td>HomeID</td>
<td>xs:string</td>
<td>Subelement</td>
<td>Unique ID to identify the subscriber’s account.</td>
</tr>
<tr>
<td>STBIDList</td>
<td>ProvisioHome</td>
<td>Subelement</td>
<td>Element that represents a list of set-tops</td>
</tr>
<tr>
<td>STBID</td>
<td>STBIDList</td>
<td>Subelement</td>
<td>Element that represents a set-top.</td>
</tr>
<tr>
<td>STBID</td>
<td>MACAddress</td>
<td>Subelement</td>
<td>Element that represents a set-top.</td>
</tr>
<tr>
<td>STBID</td>
<td>AllocationSize</td>
<td>Subelement</td>
<td>Unique ID to identify the subscriber’s set-top.</td>
</tr>
<tr>
<td>STBID</td>
<td>ServiceGroup</td>
<td>Subelement</td>
<td>Service group assigned to set-top.</td>
</tr>
</tbody>
</table>

Response
One of the following HTTP status codes is returned:
- 200 Ok—Request was successful.
400 Bad Request—Request parameters were incomplete or invalid.

If the provisioning of the set-tops fails, then a list of failed set-tops along with reason code and description is returned in the XML response. Table 8-2 describes the XML body elements, sub elements, and attributes returned in the response element.

- 430
- 431
- 432
- 434
- 443
- 500
- 531

For additional information on PVR Server reason codes, see Table 8-3

<table>
<thead>
<tr>
<th>Table 8-2</th>
<th>ProvisionHomeResponse Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
<td>Attributes/SubElements</td>
</tr>
<tr>
<td>ProvisionHomeResponse</td>
<td>—</td>
</tr>
<tr>
<td>FailedSTBIDList</td>
<td>Subelement</td>
</tr>
<tr>
<td>ProvisionHomeResponse</td>
<td>ProvisionHomeResponse Subelement</td>
</tr>
<tr>
<td>FailedSTBID</td>
<td>Subelement</td>
</tr>
<tr>
<td>FailedSTBID</td>
<td>FailedSTBIDList subelement</td>
</tr>
<tr>
<td>MACAddress</td>
<td>xs:string</td>
</tr>
<tr>
<td>ReasonCode</td>
<td>xs:integer</td>
</tr>
<tr>
<td>ReasonDescription</td>
<td>xs:string</td>
</tr>
</tbody>
</table>

**PVR Server Reason Codes**

<table>
<thead>
<tr>
<th>Table 8-3</th>
<th>PVR Server Reason Codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error Code</td>
<td>Description</td>
</tr>
<tr>
<td>430</td>
<td>Argument validation error.</td>
</tr>
<tr>
<td>431</td>
<td>Home ID already provisioned.</td>
</tr>
</tbody>
</table>
Adding an STB

Following are the request components:

**Request Direction**
PVR Scheduler to PVR Server

**Description**
This API directs the PVR server to provision a new set-top to an existing Home ID.

**Request Type**
HTTP method supported: POST.

**Request Example**

```
POST /RSDVR?action=AddSTB HTTP/1.1
Content-Type: text/xml
Connection: Close
cseq: 2
Content-Length: 257

<?xml version="1.0" encoding="utf-8"?>
<AddSTB
HomeID = "987654321234567"
MACAddress = "9B8A84EF0101"
AllocationSize = "0"
ServiceGroup = "1234"
/>```

### Error Code Description

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>432</td>
<td>Home ID does not exist.</td>
</tr>
<tr>
<td>435</td>
<td>Recording aborted.</td>
</tr>
<tr>
<td>434</td>
<td>Service group not found.</td>
</tr>
<tr>
<td>436</td>
<td>Recording already scheduled.</td>
</tr>
<tr>
<td>437</td>
<td>Request for past recording.</td>
</tr>
<tr>
<td>439</td>
<td>Server not found.</td>
</tr>
<tr>
<td>440</td>
<td>Program not available.</td>
</tr>
<tr>
<td>441</td>
<td>Exceeded user disc space quota.</td>
</tr>
<tr>
<td>442</td>
<td>Recording error.</td>
</tr>
<tr>
<td>443</td>
<td>STB already provisioned.</td>
</tr>
<tr>
<td>444</td>
<td>STB does not exist.</td>
</tr>
<tr>
<td>500</td>
<td>Internal server error.</td>
</tr>
<tr>
<td>531</td>
<td>Unable to assign server.</td>
</tr>
</tbody>
</table>

**Error Code**

432 Home ID does not exist.
435 Recording aborted.
434 Service group not found.
436 Recording already scheduled.
437 Request for past recording.
439 Server not found.
440 Program not available.
441 Exceeded user disc space quota.
442 Recording error.
443 STB already provisioned.
444 STB does not exist.
500 Internal server error.
531 Unable to assign server.
Table 8-4 describes the XML body elements, sub elements, and attributes of the AddSTB request XML.

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes/Sub Elements</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddSTB</td>
<td>---</td>
<td>Element</td>
<td>Element that represents a set-top.</td>
</tr>
<tr>
<td></td>
<td>HomeID</td>
<td>xs:string</td>
<td>Unique ID to identify the subscriber’s account.</td>
</tr>
<tr>
<td></td>
<td>MACAddress</td>
<td>xs:string</td>
<td>Unique ID to identify the subscriber’s set-top.</td>
</tr>
<tr>
<td></td>
<td>AllocationSize</td>
<td>xs:string</td>
<td>Number of blocks allocated to set-top.</td>
</tr>
<tr>
<td></td>
<td>ServiceGroup</td>
<td>xs:string</td>
<td>Service group assigned to set-top.</td>
</tr>
</tbody>
</table>

Response
One of the following HTTP status codes is returned:
- 200 Ok—Request was successful.
- 400 Bad Request—Request parameters were incomplete or invalid.

No XML body is returned in the response.
If the request fails, the status message indicates that an error occurred and one of the following reason codes is returned:
- 430
- 432
- 434
- 443
- 500

For additional information on PVR Server reason codes, see Table 8-3

**Updating an STB**

Following are the request components:

**Request Direction**
PVR Scheduler to PVR Server

**Description**
This API directs the PVR server to update an existing set-top.
Request Type
HTTP method supported: POST.

Request Body XML
POST /RSDVR?action=UpdateSTB HTTP/1.1
Content-Type: text/xml
Connection: Close
cseq: 2
Content-Length: 257
<?xml version="1.0" encoding="utf-8"?>
<UpdateSTB
HomeID = "987654321234567"
MACAddress = "9B8A84EF0101"
AllocationSize = "0"
/>

Table 8-5 describes the XML body elements, sub elements, and attributes of the UpdateSTB request XML.

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes/Sub Elements</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>UpdateSTB</td>
<td>—</td>
<td>Element</td>
<td>Element that represents a set-top.</td>
</tr>
<tr>
<td>HomeID</td>
<td>xs:string</td>
<td></td>
<td>Unique ID to identify the subscriber’s account.</td>
</tr>
<tr>
<td>MACAddress</td>
<td>xs:string</td>
<td></td>
<td>Unique ID to identify the subscriber’s set-top.</td>
</tr>
<tr>
<td>AllocationSize</td>
<td>xs:string</td>
<td></td>
<td>Number of blocks allocated to set-top.</td>
</tr>
</tbody>
</table>

Response
One of the following HTTP status codes is returned:

- 200 Ok—Request was successful.
- 400 Bad Request—Request parameters were incomplete or invalid.

No XML body is returned in the response.

If the request fails, the status message indicates that an error occurred and one of the following reason codes is returned:

- 430
- 432
- 444
- 500

For additional information on PVR Server reason codes, see Table 8-3.
Deprovisioning the Home

Following are the request components:

Request Direction
PVR Scheduler to PVR Server

Description
This API directs the PVR server to delete an existing HomeID and it’s associated set-tops and content.

Request Type
HTTP method supported: POST.

Request Example
POST /RSDVR?action=DeprovisionHome HTTP/1.1
Content-Type: text/xml
Connection: Close
cseq: 2
Content-Length: 257

<?xml version="1.0" encoding="utf-8"?>
<DeprovisionHome
HomeID = "987654321234567"
/>

Table 8-6 describes the XML body elements, sub elements, and attributes of the DeprovisionHome request XML.

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes/Sub Elements</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deprovision Home</td>
<td>—</td>
<td>Element</td>
<td>Element that the home details.</td>
</tr>
<tr>
<td></td>
<td>HomeID</td>
<td>xs:string</td>
<td>Unique ID to identify the subscriber’s account.</td>
</tr>
</tbody>
</table>

Response
One of the following HTTP status codes is returned:

- 200 Ok—Request was successful.
- 400 Bad Request—Request parameters were incomplete or invalid.

No XML body is returned in the response.

If the request fails, the status message indicates that an error occurred and one of the following reason codes is returned:

- 430
- 432
- 439
- 500

For additional information on PVR Server reason codes, see Table 8-3.
Removing an STB

Following are the request components:

**Request Direction**
PVR Scheduler to PVR Server

**Description**
This API directs the PVR server to disassociate an existing set-top from a Home account. The associated content and recording requests are unaffected.

**Request Type**
HTTP method supported: POST.

**Request Example**

```plaintext
POST /RSDVR?action=RemoveSTB HTTP/1.1
Content-Type: text/xml
Connection: Close
cseq: 2
Content-Length: 257

<?xml version="1.0" encoding="utf-8"?>
<RemoveSTB
  HomeID = "987654321234567"
  MACAddress = "9B8A84EF0101"
/>
```

Table 8-7 describes the XML body elements, sub elements, and attributes of the RemoveSTB request XML.

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes/Sub Elements</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>RemoveSTB</td>
<td>—</td>
<td>Element</td>
<td>Element that represents a set-top.</td>
</tr>
<tr>
<td></td>
<td>HomeID</td>
<td>xs:string</td>
<td>Unique ID to identify the subscriber’s account.</td>
</tr>
<tr>
<td></td>
<td>MACAddress</td>
<td>xs:string</td>
<td>Unique ID to identify the subscriber’s set-top.</td>
</tr>
</tbody>
</table>

**Response**

One of the following HTTP status codes is returned:

- 200 Ok—Request was successful.
- 400 Bad Request—Request parameters were incomplete or invalid.

No XML body is returned in the response.

If the request fails, the status message indicates that an error occurred and one of the following reason codes is returned:

- 430
- 432
- 444
• 500
For additional information on PVR Server reason codes, see Table 8-3

Get Home Profile Details

Request Direction
PVR Scheduler to PVR Server

Description
This API directs the PVR server to return the allocated space, the current free space and service group for a specified HomeID. It also returns a list of set-tops associated with the Home ID and it’s corresponding quotas

Request Example
POST /RSDVR?action=GetHomeProfile HTTP/1.1
Content-Type: text/xml
Connection: Close
cseq: 2
Content-Length: 257
<?xml version="1.0" encoding="utf-8"?>
<GetHomeProfile
HomeID = "987654321234567"
/>

Response
One of the following HTTP status codes is returned:
• 200 Ok—Request was successful.
• 400 Bad Request—Request parameters were incomplete or invalid.

The Home profile details is returned in the XML body response. Table 8-8 describes the XML body elements, sub elements, and attributes returned in the response element.

If the request fails, the status message indicates that an error occurred and one of the following reason codes is returned:
• 430
• 432
• 500
For additional information on PVR Server reason codes, see Table 8-3

Table 8-8 HomeProfile Element

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes/Sub Elements</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>HomeProfile</td>
<td>—</td>
<td>Element</td>
<td>Element that represents the home profile details.</td>
</tr>
<tr>
<td>AllocationSize</td>
<td>xs:string</td>
<td></td>
<td>Number of blocks associated to Home ID.</td>
</tr>
<tr>
<td>FreeSize</td>
<td>xs:string</td>
<td></td>
<td>Number of free blocks available for the set-tops.</td>
</tr>
</tbody>
</table>
Get STB Profile Details

**Request Direction**
PVR Scheduler to PVR Server

**Description**
This API directs the PVR server to return the allocated space, the current free space and service group for a specified set-top associated to a specific Home ID.

**Request Example**
```
POST /RSDVR?action=GetSTBProfile HTTP/1.1
Content-Type: text/xml
Connection: Close
cseq: 2
Content-Length: 257

<?xml version="1.0" encoding="utf-8"?>
<GetSTBProfile
HomeID = "987654321234567"
MACAddress = "ABCDEFFEDCBA"
/>
```

---

**Table 8-8**  
*HomeProfile Element (continued)*

<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ServiceGroup</td>
<td>xs:string</td>
<td>Service group associated to the subscriber’s account/set-top.</td>
</tr>
<tr>
<td>STBProfileList</td>
<td>List</td>
<td>Element that represents a list of set-tops associated to a specified Home ID.</td>
</tr>
<tr>
<td>MACAddress</td>
<td>xs:string</td>
<td>Element that represents the MAC Address of the set-top.</td>
</tr>
</tbody>
</table>

---

**Response Example**
```
<?xml version="1.0" encoding="utf-8"?>
<HomeProfile
AllocationSize = "160000000"
FreeSize = "120000000"
ServiceGroup = "1234"
><STBProfileList
<STBProfile
MACAddress = "9B8A84EF0101"
/>
<STBProfile
MACAddress = "9B8A84EF0102"
/>
...
</STBProfileList>
</HomeProfile>
```
Response
One of the following HTTP status codes is returned:

- 200 Ok—Request was successful.
- 400 Bad Request—Request parameters were incomplete or invalid.

The set-top profile details is returned in the XML body response. Table 8-9 describes the XML body elements, sub elements, and attributes returned in the response element.

If the request fails, the status message indicates that an error occurred and one of the following reason codes is returned:

- 430
- 432
- 444
- 500

For additional information on PVR Server reason codes, see Table 8-3

Table 8-9  STBQuota Element

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes/Sub Elements</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>STBQuota</td>
<td>—</td>
<td>Element</td>
<td>Element that represents the set-top associated to a specified Home ID.</td>
</tr>
<tr>
<td>MACAddress</td>
<td>xs:string</td>
<td></td>
<td>Element that represents the MAC Address of the set-top.</td>
</tr>
<tr>
<td>AllocationSize</td>
<td>xs:string</td>
<td></td>
<td>Number of blocks allocated to the set-top. Block size is 512 bytes.</td>
</tr>
<tr>
<td>FreeSize</td>
<td>xs:string</td>
<td></td>
<td>Number of free blocks available for the set-top. Block size is 512 bytes.</td>
</tr>
</tbody>
</table>

Response Example

```xml
<?xml version="1.0" encoding="utf-8"?>
<STBQuota
  MACAddress = "ABCDEFFEDCBA"
  AllocationSize = "160000000"
  FreeSize = "120000000"
/>```

Scheduling a Recording

Request Direction
PVR Scheduler to PVR Server

Description
This API directs the PVR server to schedule a recording of a given program at all HomeID/Set-top pairs specified in the list.
Request Example

POST /RSDVR?action=ScheduleRecording HTTP/1.1
Content-Type: text/xml
Connection: Close
cseq: 2
Content-Length: 257

<?xml version="1.0" encoding="utf-8"?>
<ScheduleRecording
AssetID = "222333444555666777"
CallSign = "ABC"
StartTime = "2005-09-02T12:00:00Z"
EndTime = "2005-09-02T12:59:59Z" >
<PVRList>
<PVRListItem
HomeID = "1234567890123456"
MACAddress = "9B8A84EF0102"
/>
<PVRListItem
HomeID = "1234567890123459"
MACAddress = "9BACEBEF0102"
/>
...
</PVRList>
</ScheduleRecording>

Table 8-10  describes the XML body elements, sub elements, and attributes of the request element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes/SubElements</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ScheduleRecording</td>
<td>Element</td>
<td></td>
<td>Element that represents the recording element.</td>
</tr>
<tr>
<td>AssetID</td>
<td>xs:string</td>
<td></td>
<td>Unique ID to identify the program content.</td>
</tr>
<tr>
<td>CallSign</td>
<td>xs:string</td>
<td></td>
<td>Unique ID to identify the program channel.</td>
</tr>
<tr>
<td>StartTime</td>
<td>xs:dateTime</td>
<td></td>
<td>Element that represents the start time of the recording in UTC time zone. The date and time format is specified in 8601:2000 format.</td>
</tr>
<tr>
<td>EndTime</td>
<td>xs:dateTime</td>
<td></td>
<td>Element that represents the end time of the recording in UTC time zone. The date and time format is specified in 8601:2000 format.</td>
</tr>
<tr>
<td>PVRList</td>
<td>Subelement</td>
<td></td>
<td>Element that represents a list of HomeID/set-top pairs to schedule the recording.</td>
</tr>
<tr>
<td>PVRList</td>
<td>ScheduleRecording</td>
<td>Subelement</td>
<td>Element that represents a list of HomeID/set-top pairs to schedule the recording.</td>
</tr>
<tr>
<td>PVRListItem</td>
<td>PVRList subelement</td>
<td></td>
<td>Element that represents a Home ID/set-top.</td>
</tr>
</tbody>
</table>
Scheduling a Recording

Response

One of the following HTTP status codes is returned:

- 200 Ok—Request was successful.
- 400 Bad Request—Request parameters were incomplete or invalid.

If the recording fails, a list of HomeID/set-top pairs in which the recording failed and the corresponding reason code/description is returned in the XML body response. Table 8-11 describes the XML body elements, sub elements, and attributes returned in the response element.

The reason codes returned in the XML are:

- 430
- 432
- 436
- 437
- 454
- 500

For additional information on PVR Server reason codes, see Table 8-3

Table 8-10  ScheduleRecording Element

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes/SubElements</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>HomeID</td>
<td>xs:string</td>
<td></td>
<td>Unique ID to identify the subscriber’s account.</td>
</tr>
<tr>
<td>MACAddress</td>
<td>xs:string</td>
<td></td>
<td>Unique ID to identify the subscriber’s set-top.</td>
</tr>
</tbody>
</table>

Table 8-11  ScheduleRecordingResponse Element

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes/SubElements</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ScheduleRecordingResponse</td>
<td>—</td>
<td>Element</td>
<td>Element that represents the scheduled recording response.</td>
</tr>
<tr>
<td>FailedPVRList</td>
<td>Subelement</td>
<td></td>
<td>Optional.Element that represents a list of failed HomeID/set-top pairs in which the recording failed. If there are no failed recordings then no list is provided.</td>
</tr>
<tr>
<td>FailedPVR</td>
<td>Subelement</td>
<td></td>
<td>Element that represents a Home ID/set-top.</td>
</tr>
<tr>
<td>FailedPVRList</td>
<td></td>
<td></td>
<td>Element that represents a list of failed HomeID/set-top pairs in which the recording failed.</td>
</tr>
<tr>
<td>HomeID</td>
<td>xs:string</td>
<td></td>
<td>Unique ID to identify the subscriber’s account.</td>
</tr>
<tr>
<td>MACAddress</td>
<td>xs:string</td>
<td></td>
<td>Unique ID to identify the subscriber’s set-top.</td>
</tr>
</tbody>
</table>
Deleting a Recording

Request Direction
PVR Scheduler to PVR Server

Description
This API directs the PVR server to delete a scheduled or existing recording of a given asset at all HomeID/Set-top pairs specified in the list.

Request Example
POST /RS DVR?action=DeleteRecording HTTP/1.1
Content-Type: text/xml
Connection: Close
cseq: 2
Content-Length: 257
<?xml version="1.0" encoding="utf-8"?>
<DeleteRecording
AssetID = "222333444555666777">
<PVRList>
<PVRLListItem
HomeID = "1234567890123456"
MACAddress = "9B8A84EF0102"/>
</PVRList>
</DeleteRecording>
Deleting a Recording

Table 8-12 describes the XML body elements, sub elements, and attributes of the request element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes/SubElements</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeleteRecording</td>
<td>Element</td>
<td></td>
<td>Element that represents the recording element.</td>
</tr>
<tr>
<td>AssetID</td>
<td>xs:string</td>
<td></td>
<td>Unique ID to identify the program content.</td>
</tr>
<tr>
<td>PVRList</td>
<td>Subelement</td>
<td></td>
<td>Element that represents a list of HomeID/set-top pairs to delete the recording.</td>
</tr>
<tr>
<td>ScheduleRecording</td>
<td>Subelement</td>
<td></td>
<td>Element that represents a list of HomeID/set-top pairs to delete the recording.</td>
</tr>
<tr>
<td>PVRLISTitem</td>
<td>Subelement</td>
<td></td>
<td>Element that represents a Home ID/set-top.</td>
</tr>
<tr>
<td>HomeID</td>
<td>xs:string</td>
<td></td>
<td>Unique ID to identify the subscriber’s account.</td>
</tr>
<tr>
<td>MACAddress</td>
<td>xs:string</td>
<td></td>
<td>Unique ID to identify the subscriber’s set-top.</td>
</tr>
</tbody>
</table>

Response

One of the following HTTP status codes is returned:

- 200 Ok—Request was successful.
- 400 Bad Request—Request parameters were incomplete or invalid.

If the record deletion fails, a list of HomeID/set-top pairs in which the delete recording failed and the corresponding reason code/description is returned in the XML body response. Table 8-13 describes the XML body elements, sub elements, and attributes returned in the response element.

The reason codes returned in the XML are:

- 430
- 432
- 438
- 500

For additional information on PVR Server reason codes, see Table 8-3.
### Table 8-13  DeleteRecordingResponse Element

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes/Sub Elements</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeleteRecordingResponse</td>
<td>—</td>
<td>Element</td>
<td>Element that represents the delete recording response.</td>
</tr>
<tr>
<td>FailedPVRList</td>
<td>Subelement</td>
<td></td>
<td>Optional Element that represents a list of failed HomeID/set-top pairs in which the recording deletion failed.</td>
</tr>
<tr>
<td>FailedPVRList</td>
<td>ScheduleRecordingResponse Subelement</td>
<td></td>
<td>Element that represents a list of failed HomeID/set-top pairs in which the record deletion failed.</td>
</tr>
<tr>
<td>FailedPVR</td>
<td>Subelement</td>
<td></td>
<td>Element that represents a Home ID/set-top.</td>
</tr>
<tr>
<td>FailedPVR</td>
<td>FailedPVRList subelement</td>
<td></td>
<td>Element that represents a Home ID/set-top.</td>
</tr>
<tr>
<td>HomeID</td>
<td>xs:string</td>
<td></td>
<td>Unique ID to identify the subscriber’s account.</td>
</tr>
<tr>
<td>MACAddress</td>
<td>xs:string</td>
<td></td>
<td>Unique ID to identify the subscriber’s set-top.</td>
</tr>
<tr>
<td>ReasonCode</td>
<td>xs:integer</td>
<td></td>
<td>Element that represents the reason code for failed recording deletion.</td>
</tr>
<tr>
<td>ReasonDescription</td>
<td>xs:string</td>
<td></td>
<td>Element that represents the reason description for failed recording deletion.</td>
</tr>
</tbody>
</table>

**Response Example**

```xml
<?xml version="1.0" encoding="utf-8"?>
/DeleteRecordingResponse>
<FailedPVRList>
  <FailedPVR
    HomeID = "1112223334445555"
    MACAddress = "AB12DC3864AE"
    ReasonCode = "431"
    ReasonDescription = "Home ID Does Not Exist"
  />
  <FailedPVR
    HomeID = "9992223337774442"
    MACAddress = "AB12DC3864AC"
    ReasonCode = "431"
    ReasonDescription = "Home ID Does Not Exist"
  />
</FailedPVRList>
</DeleteRecordingResponse>
```
Chapter 8  PVR Scheduler Interface APIs

Notifying the Recording Completion

Request Direction
PVR Server to PVR Scheduler

Description
This API notifies the PVR Scheduler when a recording request is fulfilled. The notification includes the Asset ID and a list of Home ID/set-tops in which the recording failed.

Request
HTTP method supported: POST.

Request Body XML

```
POST /RSDVR?action=RecordingComplete HTTP/1.1
Content-Type: text/xml
Connection: Close
cseq: 2
Content-Length: 257
<?xml version="1.0" encoding="utf-8"?>
<RecordingComplete
AssetID = "222333444555666777">
<FailedPVRList>
<FailedPVR
HomeID = "987654321234567"
MACAddress = "AABBCCDDEEFF"
ReasonCode = "435"
ReasonDescription = "Recording Aborted"
/>
<FailedPVR
HomeID = "987654321234568"
MACAddress = "BBAACCDDFFEE"
ReasonCode = "435"
ReasonDescription = "Recording Aborted"
/>
...
</FailedPVRList>
</RecordingComplete>
```

Table 8-14 describes the XML body elements, sub elements, and attributes of the request element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes/SubElements</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecordingComplete</td>
<td>Element</td>
<td>Element</td>
<td>Element that represents the completion of a recording request.</td>
</tr>
<tr>
<td>AssetID</td>
<td>xs:string</td>
<td>Subelement</td>
<td>Optional.Element that represents a list of failed HomeID/set-top pairs in which the recording request failed.</td>
</tr>
</tbody>
</table>
### Table 8-14 RecordingComplete Element

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes/SubElements</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>FailedPVRList</td>
<td>RecordingComplete Subelement</td>
<td>xs:string</td>
<td>Element that represents a list of failed HomeID/set-top pairs in which the record request failed.</td>
</tr>
<tr>
<td>FailedPVR</td>
<td>FailedPVRList Subelement</td>
<td>xs:string</td>
<td>Element that represents a Home ID/set-top.</td>
</tr>
<tr>
<td>HomeID</td>
<td>xs:string</td>
<td></td>
<td>Unique ID to identify the subscriber’s account.</td>
</tr>
<tr>
<td>MACAddress</td>
<td>xs:string</td>
<td></td>
<td>Unique ID to identify the subscriber’s set-top.</td>
</tr>
<tr>
<td>ReasonCode</td>
<td>xs:integer</td>
<td></td>
<td>Element that represents the reason code for failed recording.</td>
</tr>
<tr>
<td>ReasonDescription</td>
<td>xs:string</td>
<td></td>
<td>Element that represents the reason description for failed recording.</td>
</tr>
</tbody>
</table>

If the recording request fails on a set-top, one of the following reason code is specified in the XML.

- 435
- 440
- 441
- 442
- 500

For additional information on PVR Server reason codes, see Table 8-3.

**Response**

One of the following HTTP status codes is returned:

- 200 Ok—Request was successful.
- 400 Bad Request—Request parameters were incomplete or invalid.

No XML is returned in the response.
VBO APIs

In Cisco VDS-TV 3.7 Release, Cisco Video Back Office (VBO) is introduced in the VDS-TV network replacing Ericsson’s ISA based backoffice OpenStream. To enable communication with VBO, VDS-TV supports HTTP based interfaces for stream setup and content acquisitions.

This chapter describes the format and content of VVIM/CDSM APIs for VBO.

- Get Content List, page 9-1
- GetSessionPlay History, page 9-4

All HTTP request messages follow the format given in the “HTTP Headers” section on page 1-3.

Get Content List

The Ingest Manager invokes the HTTP GET method to the Content Server to retrieve the list of content based on a specific criteria.

Request

HTTP method supported: GET.

Request Example

URI:
http://<device-IP>/contentserver/api.php?messageType=GetContentList

where <device-IP> is Content Ingest Manager IP

Response

If the request fails, one of the following HTTP status codes is returned:

- 200 Ok—Request was successful.
- 400 Bad Request—Request parameters were incomplete or invalid.
- 500 Internal Server Error

If the request succeeds, the List element is returned in the XML body response.
Table 9-1 describes the XML body elements, subelements, and attributes returned in the List element.

### Table 9-1 GetContentListResponse List Element

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes/SubElements</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ContentList</td>
<td>—</td>
<td>List Element</td>
<td>Element that represents a list of contents.</td>
</tr>
<tr>
<td>ContentFile</td>
<td>—</td>
<td>ContentList</td>
<td>Element that represents a content.</td>
</tr>
<tr>
<td>contentId</td>
<td>xs:string</td>
<td></td>
<td>Unique ID (PAID =ProviderId:AssetId)of the content item (not physical content file). This element is required for Content Servers delivering legacy QAM based content.</td>
</tr>
<tr>
<td>contentPath</td>
<td>xs:string</td>
<td></td>
<td>Logical path (including file name) used to locate the content file. This element is required for Content Servers delivering IP content and may be included for QAM based content.</td>
</tr>
<tr>
<td>volume</td>
<td>xs:string</td>
<td></td>
<td>Name of the volume where the content is stored. This element is required for Content Servers delivering legacy QAM based content.</td>
</tr>
<tr>
<td>TransferStatus</td>
<td>—</td>
<td>ContentList</td>
<td>Element that represents the transfer status of a content.</td>
</tr>
<tr>
<td>statusCode</td>
<td>xs:string</td>
<td></td>
<td>Status code to indicate the successful/failed transfer of content from content server to ingest manager.</td>
</tr>
<tr>
<td>statusText</td>
<td>xs:string</td>
<td></td>
<td>Element that represents the status text to indicate the successful/failed transfer of content from content server to ingest manager.</td>
</tr>
<tr>
<td>ContentMetadata</td>
<td>—</td>
<td>ContentList</td>
<td>Element that represents the metadata of a content.</td>
</tr>
</tbody>
</table>

### Table 9-1 GetContentListResponse List Element (continued)

<table>
<thead>
<tr>
<th>Element</th>
<th>xs:string</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>xs:string</td>
<td>“InService” or “OutOfService” depends upon the state of the content. If content is ready to play out then it should be “InService” if not then it should be “OutOfService”. It should be used only in the content query requests like “GetContentInfo”.</td>
</tr>
<tr>
<td>fileSize</td>
<td>xs:string</td>
<td>Element that represents the size of the file in bytes.</td>
</tr>
<tr>
<td>duration</td>
<td>xs:string</td>
<td>Element that represents the actual length of content in milliseconds.</td>
</tr>
<tr>
<td>avgBitrate</td>
<td>xs:string</td>
<td>Element that represents the target average bit rate of highest quality version of content in bps.</td>
</tr>
</tbody>
</table>
| encryptionType   | xs:string| Element that represents the pre-encryption type for non-ABR content. Supported values include:  
|                  |           | • NONE  
|                  |           | • OLES  
|                  |           | • PKES |

HTTP/1.1 200 OK  
Content-Type: application/xml  
Content-Length: 913  

```xml
<GetContentListResponse xmlns:ns2="urn:com:cisco:vbo:ingest">
  <ns2:Response code="0">NO ERROR</ns2:Response>
  <ns2:ContentList>
    <ns2:ContentFile  
        contentId="urn:X-ADI11:indemand.com:INTL0609000007055610"  
        contentPath="XYZ.mpg" volume="Volume1" />
    <ns2:TransferStatus statusCode="102" statusText="TRANSFER SUCCESS"/>
    <ns2:ContentMetadata  
        state="InService"  
        fileSize="1709877210"  
        duration="6901230"  
        avgBitrate="3750000"  
        encryptionType="PKES" />
    <ns2:ContentFile  
        contentId="urn:X-ADI11:NBC.com:NBC0609000007055610"  
        contentPath="ABC.mpg" volume="Volume1" />
    <ns2:TransferStatus statusCode="102" statusText="TRANSFER SUCCESS"/>
    <ns2:ContentMetadata  
        state="InService"  
        fileSize="1709877210"  
        duration="6901230"  
        avgBitrate="3750000"  
        encryptionType="OLES" />
  </ns2:ContentList>
</GetContentListResponse>
```
GetSessionPlay History

The ODRM invokes HTTP GET method to retrieve the session play history based on the server session ID.

Request
HTTP method supported: GET.
Required: ServerSessionId

Request Example
http://<cdsm_hostIP>/apis/TrickmodeEvents.php?messageType=GetPlayoutHistory&ServerSessionId=be074250-cc5a-11d9-8cd5-0800200c9a66

Response
If the request fails, one of the following HTTP status codes is returned:
- 200 Ok—Request was successful.
- 400 Bad Request—Request parameters were incomplete or invalid.
- 500 Internal Server Error
If the request succeeds, the List element is returned in the XML body response.
Table 9-2 describes the XML body elements, subelements, and attributes returned in the List element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes/Sub Elements</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>PlayoutHistory</td>
<td>—</td>
<td>List Element</td>
<td>Element that represents a list playout history.</td>
</tr>
<tr>
<td>StreamResources</td>
<td>Subelement</td>
<td></td>
<td>Element that represents a stream resource</td>
</tr>
<tr>
<td>StreamResources</td>
<td>—</td>
<td>PlayoutHistory Subelement</td>
<td>Element that represents a stream resource.</td>
</tr>
<tr>
<td>ResourceId</td>
<td>Subelement</td>
<td></td>
<td>Element that represents a unique ID to identify this Streaming Server resource for this Playout History.</td>
</tr>
<tr>
<td>SopName</td>
<td>Subelement</td>
<td></td>
<td>Element that represents Streaming Server output port name.</td>
</tr>
<tr>
<td>ProviderId</td>
<td>Subelement</td>
<td></td>
<td>Element that represents ADI provider ID.</td>
</tr>
<tr>
<td>AssetId</td>
<td>Subelement</td>
<td></td>
<td>Element that represents ADI asset ID.</td>
</tr>
<tr>
<td>FileURL</td>
<td>Subelement</td>
<td></td>
<td>Element that represents Volume name, CDN name, or full path to the content file.</td>
</tr>
</tbody>
</table>
### Table 9-2  
**PlayoutHistory List Element (continued)**

<table>
<thead>
<tr>
<th>Element</th>
<th>Subelement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Slot</strong></td>
<td>Subelement</td>
<td>Element that represents Index of the asset in the PlayList. An index is used instead of AssetID /ProviderID since a given piece of content may appear multiple times in a play list. This is zero based.</td>
</tr>
<tr>
<td><strong>EventHistory</strong></td>
<td>—</td>
<td>Element that represents a Event History.</td>
</tr>
<tr>
<td><strong>StartEvent</strong></td>
<td>—</td>
<td>Element that represents a Event Start.</td>
</tr>
<tr>
<td><strong>EventHistory</strong></td>
<td>—</td>
<td>Element that represents a Event History.</td>
</tr>
<tr>
<td><strong>ResourceId</strong></td>
<td>Subelement</td>
<td>Element that represents the ID of the StreamerResource associated with this event.</td>
</tr>
<tr>
<td><strong>NPT</strong></td>
<td>Subelement</td>
<td>Element that represents the normal play time in this streamer resource when the event occurred. Valid values include:</td>
</tr>
</tbody>
</table>
|                   |                         | - `<number>`  
|                   |                         | - BOS  
|                   |                         | - EOS  
|                   |                         | The NPT is in seconds with type float.                                     |
| **State**         | Subelement              | Element that represents the playout state. Valid values include:            |
|                   |                         | - PLAY  
|                   |                         | - PAUSE  
| **Scale**         | Subelement              | Element that represents the scale for the PLAY state. For example, 1.0 is normal speed. |
| **StreamControl Event** | Subelement              | Element that represents a Stream Control Event.                             |
| **ResourceId**    | Subelement              | Element that represents the ID of the StreamerResource associated with this event. |

**Notes:**
- `<number>`: Indicates a number
- BOS: Begin of Stream
- EOS: End of Stream
Table 9-2 PlayoutHistory List Element (continued)

<table>
<thead>
<tr>
<th>Subelement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NPT</td>
<td>Element that represents the normal play time in this streamer resource when the event occurred. Valid values include:</td>
</tr>
<tr>
<td></td>
<td>* &lt;number&gt;</td>
</tr>
<tr>
<td></td>
<td>* BOS</td>
</tr>
<tr>
<td></td>
<td>* EOS</td>
</tr>
<tr>
<td></td>
<td>The NPT is in seconds with type float.</td>
</tr>
<tr>
<td>State</td>
<td>Element that represents the playout state. Valid values include:</td>
</tr>
<tr>
<td></td>
<td>* PLAY</td>
</tr>
<tr>
<td></td>
<td>* PAUSE</td>
</tr>
<tr>
<td>Scale</td>
<td>Element that represents the scale for the PLAY state. For example, 1.0 is normal speed.</td>
</tr>
<tr>
<td>EndEvent</td>
<td>Element that represents a Event End.</td>
</tr>
<tr>
<td>EndEvent</td>
<td>EventHistory Subelement</td>
</tr>
<tr>
<td>Reason</td>
<td>Element that represents the source of the delivery termination. Valid values include:</td>
</tr>
<tr>
<td></td>
<td>* ERROR</td>
</tr>
<tr>
<td></td>
<td>* SERVER</td>
</tr>
<tr>
<td></td>
<td>* USER</td>
</tr>
<tr>
<td>ResourceId</td>
<td>Element that represents the ID of the StreamerResource associated with this event.</td>
</tr>
<tr>
<td>NPT</td>
<td>Element that represents the normal play time in this streamer resource when the event occurred. Valid values include:</td>
</tr>
<tr>
<td></td>
<td>* &lt;number&gt;</td>
</tr>
<tr>
<td></td>
<td>* BOS</td>
</tr>
<tr>
<td></td>
<td>* EOS</td>
</tr>
<tr>
<td></td>
<td>The NPT is in seconds with type float.</td>
</tr>
<tr>
<td>EventDate</td>
<td>Element that represents the date/time the event occurred. This is in ISO 8601:2000 format.</td>
</tr>
</tbody>
</table>

Response Example

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ns2:PlayoutHistory xmlns:ns2="urn:net:beaumaris:common:xsd:1">
  <ns3:ServerSessionId>be074250-cc5a-11d9-8cd5-0800200c9a66</ns3:ServerSessionId>
  <PlayoutHistory>
    <StreamResources>
      <ResourceId>1</ResourceId>
      <SopName></SopName>
      <ProviderId>comcast.com</ProviderId>
    </StreamResources>
    <EndEvent>
      <EventHistory>
      </EventHistory>
    </EndEvent>
    <NPT>
      <number>123.45</number>
    </NPT>
    <State>
      <PLAY/>
    </State>
    <Scale>
      <number>1.0</number>
    </Scale>
    <Reason>
      <ERROR/>
    </Reason>
    <ResourceId>
      <number>123456</number>
    </ResourceId>
    <NPT>
      <number>123.45</number>
    </NPT>
    <EventDate>
      <number>2023-05-01T12:00:00Z</number>
    </EventDate>
  </PlayoutHistory>
</ns2:PlayoutHistory>
```
<AssetId>EGEZ0595967209323857</AssetId>
<FileURL>CDN01</FileURL>
<Slot>1</Slot>
</StreamResources>
<EventHistory>
<StartEvent>
<EventDate>2010-01-25T10:00:00Z</EventDate>
<ResourceId>1</ResourceId>
<NPT>0.00</NPT>
<State>PLAY</State>
<Scale>1.0</Scale>
</StartEvent>
<StreamControlEvent>
<EventDate>2010-01-25T10:00:20Z</EventDate>
<ResourceId>1</ResourceId>
<NPT>20.00</NPT>
<State>PLAY</State>
<Scale>10.0</Scale>
</StreamControlEvent>
<StreamControlEvent>
<EventDate>2010-01-25T10:00:25Z</EventDate>
<ResourceId>1</ResourceId>
<NPT>70.00</NPT>
<State>PLAY</State>
<Scale>1.0</Scale>
</StreamControlEvent>
<EndEvent>
<Reason>USER</Reason>
<ResourceId>1</ResourceId>
<NPT>90.00</NPT>
<EventDate>2010-01-25T10:00:45Z</EventDate>
</EndEvent>
</EventHistory>
</ns2:PlayoutHistory>
TV Playout APIs

This chapter describes the format and content of the TV Playout application program interface (API) messages. The TV Playout API provides a way to retrieve detailed information on content currently playing on playout and barker channels, import TV Playout schedules, create barker streams and playlists, and start and stop barker streams. The TV Playout API messages consist of the following:

- TV Playout Schedule Exporter, page 10-2
- TV Playout Schedule Importer, page 10-4
- TV Playout Contents Currently Playing, page 10-7
- Barker Streams Currently Playing, page 10-8
- TV Playout Channels, page 10-9
- TV Playout Stream Report, page 10-10
- Create Barker Streams, page 10-13
- Create Playlist, page 10-14
- Start/Stop Barker Streams, page 10-15
- Get All Barker Streams, page 10-16

All HTTP request messages of the TV Playout API follow the HTTP GET or POST format. The list of the status codes used in TV Playout responses and a description of the error conveyed by each status code is presented in “TV Playout Errors” section on page 10-17.
TV Playout Schedule Exporter

The TV Playout Schedule Exporter request-response message retrieves a TV Playout schedule.

**Request**

HTTP method supported: POST request with overloaded GET.

Element: **Request**

Table 10-1 describes the XML attributes of the Request element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
<td>—</td>
<td>List element</td>
<td>Element that represents a request.</td>
</tr>
<tr>
<td>Type</td>
<td>xs:string</td>
<td></td>
<td>Type of request; that is, TVPlayoutStreams.</td>
</tr>
<tr>
<td>Name</td>
<td>xs:string</td>
<td></td>
<td>Name of the request; that is, Schedule.</td>
</tr>
<tr>
<td>ChannelList</td>
<td>xs:string</td>
<td></td>
<td>List of channels to be included in the schedule output.</td>
</tr>
<tr>
<td>FromDate</td>
<td>xs:string</td>
<td></td>
<td>Schedule start date in either YYYY-MM-DD or YYYY-M-D format.</td>
</tr>
<tr>
<td>ToDate</td>
<td>xs:string</td>
<td></td>
<td>Schedule end date in either YYYY-MM-DD or YYYY-M-D format.</td>
</tr>
</tbody>
</table>

**Request Example**

In the following example, a TV Playout Schedule is exported based on the ChannelList and the fromDate and toDate specified in the schedule_request.xml file:

```bash
curl -o reply_1_1.xml -F "fileupload=@schedule_request.xml" "http://209.165.201.1/api/services/configure/array/playout/(_method=GET)"
```

The schedule_request.xml file contains the following contents:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Request
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns:ws="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"
 Type="TVPlayoutStreams"
 Name="Schedule"
 ChannelList="test01,test02"
 FromDate="2010-09-28"
 ToDate="2010-09-28"
/>
```

**Response**

If the request fails, one of the following HTTP status codes is returned:

- 400 Bad Request—Request parameters were incomplete or invalid.
- 500 Internal Server Error—Error occurred while retrieving the schedule.

An explanation for the error is presented in the Error element in the body of the response message.

If the request succeeds, the status code 200 OK is returned and the Schedule element is returned in the XML body response.
Table 10-2 describes the XML body elements, subelements, and attributes returned in the Exporter Schedule element.

Table 10-2  **TV Playout Exporter Schedule Element**

<table>
<thead>
<tr>
<th>Element</th>
<th>Attribute/Subelements</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schedule</td>
<td>—</td>
<td>List element</td>
<td>Element that represents a schedule.</td>
</tr>
<tr>
<td>FromDate</td>
<td>xs:string</td>
<td>Schedule start date.</td>
<td></td>
</tr>
<tr>
<td>ToDate</td>
<td>xs:string</td>
<td>Schedule end date.</td>
<td></td>
</tr>
<tr>
<td>Version</td>
<td>xs:string</td>
<td>Version of the API application.</td>
<td></td>
</tr>
<tr>
<td>Generator</td>
<td>xs:string</td>
<td>Name of the API application.</td>
<td></td>
</tr>
<tr>
<td>CreateTime</td>
<td>xs:string</td>
<td>Date and time the schedule was generated by the API application.</td>
<td></td>
</tr>
<tr>
<td>List</td>
<td>Subelement</td>
<td>Element that represents a list of content objects.</td>
<td></td>
</tr>
<tr>
<td>Channel</td>
<td>Subelement</td>
<td>Element that represents a channel.</td>
<td></td>
</tr>
<tr>
<td>List</td>
<td>Schedule subelement</td>
<td>Element that represents a list of content objects.</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>xs:string</td>
<td>Type of list element. This is always Content.</td>
<td></td>
</tr>
<tr>
<td>Content</td>
<td>Subelement</td>
<td>Element that represents a content object.</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>xs:string</td>
<td>Name of the content file.</td>
<td></td>
</tr>
<tr>
<td>IngestPath</td>
<td>xs:string</td>
<td>Full path of the content file on the FTP host, including the username and password.</td>
<td></td>
</tr>
<tr>
<td>long-name</td>
<td>xs:string</td>
<td>Localized name of content object. Only listed if Localized EPG Extensions is enabled.</td>
<td></td>
</tr>
<tr>
<td>long-description</td>
<td>xs:string</td>
<td>Localized description of content object. Only listed if Localized EPG Extensions is enabled.</td>
<td></td>
</tr>
<tr>
<td>Channel</td>
<td>Schedule subelement</td>
<td>Element that represents a channel.</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>xs:string</td>
<td>Name of the output channel.</td>
<td></td>
</tr>
<tr>
<td>Program</td>
<td>Subelement</td>
<td>Element that represents a program. A program refers to content that is scheduled to play on a channel during a specific timeslot.</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>xs:string</td>
<td>Name of a content object or playlist.</td>
<td></td>
</tr>
<tr>
<td>StartTime</td>
<td>xs:string</td>
<td>Date and time the program is scheduled to start.</td>
<td></td>
</tr>
<tr>
<td>EndTime</td>
<td>xs:string</td>
<td>Date and time the program is scheduled to finish.</td>
<td></td>
</tr>
<tr>
<td>List (for Program Type=Playlist)</td>
<td>Subelement</td>
<td>Element that represents a list of content objects for the playlist.</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>xs:string</td>
<td>Name of the content file.</td>
<td></td>
</tr>
<tr>
<td>Loops</td>
<td>xs:integer</td>
<td>Number of times the content loops.</td>
<td></td>
</tr>
</tbody>
</table>
Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Schedule xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:ws="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"
FromDate="2011-10-11"
ToDate="2011-10-11"
Version="1.0"
Generator="CDSM-2.5 CDGM GUI"
CreateTime='2011-10-10:05:57:18' >
  <List Type="Content">
    <Content Name="vic.mpg" IngestPath="ftp://root:rootroot@172.22.98.184/mpeg2/vic.mpg"/>
    <Content Name="bournesupremacybourneidentitydblf.mpg"
       IngestPath="ftp://root:rootroot@172.22.98.184/mpeg2/bournesupremacybourneidentitydblf.mpg"/>
    <Content Name="clockstoppers.mpg"
       IngestPath="ftp://root:rootroot@172.22.98.184/mpeg2/clockstoppers.mpg"/>
    <Content Name="GrandTorinoTS.mpg"
       IngestPath="ftp://root:rootroot@172.22.98.184/mpeg2/GrandTorinoTS.mpg" long-name="TESTING_LOCALIZED"
       long-description="LOCAL_LOCAL_LOCAL"/>
  </List>
  <Channel Name="CHAN-1">
    <Program Type="PLAYLIST" Name="Play_NEW" StartTime="2011-10-12:15:30:00"
       EndTime="2011-10-12:22:02:18" >
      <List Type="Content">
        <Content Name="vic.mpg" Loops="5" />
        <Content Name="bournesupremacybourneidentitydblf.mpg" Loops="1" />
        <Content Name="clockstoppers.mpg" Loops="5" />
      </List>
    </Program>
  </Channel>
  <Channel Name="CHAN-2">
    <Program Type="CONTENT" Name="GrandTorinoTS.mpg" StartTime="2011-10-12:17:00:00"
  </Channel>
</Schedule>
```

TV Playout Schedule Importer

The TV Playout Schedule Importer request-response message imports a TV playout schedule into the Cisco VDS.

Request

HTTP method supported: POST request with overloaded GET.

Element: Schedule

Table 10-3 describes the XML body elements, subelements, and attributes included in the Schedule element.
Chapter 10  TV Playout APIs

TV Playout Schedule Importer

**Table 10-3  TV Playout Importer Schedule Element**

<table>
<thead>
<tr>
<th>Element</th>
<th>Attribute/Subelements</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schedule</td>
<td>—</td>
<td>List element</td>
<td>Element that represents a schedule.</td>
</tr>
<tr>
<td>FromDate</td>
<td>xs:string</td>
<td></td>
<td>Schedule start date.</td>
</tr>
<tr>
<td>ToDate</td>
<td>xs:string</td>
<td></td>
<td>Schedule end date.</td>
</tr>
<tr>
<td>Version</td>
<td>xs:string</td>
<td></td>
<td>Version of the API application.</td>
</tr>
<tr>
<td>Generator</td>
<td>xs:string</td>
<td></td>
<td>Name of the API application.</td>
</tr>
<tr>
<td>CreateTime</td>
<td>xs:string</td>
<td></td>
<td>Date and time the schedule was generated by the API application.</td>
</tr>
<tr>
<td>List</td>
<td>—</td>
<td>Subelement</td>
<td>Element that represents a list of content objects.</td>
</tr>
<tr>
<td>Channel</td>
<td>—</td>
<td>Subelement</td>
<td>Element that represents a channel.</td>
</tr>
<tr>
<td>List</td>
<td>—</td>
<td>Subelement</td>
<td>Element that represents a list of content objects.</td>
</tr>
<tr>
<td>Type</td>
<td>xs:string</td>
<td></td>
<td>Type of list element. This is always Content.</td>
</tr>
<tr>
<td>Content</td>
<td>Subelement</td>
<td></td>
<td>Element that represents a content object.</td>
</tr>
<tr>
<td>Name</td>
<td>xs:string</td>
<td></td>
<td>Name of the content file.</td>
</tr>
<tr>
<td>IngestPath</td>
<td>xs:string</td>
<td></td>
<td>Full path of the content file on the FTP host, including the username and password.</td>
</tr>
<tr>
<td>long-name</td>
<td>xs:string</td>
<td></td>
<td>Localized name of content object. Only listed if Localized EPG Extensions is enabled.</td>
</tr>
<tr>
<td>long-description</td>
<td>xs:string</td>
<td></td>
<td>Localized description of content object. Only listed if Localized EPG Extensions is enabled.</td>
</tr>
<tr>
<td>Channel</td>
<td>—</td>
<td>Subelement</td>
<td>Element that represents a channel.</td>
</tr>
<tr>
<td>Program</td>
<td>Subelement</td>
<td></td>
<td>Element that represents a program. A program refers to content that is scheduled to play on a channel during a specific timeslot.</td>
</tr>
<tr>
<td>Type</td>
<td>xs:string</td>
<td></td>
<td>Type of program; that is, Content or Playlist.</td>
</tr>
<tr>
<td>Name</td>
<td>xs:string</td>
<td></td>
<td>Name of a content object or playlist.</td>
</tr>
<tr>
<td>StartTime</td>
<td>xs:string</td>
<td></td>
<td>Date and time the program is scheduled to start.</td>
</tr>
<tr>
<td>EndTime</td>
<td>xs:string</td>
<td></td>
<td>Date and time the program is scheduled to finish.</td>
</tr>
</tbody>
</table>

**Request Example**

In the following example, the TV Playout Schedule specified in the schedule.xml file is imported into the VDS:

```bash
curl -o reply_1_2.xml -F fileupload=schedule.xml
 'http://209.165.201.1/api/services/configure/array/playout'
```
The schedule.xml file contains the following contents:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Schedule xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:ws="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"
    FromDate="2011-10-11"
   ToDate="2011-10-11"
    Version="1.0"
    Generator="CDSM-2.5 CDSM GUI"
    CreateTime="2011-10-10:05:57:18">
    <List Type="Content">
        <Content Name="vic.mpg"
            IngestPath="ftp://root:rootroot@172.22.98.184/mpeg2/vic.mpg"/>
        <Content Name="bourne-supremacy bourne identity dblf.mpg"
            IngestPath="ftp://root:rootroot@172.22.98.184/mpeg2/bourne-supremacy bourne identity dblf.mpg"/>
        <Content Name="clockstoppers.mpg"
            IngestPath="ftp://root:rootroot@172.22.98.184/mpeg2/clockstoppers.mpg"/>
        <Content Name="Grand Torino TS.mp4"
            IngestPath="ftp://root:rootroot@172.22.98.184/mpeg2/Grand Torino TS.mp4"
            long-name="TESTING LOCALIZED" long-description="LOCAL LOCAL LOCAL"/>
    </List>
    <Channel Name="CHAN-1">
        <Program Type="PLAYLIST" Name="Play_NEW" StartTime="2011-10-15:15:30:00" EndTime="2011-10-15:22:02:18"/>
    </Channel>
    <Channel Name="CHAN-2">
        <Program Type="CONTENT" Name="Grand Torino TS.mp4" StartTime="2011-10-15:17:00:00" EndTime="2011-10-15:22:48:26"/>
    </Channel>
</Schedule>

Response

If the request fails, one of the following HTTP status codes is returned:

- 400 Bad Request—Request parameters were incomplete or invalid.
- 409 Conflict—Conflict exists in the schedule.
- 500 Internal Server Error—Error occurred while importing the schedule.

An explanation for the error is presented in the Error element in the body of the response message.

Note

If there are conflicts in the imported schedule, then the conflicts are returned as part of the API response.

If the request succeeds, the status code 200 OK is returned.
TV Playout Contents Currently Playing

The TV Playout Contents Currently Playing request-response message returns a list of content currently being played by TV playout.

**Request**

HTTP method supported: GET.

**Request Example**

```
curl -o reply_1_3.xml "http://209.165.201.1/api/services/monitor/array/playout/playing"
```

**Response**

If the request fails, the HTTP status code 500 Internal Server Error is returned. This status code indicates that an error occurred while retrieving the list of contents currently playing.

If the request succeeds, the List element is returned in the XML body response.

**Table 10-4** describes the XML body elements, subelements, and attributes returned in the List element.

**Table 10-4** TV Playout Contents Currently Playing List Element

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes/Sub Elements</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>List</td>
<td>—</td>
<td>List Element</td>
<td>Element that represents a list of playout content currently playing.</td>
</tr>
<tr>
<td>Type</td>
<td>xs:string</td>
<td></td>
<td>Type of list. This is always TVPlayoutCurrentPlay.</td>
</tr>
<tr>
<td>Timestamp</td>
<td>xs:string</td>
<td></td>
<td>Date and time the response was generated.</td>
</tr>
<tr>
<td>Channel</td>
<td>—</td>
<td>Subelement</td>
<td>Element that represents a channel.</td>
</tr>
<tr>
<td>Channel</td>
<td>Name</td>
<td>xs:string</td>
<td>Name of the output channel.</td>
</tr>
<tr>
<td>Program</td>
<td>—</td>
<td>Subelement</td>
<td>Element that represents a program. A program refers to content that is scheduled to play on a channel during a specific timeslot.</td>
</tr>
<tr>
<td>Program</td>
<td>Name</td>
<td>xs:string</td>
<td>Name of a content object. This attribute is specified if a content object is associated with the program.</td>
</tr>
<tr>
<td>Program</td>
<td>Playlist</td>
<td>xs:string</td>
<td>Name of a playlist. This attribute is specified if a playlist is associated with the program.</td>
</tr>
<tr>
<td>Program</td>
<td>StartTime</td>
<td>xs:string</td>
<td>Program start date and time.</td>
</tr>
<tr>
<td>Program</td>
<td>EndTime</td>
<td>xs:string</td>
<td>Program end date and time.</td>
</tr>
<tr>
<td>Program</td>
<td>Playing</td>
<td>xs:string</td>
<td>Status that indicates if the content is playing or not.</td>
</tr>
</tbody>
</table>

**Response Example**

```
<?xml version="1.0" encoding="UTF-8"?>
<List Type="TVPlayoutCurrentPlay"
xmlns:xsi="http://www.w3.org/2001/XMLSchema"
xmlns:ws="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"
TimeStamp="2010-5-30:1:00:00"
```
Barker Streams Currently Playing

The Barker Streams Currently Playing request-response message returns a list of Barker streams currently being played.

Request

HTTP method supported: GET.

Request Example

curl -o reply_1_4.xml
"http://209.165.201.1/api/services/monitor/array/barkerstreams/playing"

Response

If the request fails, the HTTP status code 500 Internal Server Error is returned. This status code indicates that an error occurred while retrieving the list of Barker streams currently playing.

An explanation for the error is presented in the Error element in the body of the response message.

If the request succeeds, the List element is returned in the XML body response.

Table 10-5 describes the XML body elements, subelements, and attributes returned in the List element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes/Sub Elements</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>List</td>
<td>—</td>
<td>List Element</td>
<td>Element that represents a list of Barker streams currently playing.</td>
</tr>
<tr>
<td>Type</td>
<td>xs:string</td>
<td></td>
<td>Type of list. This is always BarkerStreamsCurrentlyPlaying.</td>
</tr>
<tr>
<td>Timestamp</td>
<td>xs:string</td>
<td></td>
<td>Date and time the response was generated.</td>
</tr>
<tr>
<td>Stream</td>
<td>Subelement</td>
<td></td>
<td>Element that represents a Barker stream.</td>
</tr>
</tbody>
</table>
Tv Playout Channels

The TV Playout Channels request-response message returns a list of channels configured in the TV Playout.

Request
HTTP method supported: GET.

Request Example

```
curl -o reply_1_5.xml "http://209.165.201.1/api/services/configure/system/outputchannels"
```

Response
If the request fails, the HTTP status code 500 Internal Server Error is returned. This status code indicates that an error occurred while retrieving the list of channels configured in the TV Playout.
If the request succeeds, the List element is returned in the XML body response with list type set to Channels. Table 10-6 describes the XML body elements, subelements, and attributes returned in the List element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes/Sub Elements</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>List</td>
<td></td>
<td>List Element</td>
<td>Element that represents a list of playout channels.</td>
</tr>
<tr>
<td>Type</td>
<td>xs:string</td>
<td></td>
<td>Type of list. This is always Channels.</td>
</tr>
<tr>
<td>Timestamp</td>
<td>xs:string</td>
<td></td>
<td>Date and time the response was generated.</td>
</tr>
<tr>
<td>Channel</td>
<td>Subelement</td>
<td>Channel</td>
<td>Element that represents a channel.</td>
</tr>
<tr>
<td>Name</td>
<td>xs:string</td>
<td>Name</td>
<td>Name of the channel.</td>
</tr>
</tbody>
</table>

Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<List Type="Channels"
     xmlns:xs="http://www.w3.org/2001/XMLSchema"
     xmlns:ws="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"
     TimeStamp="2010-5-30:1:00:00"
>
  <Channel Name="test01"/>
  <Channel Name="test02"/>
</List>
```

TV Playout Stream Report

The TV Playout Stream Report request-response message returns a report of TV Playout streams for a specified time interval.

**Request**

HTTP method supported: POST request with overloaded GET.

**Note**

If a channel is not specified, the report returns a list of all channels. If a time interval is not specified, the report presents information for streams occurring in the last 24 hours.

Element: `ReportRequest`
Table 10-7 describes the XML attributes of the ReportRequest element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReportRequest</td>
<td>—</td>
<td>List Element</td>
<td>Element that represents a report request.</td>
</tr>
<tr>
<td>Type</td>
<td>xs:string</td>
<td></td>
<td>Type of request. This is always TVPlayoutStreams.</td>
</tr>
<tr>
<td>Channel</td>
<td>xs:string</td>
<td></td>
<td>Channel for whom the data is being requested.</td>
</tr>
<tr>
<td>FromDate</td>
<td>xs:string</td>
<td></td>
<td>Report start date in the format YYYY-MM-DD. If this parameter is not specified, it is set to the current date and time minus 24 hours, and the message response returns stream details starting from this point in time.</td>
</tr>
<tr>
<td>ToDate</td>
<td>xs:string</td>
<td></td>
<td>Report end date in the format YYYY-MM-DD. If this parameter is not specified, it is set to the current date and time and the message response returns stream details ending at the current time.</td>
</tr>
</tbody>
</table>

**Request Example**

In the following example, a list of TV Playout streams are requested based on the fromDate and toDate specified in the request.xml file:

```bash
curl -o reply_1_6.xml -F "fileupload=@request.xml" "http://209.165.201.1/api/services/report/system/streams/playout/\(_method=GET\)"
```

The request.xml file contains the following contents:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ReportRequest
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:ws="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"
    Type="TVPlayoutStreams">
    <param name="Channel" value="test01"/>
    <param name="FromDate" value="2010-05-30"/>
    <param name="ToDate" value="2010-05-30"/>
</ReportRequest>
```

**Response**

If the request fails, one of the following HTTP status codes is returned:
- 400 Bad Request—Request parameters were incomplete or invalid.
- 500 Internal Server Error—Error occurred while retrieving the stream report.

An explanation for the error is presented in the Error element in the body of the response message.
If the request succeeds, the Report element is returned in the XML body response.

**Table 10-8** describes the XML body elements, subelements, and attributes returned in the Report element.

**Table 10-8  TV Playout Stream Report Element**

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes/Sub Elements</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Report</td>
<td>—</td>
<td>List Element</td>
<td>Element that represents a playout streams report.</td>
</tr>
<tr>
<td>Type</td>
<td>xs:string</td>
<td>Type of report. This is always TVPlayoutStreams.</td>
<td></td>
</tr>
<tr>
<td>FromDate</td>
<td>xs:string</td>
<td>Report start date.</td>
<td></td>
</tr>
<tr>
<td>ToDate</td>
<td>xs:string</td>
<td>Report end date.</td>
<td></td>
</tr>
<tr>
<td>Channel</td>
<td>xs:string</td>
<td>Channel for whom the data is being presented.</td>
<td></td>
</tr>
<tr>
<td>Stream</td>
<td>Sub-Element</td>
<td>Element that represents a stream.</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>xs:string</td>
<td>Type of stream. This is always Barker.</td>
<td></td>
</tr>
<tr>
<td>Channel</td>
<td>xs:string</td>
<td>Channel for whom the data is being presented.</td>
<td></td>
</tr>
<tr>
<td>Content</td>
<td>xs:string</td>
<td>Identifier of the content object being streamed.</td>
<td></td>
</tr>
<tr>
<td>StartTime</td>
<td>xs:string</td>
<td>Stream start time.</td>
<td></td>
</tr>
<tr>
<td>EndTime</td>
<td>xs:string</td>
<td>Stream end time.</td>
<td></td>
</tr>
<tr>
<td>ServerId</td>
<td>xs:integer</td>
<td>Identifier of the Streamer.</td>
<td></td>
</tr>
<tr>
<td>Status</td>
<td>xs:string</td>
<td>Information on whether the stream was successful.</td>
<td></td>
</tr>
</tbody>
</table>

**Response Example**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Report
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:ws="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"
  Type="TVPlayoutStreams"
  Channel="test01"
  FromDate="2010-5-30"
  ToDate="2010-5-30"
  TimeStamp="2010-5-30:1:00:00" >
  <Channel Name="test01">
    <Stream Type="Barker"
      Channel="test01"
      Content="BBQ1.mpg"
      StartTime="2010-5-30:1:00:00"
      EndTime="2010-5-30:1:23:00"
      ServerId="186"
      Status="Success"/>
    <Stream Type="Barker"
      Channel="test01"
      Content="BBQ2.mpg"
      StartTime="2010-5-30:2:00:00"
      EndTime="2010-5-30:2:23:00"
  </Channel>
</Report>
```
Create Barker Streams

The Create Barker Streams request-response message creates Barker streams.

Request

HTTP method supported: POST request.

Element: List

Table 10-9 describes the XML body elements, subelements, and attributes included in the List element.

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes/Sub Elements</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>List</td>
<td></td>
<td>List Element</td>
<td>Element that represents a Barker stream.</td>
</tr>
<tr>
<td>Type</td>
<td>xs:string</td>
<td></td>
<td>Type of list. This is always Barker.</td>
</tr>
<tr>
<td>Name</td>
<td>xs:string</td>
<td></td>
<td>Name of the Barker Stream.</td>
</tr>
<tr>
<td>Channel</td>
<td>xs:string</td>
<td></td>
<td>Name of the channel.</td>
</tr>
<tr>
<td>Content</td>
<td>Subelement</td>
<td></td>
<td>Element that represents a content object.</td>
</tr>
<tr>
<td>Name</td>
<td>xs:string</td>
<td></td>
<td>Name of the content file.</td>
</tr>
<tr>
<td>Loop</td>
<td>xs:integer</td>
<td></td>
<td>Number of loops.</td>
</tr>
</tbody>
</table>

Request Example

In the following example, a Barker stream is created based on the contents of the barker.xml file:

curl -o reply_1_7.xml -F "fileupload=barker.xml" "http://209.165.201.1/api/services/configure/array/barkerstream"

The barker.xml file contains the following contents:

```xml
<?xml version="1.0" encoding="UTF-8"?>
      Type="Barker"
      Name="Barker1"
      Channel="Channel1">
  <Content Name="BBX_00_102000004.mpg" Loops="2"/>
  <Content Name="BBX_00_102000005.mpg" Loops="4"/>
</List>
```
Response
If the request fails, one of the following HTTP status codes is returned:

- 400 Bad Request—Invalid schema.
- 409 Conflict—Error occurred due to conflicting names.
- 500 Internal Server Error—Error occurred while creating the barker stream. An explanation for the error is presented in the Error element in the body of the response message.

If the request succeeds, the status code 200 OK is returned.

Create Playlist

The Create Playlist request-response message creates a playlist.

Request
HTTP method supported: POST request.

Element: List

Table 10-10 describes the XML body elements, subelements, and attributes included in the List element.

### Table 10-10 Create Playlist List Element

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes/Sub Elements</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>List</td>
<td>—</td>
<td>List Element</td>
<td>Element that represents a playlist.</td>
</tr>
<tr>
<td>Type</td>
<td>xs:string</td>
<td></td>
<td>Type of list. This is always Playlist</td>
</tr>
<tr>
<td>Name</td>
<td>xs:string</td>
<td></td>
<td>Name of the playlist.</td>
</tr>
<tr>
<td>Content</td>
<td>Subelement</td>
<td></td>
<td>Element that represents a content object.</td>
</tr>
<tr>
<td>Name</td>
<td>xs:string</td>
<td></td>
<td>Name of the content file.</td>
</tr>
<tr>
<td>Loop</td>
<td>xs:integer</td>
<td></td>
<td>Number of loops.</td>
</tr>
</tbody>
</table>

Request Example
In the following example, a barker stream is created based on the contents of the playlist.xml file:

```bash
curl -o reply_1_7.xml -F "fileupload=playlist.xml" "http://209.165.201.1/api/services/configure/array/barkerstream"
```

The playlist.xml file contains the following contents:

```xml
<?xml version="1.0" encoding="UTF-8"?>

  Type="Playlist"

  Name="Playlist1">

  <Content Name="BBX_00_102000004.mpg" Loops="2"/>

  <Content Name="BBX_00_102000005.mpg" Loops="4"/>

</List>
```
Response
If the request fails, one of the following HTTP status codes is returned:

- 400 Bad Request—Invalid schema.
- 409 Conflict—Error occurred due to conflicting names.

500 Internal Server Error—Error occurred while creating the playlist. An explanation for the error is presented in the Error element in the body of the response message.

If the request succeeds, the status code 200 OK is returned.

Start/Stop Barker Streams

The Stop/Start Barker Streams request-response message is used to start or stop a specific barker stream.

Request
HTTP method supported: GET.

Start Barker Stream Request Example
```
curl -o reply_1_5.xml
'http://209.165.201.1/api/services/configure/array/id/barkerstream/bAction/start/bName/WebServicesTest01'
```

Stop Barker Stream Request Example
```
curl -o reply_1_5.xml
'http://209.165.201.1/api/services/configure/array/id/barkerstream/bAction/stop/bName/WebServicesTest01'
```

Response
If the request fails, the HTTP status code 500 Internal Server Error is returned.

If the request succeeds, the Barker element is returned in the XML body response.

Table 10-11 describes the XML body elements and attributes returned in the Barker element.
Get All Barker Streams

The Get All Barker Streams request-response message returns a list of all of the barker streams in the system.

**Request**

HTTP method supported: GET.

**Request Example**

curl -o reply_1_5.xml "http://209.165.201.1/api/services/monitor/array/barkerstreams/all"

**Response**

If the request fails, the HTTP status code 500 Internal Server Error is returned.

If the request succeeds, the List element is returned in the XML body response with list type set to BarkerStreams. Table 10-12 describes the XML body elements, subelements, and attributes returned in the List element.

<table>
<thead>
<tr>
<th>Table 10-12</th>
<th>Get All Barker Streams List Element</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Element</strong></td>
<td><strong>Attributes/SubElements</strong></td>
</tr>
<tr>
<td>List</td>
<td>—</td>
</tr>
<tr>
<td>Type</td>
<td>xs:string</td>
</tr>
<tr>
<td>Timestamp</td>
<td>xs:string</td>
</tr>
<tr>
<td>Barker</td>
<td>Subelement</td>
</tr>
</tbody>
</table>
Chapter 10      TV Playout APIs

TV Playout Errors

Errors are reported in the API response message and are also logged to the CDSM. HTTP status codes are used to convey information about the error.

The following is a list of the status codes used in TV Playout responses and includes a description of the error conveyed by each status code:

- **200 OK**—Request was successful.
- **400 Bad Request**—Request parameters were incomplete or invalid.
- **409 Conflict**—Request conflicts with an established rule.
- **500 Internal Server Error**—Unexpected error occurred while fulfilling the request.

The body of the response includes a description of the problem. For example, if the response 400 Bad Request is returned, the body of the response may suggest that the request did not adhere to the XML schema file or that the time range was outside an acceptable range. **Error XML in Body Response**

When an error is reported, the Error element is returned in the XML file in the response. In addition to the status code and error description, the XML file may also include the request that was submitted. The following is an example of the Error xml file returned with the status code 400 Bad Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<Error xmlns:a="http://www.w3.org/2001/XMLSchema"
   xmlns:b="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"
   Code="1010"
   Message="XML Validation: Line 4: Unknown attribute NName">
   <Request xmlns:a="http://www.w3.org/2001/XMLSchema"
      xmlns:b="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"
      NName="Schedule"
      FromDate="2010-05-30"/>
</Error>
```

Table 10-12 Get All Barker Streams List Element (continued)

<table>
<thead>
<tr>
<th>Barker</th>
<th>List Element</th>
<th>Element that represents a barker stream.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>xs:string</td>
<td>Name of the barker stream.</td>
</tr>
<tr>
<td>ChannelName</td>
<td>xs:string</td>
<td>Name of the output channel.</td>
</tr>
<tr>
<td>DestinationIP</td>
<td>xs:integer</td>
<td>Destination IP address.</td>
</tr>
<tr>
<td>DestinationPort</td>
<td>xs:integer</td>
<td>Destination port.</td>
</tr>
<tr>
<td>Status</td>
<td>xs:string</td>
<td>Status of the barker stream.</td>
</tr>
</tbody>
</table>

Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?><List Type="BarkerStreams" TimeStamp="2011-06-01:13:07:48">
  <Barker Name="Barker1" ChannelName="Channel1" DestinationIP="209.165.201.1:45" DestinationPort="12345" Status="Stopped"/>
  <Barker Name="Barker2" ChannelName="Channel2" DestinationIP="209.165.201.1:46" DestinationPort="16385" Status="Playing"/>
  <Barker Name="Barker3" ChannelName="Channel3" DestinationIP="209.165.201.1:47" DestinationPort="16386" Status="Playing"/>
</List>```
In this example, a 400 Bad Request error was returned because an unknown ‘NName’ attribute was found. The request is included and shows the invalid attribute in line 4 of the request.
CHAPTER 11

Configuration APIs

This chapter describes the format and content of the Configuration Application Program Interface (API) messages. This chapter consists of the following sections:

- System Level Configuration API, page 11-1
- Array Level Configuration API, page 11-33
- Server Level Configuration API, page 11-60
- Maintain Section Configuration API, page 11-85

**System Level Configuration API**

The System Level Configuration APIs offers the ability to download or upload an XML file that contains the system level configuration settings by way of an API call using any REpresentational State Transfer (REST) client.

Following are the System level configuration APIs:

- System DNS
- System NTP Server
- System Host Service
- System Array Name
- System VOD Market List
- System Content Distribution Rules
- System Priority Rule List
- System Metadata Normalization Rules
- System QAM Gateway
- System Headend Setup
- System Stream Destination
- System Distributed/Shared ISA Setup
- System Ingest Manager
- System Authentication Manager
- System Ingest Tuning
- System MPEG Tuning
System DNS

The System DNS list request-response message returns a list of all configured system DNS.

GET ALL System DNS Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl http://<cdsm_host>/api/services/configure/system/id/dns

Response

If the request succeeds, the list element is returned in the XML body response containing the list of System DNS.

Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?><DNS xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><DomainSuffix>TestDomain</DomainSuffix><DNSServer>10.197.86.134</DNSServer><DNSServer>10.197.86.135</DNSServer></DNS>
```

POST System DNS Configuration

Following are the request components:

Request Type: HTTP POST

Request

curl --form upload=@/root/dns.xml --form press=OK
"http://<cdsm_host>/api/services/configure/system/id/dns"
Request Body: XML

Request Example
Following is the XML body for the POST request to upload an individual system level DNS:

```xml
<?xml version="1.0" encoding="UTF-8"?>
```

System NTP Server

The System NTP Server list request-response message returns a list of all configured system NTP server.

GET ALL System NTP Server Configuration

Following are the request components:

Request Type: HTTP GET

Request
```
curl http://<cdsm_host>/api/services/configure/system/id/ntp
```

Response
If the request succeeds, the list element is returned in the XML body response containing the list of System NTP.

Response Example
```xml
<?xml version="1.0" encoding="UTF-8"?>
<NTP xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
<NTPServer>10.197.86.134</NTPServer>
</NTP>
```

POST System NTP Server Configuration

Following are the request components:

Request Type: HTTP POST

Request
```
curl --form upload=@/root/ntp.xml --form press=OK
"http://<cdsm_host>/api/services/configure/system/id/ntp"
```

Request Body: XML

Request Example
Following is the XML body for the POST request to upload an individual system level NTP Server:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<NTP xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
<NTPServer>10.197.86.134</NTPServer>
```
System Host Service

The System Host Service list request-response message returns a list of all configured system host service.

GET ALL System Host Service Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl http://<cdsm_host>/api/services/configure/system/id/hostservice

Response

If the request succeeds, the list element is returned in the XML body response containing the list of System host Service.

Response Example

<?xml version="1.0" encoding="UTF-8"?>
<HostServiceList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
<HostService Name='Test_Host' IPAddress='10.197.103.74'></HostService>
</HostServiceList>

POST System Host Service Configuration

Following are the request components:

Request Type: HTTP POST

Request

curl --form upload=@/root/hostservice.xml --form press=OK
"http://<cdsm_host>/api/services/configure/system/id/hostservice"

Request Body: XML

Request Example

Following is the XML body for the POST request to upload an individual system level Host Service:

<?xml version="1.0" encoding="UTF-8"?>
<HostServiceList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
<HostService Name='Test_Host' IPAddress='10.197.103.74'></HostService>
</HostServiceList>
System Array Name

The System Array Name list request-response message returns array name configured in the system.

GET ALL System Array Name Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl http://<cdsm_host>/api/services/configure/system/id/arrayname

Response

If the request succeeds, the list element is returned in the XML body response containing array name configured in the system.

Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?><ArrayNameConfig
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><ArrayName
ArrayID='1991' Name='Array1' Vendor='ISA-OpenStream'></ArrayName> </ArrayNameConfig>
```

POST System Array Name Configuration

Following are the request components:

Request Type: HTTP POST

Request

curl --form upload=@/root/arrayname.xml --form press=OK
"http://<cdsm_host>/api/services/configure/system/id/arrayname"

Request Body: XML

Request Example

Following is the XML body for the POST request to upload an individual system level Array Name:

```xml
<?xml version="1.0" encoding="UTF-8"?><ArrayNameConfig
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><ArrayName
ArrayID='1991' Name='Array1' Vendor='ISA-OpenStream'></ArrayName> </ArrayNameConfig>
```
System VOD Market List

The System VOD Market list request-response message returns a list of all configured system VOD markets including any division names assigned to each market.

GET VOD Market Configuration

Request Type
HTTP method supported: GET.

Request Format

```bash
curl http://<cdsm_host>/api/services/configure/system/id/vodmarkets
```

Response
If the request succeeds, the list element is returned in the XML body response containing the list of VOD markets including any division names assigned to each market. Table 11-1 describes the XML body elements, sub elements, and attributes returned in the list element.

<table>
<thead>
<tr>
<th>Table 11-1 MBOList /AMSList Element</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Element</strong></td>
</tr>
<tr>
<td>MBOList/AMSList</td>
</tr>
<tr>
<td>MBODomain Name</td>
</tr>
<tr>
<td>IPAddress</td>
</tr>
<tr>
<td>Port</td>
</tr>
<tr>
<td>Division</td>
</tr>
<tr>
<td>Maintenance</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Response Example for ATIS C2 server deployment

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<MBOList xmlns="http://www.cisco.com/schemas/VCP8U/CDS-TV/R0/ciscowebsvcs">
  <MBO DomainName="BMS1" IPAddress="172.22.98.36" Port="5000" Division="" Maintenance="False"></MBO>
  <MBO DomainName="BMS2" IPAddress="172.22.98.1" Port="5000" Division="" Maintenance="False"></MBO>
</MBOList>
```
System Level Configuration API

Response Example for MVOD deployment

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AMSList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
  <AMS DomainName="BMS1" IPAddress="172.22.98.36" Port="5000" Division="" Maintenance="False"></AMS>
  <AMS DomainName="BMS2" IPAddress="172.22.98.1" Port="5000" Division="" Maintenance="False"></AMS>
  <AMS DomainName="BMS3" IPAddress="172.22.98.12" Port="3000" Division="" Maintenance="False"></AMS>
</AMSList>
```

POST VOD Market Configuration

Following are the request components:

**Request Type**
HTTP method supported: POST.

**Request Format**
curl --form upload=@/root/vodmarkets.xml --form press=OK
"http://<cdsm_host>/api/services/configure/system/id/vodmarkets"

**Request Body: XML**

The POST REST API can be used either to modify one or more VOD market configuration settings or update the entire VOD market configuration page with the new values. If the DomainName in the XML matches an existing DomainName in the system then the existing record is updated with the new values present in the XML.

**Request Example**

Following is the XML body for the POST request to upload the VOD Market:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<MBOList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
  <MBO DomainName="BMS1" IPAddress="172.22.99.80" Port="5000" Division="Div_A" Maintenance="False"></MBO>
  <MBO DomainName="BMS3" IPAddress="172.22.98.12" Port="3000" Division="" Maintenance="False"></MBO>
</MBOList>
```
System Content Distribution Rules

GET Content Distribution Rules Configuration

The System Content Distribution Rules list request-response message returns a list of all configured system content distribution rules in the system.

Request Type
HTTP method supported: GET.

Request Format

curl http://<cdsm_host>/api/services/configure/system/id/contentrules

Response
If the request succeeds, the List element is returned in the XML body response containing a list of configured content distribution rules in the system.

Table 11-2 describes the XML body elements, subelements, and attributes returned in the List element.

<table>
<thead>
<tr>
<th>Table 11-2 ContentRule List Element</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Element</strong></td>
</tr>
<tr>
<td>ContentRule List</td>
</tr>
<tr>
<td>Provider</td>
</tr>
<tr>
<td>Product</td>
</tr>
<tr>
<td>Market</td>
</tr>
<tr>
<td>Protocol</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Exclude</td>
</tr>
<tr>
<td>Status</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Response Example

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ContentRuleList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
```

Cisco VDS-TV API Guide
GET Content Distribution Rules list (based on filters)

The Content Distribution Rules list request-response message returns a list of configured content distribution rules in the system based on different filter criteria.

**Request Type**

HTTP method supported: GET.

**Request Examples for different filter criteria**

Filter By: none
```
```

Filter By: Provider and VOD Market Site
```
```

Filter By: Provider
```
```

Filter By: Protocol, VOD Market Site and Provider
```
```

Filter By: Exclude
```
```

**Response**

If the request succeeds, the list element is returned in the XML body response containing a list of configured content distribution rules in the system.

**Table 11-2** describes the XML body elements, subelements, and attributes returned in the List element.

**Response Example (no filters)**
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ContentRuleList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
  <ContentRule Provider="FOX" Product="VOD!" Market="BMS4" Protocol="FTP" Exclude="None" Status="disabled"></ContentRule>
  <ContentRule Provider="HBO" Product="MOD" Market="BMS1" Protocol="ATIS" Exclude="None" Status="enabled"></ContentRule>
</ContentRuleList>
```
<ContentRule Provider="HBO" Product="MOD" Market="BMS2" Protocol="FTP" Exclude="None" Status="enabled"></ContentRule>
<ContentRule Provider="SPEED" Product="VOD" Market="BMS4" Protocol="FTP" Exclude="None" Status="disabled"></ContentRule>
</ContentRuleList>

Response Example (with protocol = FTP, provider = HBO, market = BMS2)

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ContentRuleList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
  <ContentRule Provider="HBO" Product="MOD" Market="BMS2" Protocol="FTP" Exclude="None" Status="enabled"></ContentRule>
</ContentRuleList>
```

**POST Content Distribution Rules Configuration**

Following are the request components:

**Request Type**
HTTP method supported: POST.

**Request Format**

curl --form upload=@/root/contentrules.xml --form press=OK "http://<cdsm_host>/api/services/configure/system/id/contentrules"

**Request Body : XML**

**Request Example**
Following is the XML body for the POST request to upload the Content Distribution Rules:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ContentRuleList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
  <ContentRule Provider="ALL" Product="ALL" Market="ALL" Protocol="FTP" Status="Enabled" Exclude="None"></ContentRule>
  <ContentRule Provider="HBO2!" Product="VOD" Market="BMS4" Protocol="ATIS" Status="Enabled" Exclude="None"></ContentRule>
</ContentRuleList>
```

**System Priority Rule List**

The Priority Rule list request-response message returns a list of all configured package priority rules configured in the system

**GET Priority Rule list**

**Request Type**
HTTP method supported: GET.

**Request**
curl http://<cdsm_host>/api/services/configure/system/id/packagepriorityrules
Response
If the request succeeds, the list element is returned in the XML body response containing the list of package priority rules.

Table 11-3 describes the XML body elements, sub elements, and attributes returned in the List element.

Table 11-3  PackageRuleList Element

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes/Sub Elements</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>PackageRuleList</td>
<td>—</td>
<td>List Element</td>
<td>Element that represents package priority rule list.</td>
</tr>
<tr>
<td>Element</td>
<td>Subelement</td>
<td></td>
<td>Element that represents a package priority rule.</td>
</tr>
<tr>
<td>Element</td>
<td>—</td>
<td>PackagePriorityRule</td>
<td>Element that represents a package priority rule</td>
</tr>
<tr>
<td>Asset id</td>
<td>xs:string</td>
<td></td>
<td>A string containing the identifying name of the asset. An Asset_ID shall uniquely identify an asset within a provider’s namespace defined by the Provider_ID attribute. All Asset_IDs will have a fixed length of 20, with the first 4 characters alpha and the last 16 characters numeric.</td>
</tr>
<tr>
<td>Title name</td>
<td>xs:string</td>
<td></td>
<td>A string containing the identifying name of the title.</td>
</tr>
<tr>
<td>Provider id</td>
<td>xs:string</td>
<td></td>
<td>A unique identifier for the asset’s provider.</td>
</tr>
<tr>
<td>Product name</td>
<td>xs:string</td>
<td></td>
<td>A unique identifier for the product (within the provider’s namespace)</td>
</tr>
<tr>
<td>Window days</td>
<td>xs:string</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Additional match</td>
<td>xs:string</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rule expiration</td>
<td>xs:string</td>
<td></td>
<td>Number of days after which the package priority rule expires.</td>
</tr>
</tbody>
</table>

Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?><PackageRuleList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><PackageRule AssetId='Dummy_mAssetId1' Key='1490933067' Title='Dummym_Title1' ProviderId='Dummym_ProviderId1' Product='Dummy_mProduct1' WindowsDays='Yes' AdditionalMatch='' RuleExpiration='172800'></PackageRule></PackageRuleList>
```

POST Priority Rule list

Following are the request components.
Request Type: HTTP POST.

Request

curl --form upload=@/root/pkg_rules.xml --form press=OK
http://<cdsm_host>/api/services/configure/system/id/packagepriorityrules

Request Body XML

Request Example

Following is the XML body for the POST request to upload Package priority rules:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<PackageRuleList
    xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
    <PackageRule
        AssetId='Dummy_mAssetId1' Key='1490933067' Title='Dummym_Title1'
        ProviderId='Dummym_ProviderId1' Product='Dummy_mProduct1' WindowsDays='Yes'
        AdditionalMatch='' RuleExpiration='172800'></PackageRule>
</PackageRuleList>
```

System Metadata Normalization Rules

GET Metadata Normalization Rules list

The Metadata Normalization Rules list request-response message returns a list of all configured Metadata Normalization rules in the system.

Request Type

HTTP method supported: GET

Request Format

curl http://<cdsm_host>/api/services/configure/system/id/metadatanormalizationrules

Response

If the request succeeds, a list element is returned in the XML body response containing a list of configured metadata normalization rules in the system.

Table 11-4 describes the XML body elements, sub elements, and attributes returned in the list element.

<table>
<thead>
<tr>
<th>Table 11-4 MetadataRuleList List Element</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Element</strong></td>
</tr>
<tr>
<td>MetadataRuleList</td>
</tr>
<tr>
<td>Provider</td>
</tr>
<tr>
<td>Product</td>
</tr>
<tr>
<td>Market</td>
</tr>
<tr>
<td>Price</td>
</tr>
</tbody>
</table>
Table 11-4  MetadataRuleList List Element

<table>
<thead>
<tr>
<th>Element</th>
<th>Attributes/Sub Elements</th>
<th>Data Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Price</td>
<td>xs:string</td>
<td></td>
<td>New price of the asset. The price of the asset is replaced by the New Price if there is a match in the metadata normalization rule.</td>
</tr>
<tr>
<td>Exclude</td>
<td>xs:string</td>
<td></td>
<td>The domain name of the market site to be excluded from package publishing. This may contain more than one market site separated by comma.</td>
</tr>
</tbody>
</table>

Response Example

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<MetadataRuleList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
    <MetadataRule Provider="ALL" Product="ALL" Market="ALL" Price="4.45" New Price="4.95" Exclude="None"></MetadataRule>
    <MetadataRule Provider="HBO2!" Product="VOD" Market="BMS4" Price="4.25" New Price="6.00" Exclude="None"></MetadataRule>
    <MetadataRule Provider="HBO2" Product="VOD" Market="BMS2" Price="3.00" New Price="4.50" Exclude="None"></MetadataRule>
    <MetadataRule Provider="HBO2" Product="VOD" Market="DIV1" Price="6.50" New Price="8.00" Exclude="BMS1,BMS2,BMS3"></MetadataRule>
    <MetadataRule Provider="HBO2" Product="VOD" Market="DIV2" Price="4.50" New Price="7.00" Exclude="BMS4"></MetadataRule>
</MetadataRuleList>
```

POST Metadata Normalization Rules Configuration

Following are the request components:

Request Type
HTTP method supported: POST.

Request Format
```
curl --form upload=/root/metadatanormalizationrules.xml --form press=OK
"http://<cdsm_host>/api/services/configure/system/id/metadatanormalizationrules"
```

Request Body XML

Request Example

Following is the XML body for the POST request to upload the Metadata Normalization Rules:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<MetadataRuleList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
    <MetadataRule Provider="ALL" Product="ALL" Market="ALL" Price="3.95" New Price="4.95" Exclude="None"></MetadataRule>
</MetadataRuleList>
```
System QAM Gateway

The System QAM Gateway list request-response message returns a list of all configured system QAM Gateway.

GET ALL System QAM Gateway Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl http://<cdsm_host>/api/services/configure/system/id/qam

Response

If the request succeeds, the list element is returned in the XML body response containing the list of System QAM Gateway.

Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<QAMList
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><QAM IP='192.168.2.42'><QAMStreamGroupPreference StreamGroupName='Sg' QAMMAC=''
Preference='High'><Server ServerID='1255' GroupID='3000' QAMMAC=''/><Server
ServerID='1254' GroupID='3000' QAMMAC=''/></QAMStreamGroupPreference></QAM><QAM IP='192.168.2.43'><QAMStreamGroupPreference StreamGroupName='Sg' QAMMAC=''
Preference='High'><Server ServerID='1255' GroupID='3000' QAMMAC=''/><Server
ServerID='1254' GroupID='3000' QAMMAC=''/></QAMStreamGroupPreference></QAM></QAMList>
```

POST System QAM Gateway Configuration

Following are the request components:

Request Type: HTTP POST

Request

curl --form upload=@/root/qam.xml --form press=OK
"http://<cdsm_host>/api/services/configure/system/id/qam"
Request Body: XML

Request Example
Following is the XML body for the POST request to upload an individual system level QAM Gateway.

```xml
<?xml version="1.0" encoding="UTF-8"?><QAMList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><QAM IP='192.168.2.42'><QAMStreamGroupPreference StreamGroupName='Sg' QAMMAC=''><Preference='High'></Server ServerID='1255' GroupID='3000' QAMMAC=''/><Server ServerID='1254' GroupID='3000' QAMMAC=''/></QAMStreamGroupPreference></QAM><QAM IP='192.168.2.43'><QAMStreamGroupPreference StreamGroupName='Sg' QAMMAC=''><Preference='High'></Server ServerID='1255' GroupID='3000' QAMMAC=''/><Server ServerID='1254' GroupID='3000' QAMMAC=''/></QAMStreamGroupPreference></QAM></QAMList>
```

System Headend Setup

The System Headend Setup list request-response message returns a list of all configured system Headend setup.

GET ALL System Headend Setup Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl http://<cdsm_host>/api/services/configure/system/id/headend

Response

If the request succeeds, the list element is returned in the XML body response containing the list of System headend setup including any division names assigned to the market with configurations.

Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?><Headend xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><ServiceGroupToStreamGroup ServiceGroup='1' StreamGroup='Sg'></ServiceGroupToStreamGroup></Headend>
```

POST System Headend Setup Configuration

Following are the request components:

Request Type: HTTP POST

Request

```bash
curl --form upload=@/root/headend.xml --form press=OK "http://<cdsm_host>/api/services/configure/system/id/headend"
```
Request Body: XML

Request Example
Following is the XML body for the POST request to upload an individual system level Headend Setup configurations:

```xml
<?xml version="1.0" encoding="UTF-8"?><Headend
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><ServiceGroupToStreamGro
up ServiceGroup='1' StreamGroup='Sg'></ServiceGroupToStreamGroup></Headend>
```

System Stream Destination

The System Stream Destination list request-response message returns a list of all configured system stream destination.

GET ALL System Stream Destination Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl http://<cdsm_host>/api/services/configure/system/id/streamdestination

Response

If the request succeeds, the list element is returned in the XML body response containing the list of System stream distribution.

Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?><StreamDestinationList
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><StreamDestination
SubnetAddress='192.168.2.0' SubnetMask='255.255.255.0'><StreamGroupPreference
StreamGroupName='SG2' Preference='None'></StreamGroupPreference><StreamGroupPreference
StreamGroupName='SG'
Preference='High'></StreamGroupPreference></StreamDestination></StreamDestinationList>
```

POST System Stream Destination Configuration

Following are the request components:

Request Type: HTTP POST

Request

curl --form upload=@/root/streamdestination.xml --form press=OK
"http://<cdsm_host>/api/services/configure/system/id/streamdestination"
Request Body: XML

Request Example
Following is the XML body for the POST request to upload an individual system level Stream Destination:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<StreamDestinationList
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
<StreamDestination
SubnetAddress='192.168.2.0' SubnetMask='255.255.255.0'>
<StreamGroupPreference
StreamGroupName='SG2' Preference='None'/>
<StreamGroupPreference
StreamGroupName='SG' Preference='High'/>
</StreamDestination>
</StreamDestinationList>
```

System Distributed/Shared ISA Setup

The System Distributed/Shared ISA Setup list request-response message returns Distributed/Shared ISA setup configuration.

GET ALL System Distributed/Shared ISA Setup Configuration

Following are the request components:

Request Type: HTTP GET

Request
curl http://<cdsm_host>/api/services/configure/system/id/isa

Response
If the request succeeds, the list element is returned in the XML body response containing Distributed/Shared ISA setup configuration.

Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ISASetup
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
<ISA-GeneralSettings
VaultMasterIP='10.197.86.198' VaultMasterPort='3200' WebServicePort='8080'
MSASupport='Disable' Pre-EncryptionSupport='Disable' TME-SCE='Disable'
FTPOutPort='5500'/>
<ISA-ContentStoreSettings
Name='ArroyoVideoContentStore' Kind='Factory' FactoryID='ArroyoContentStoreFactory'
FactoryKind='Factory' NoOfThreads='32'
FTPServerPort='4000'/>
</ISA-GeneralSettings>
</ISASetup>
```

POST System Distributed/Shared ISA Setup Configuration

Following are the request components:

Request Type: HTTP POST

Request
curl --form upload=@/root/isa.xml --form press=OK
"http://<cdsm_host>/api/services/configure/system/id/isa"
Request Body: XML

Request Example
Following is the XML body for the POST request to upload an individual system level Distributed/Shared ISA Setup.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ISASetup
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><ISA-GeneralSettings
VaultMasterIP='10.197.86.198' VaultMasterPort='3200' WebServicePort='8080'
MSASupport='Disable' Pre-EncryptionSupport='Disable' 
FTPOutPort='5500'></ISA-GeneralSettings><ISA-ContentStoreSettings
Name='ArroyoVideoContentStore' Kind='Factory' FactoryID='ArroyoContentStoreFactory'
FactoryKind='Factory' NoOfThreads='32'
FTPServerPort='4000'></ISA-ContentStoreSettings><CORBAEventChannels
EventChannelID='EventChannels' EventChannelKind='Context'
ContentChannelID='ContentChannel' ContentChannelKind='Factory'
FactoriesID='Factories' FactoriesKind='Factory' EventChannelFactory='NotifyEventChannelFactory'
LoadQueryInterval='3'></CORBAEventChannels></ISASetup>
```

System Ingest Manager

The System Ingest Manager list request-response message returns Ingest Manager configuration.

GET ALL System Ingest Manager Configuration

Following are the request components:

**Request Type: HTTP GET**

**Request**
curl http://<cdsm_host>/api/services/configure/system/id/ingestmanager

**Response**
If the request succeeds, the list element is returned in the XML body response containing the Ingest Manager configuration.

**Response Example**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<IngestManagerSettings
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><GeneralSettings
Host='10.197.86.177' CallbackPort='7777' AddlPackWindow='4' FTPTimeOut='3600'
PublishMode='On' MaxRetryCount='2' UseAssetID='No' ManageCORBA='Yes' RequireNotice='Yes'
MetaDataPublish='Enable' MetaDataPublishURL0='ftp://isa:calypso@10.197.86.181/ftp'
MetaDataPublishURL1='ftp://isa:calypso@10.197.86.181/fyp' PublishTimeOut='240'
NumberOfThreadsPerMarket='4' EnableLoadBalancing='Yes' FullyQualifiedNames='Yes'
QualifyDelimiter='1' ResetMarketQueue='No' TCPKeepAliveSupport='Disable'
MarketConnectTimeout='2'></GeneralSettings><IngestSettings IngestMask='1'
NameServiceHost='10.197.86.177' NameServicePort='5000' NotifyServiceHost='10.197.86.177'
NotifyServicePort='5010' NotifyEventChannelFactory='NotifyEventChannelFactory'
EventChannelsID='EventChannels' EventChannelsKind='Factory'
FactoriesID='Factories' FactoriesKind='Factory' PackageChannelID='PackageChannel'
PackageChannelKind='Factory' PackageFactoryID='PackageFactory'
PackageFactoryKind='Factory'></IngestSettings></IngestManagerSettings>
```
**POST System Ingest Manager Configuration**

Following are the request components:

**Request Type: HTTP POST**

**Request**

curl --form upload=@/root/ingestmanager.xml --form press=OK
'http://<cdsm_host>/api/services/configure/system/id/ingestmanager'

**Request Body: XML**

**Request Example**

Following is the XML body for the POST request to upload an individual system level Ingest Manager:

```xml
<?xml version="1.0" encoding="UTF-8"?><IngestManagerSettings xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><GeneralSettings Host='10.197.86.177' CallbackPort='7777' Add1PackWindow='4' FTPTimeOut='3600' PublishMode='On' MaxRetryCount='2' UseAssetID='No' ManageCORBA='Yes' RequireNotice='Yes' MetaDataPublish='Enable' MetaDataPublishURL0='ftp://isa:calypso@10.197.86.181/ftp' MetaDataPublishURL1='ftp://isa:calypso@10.197.86.181/fyp' PublishTimeOut='240' NumberOfThreadsPerMarket='4' EnableLoadBalancing='Yes' FullyQualifiedID='Yes' QualifyDelimiter='1' ResetMarketQueue='No' TCPKeepAliveSupport='Disable' MarketConnectTimeout='2' /></GeneralSettings><IngestSettings IngestMask='1' NameServiceHost='10.197.86.177' NameServicePort='5000' NotifyServiceHost='10.197.86.177' NotifyServicePort='5010' NotifyEventChannelFactory='NotifyEventChannelFactory' EventChannelsID='EventChannels' EventChannelsKind='Context' FactoriesID='Factories' FactoriesKind='Context' PackageChannelID='PackageChannel' PackageChannelKind='Factory' PackageFactoryID='PackageFactory' PackageFactoryKind='Factory' PackageFactoryName='PackageFactory' PackageFactoryServerId='999'></IngestSettings><ContentStoreSettings Type='ISA' URL='http://10.197.86.177:3200'></ContentStoreSettings><EncryptionSettings Type='Disable'></EncryptionSettings></IngestManagerSettings>
```

**System Authentication Manager**

The System Authentication Manager list request-response message returns authentication manager configuration.

**GET ALL System Authentication Manager Configuration**

Following are the request components:

**Request Type: HTTP GET**

**Request**

curl http://<cdsm_host>/api/services/configure/system/id/authentication
Response  
If the request succeeds, the list element is returned in the XML body response containing authentication manager configuration.

Response Example  
```xml
<?xml version="1.0" encoding="UTF-8"?><AuthenticationManager xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><AuthenticationManagerSettings Protocol = 'EventIS' IPAddress = '0.0.0.0' Port = '7794' TimeOut = '5' EventISHostName = 'Hostname2' EventISPort = '20' ServerThreadPool = '5'></AuthenticationManagerSettings></AuthenticationManager>
```

POST System Authentication Manager Configuration  
Following are the request components:

Request Type: HTTP POST  

Request  
```
curl --form upload=@/root/authentication.xml --form press=OK "http://<cdsm_host>/api/services/configure/system/id/authentication"
```

Request Body: XML  

Request Example  
Following is the XML body for the POST request to upload an individual system level Authentication Manager:

```xml
<?xml version="1.0" encoding="UTF-8"?><AuthenticationManager xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><AuthenticationManagerSettings Protocol = 'EventIS' IPAddress = '0.0.0.0' Port = '7794' TimeOut = '5' EventISHostName = 'Hostname2' EventISPort = '20' ServerThreadPool = '5'></AuthenticationManagerSettings></AuthenticationManager>
```

System Ingest Tuning  
The System Ingest Tuning list request-response message returns Ingest tuning configuration.

GET ALL System Ingest Tuning Configuration  
Following are the request components:

Request Type: HTTP GET  

Request  
```
curl http://<cdsm_host>/api/services/configure/system/id/ingesttuning
```

Response  
If the request succeeds, the list element is returned in the XML body response containing the Ingest tuning configuration.

Response Example
POST System Ingest Tuning Configuration

Following are the request components:

Request Type: HTTP POST

Request

curl --form upload=@/root/ingesttuning.xml --form press=OK
"http://<cdsm_host>/api/services/configure/system/id/ingesttuning"

Request Body: XML

Request Example

Following is the XML body for the POST request to upload an individual system level Ingest tuning:

<?xml version="1.0" encoding="UTF-8"?><IngestTuningSettings
xmlns='http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs'>
<TrickModeSettings
Speed0='5X' Speed1='-5X' Speed2='-15X' Speed3='0X' Speed4='0X' Speed5='0X' Speed6='0X'
Speed7='0X' Speed8='0X' Speed9='0X' />
<FailIngestSettings PSIErrors='Disable'
BitRateErrors='Disable' ErrorCountMethod='Per Sample' NumOfMinutes='0 min'
DiscontinuityRate='Ignore' NumOfPictureGaps='3' PictureGapDuration='Ignore'
ContinuityCntErrors='Ignore' NumOfSyncLossErrors='3' SyncLossDuration='5 sec'
/></IngestTuningSettings>

System MPEG Tuning

The System MPEGTuning list request-response message returns MPEG tuning configuration.

GET ALL System MPEG Tuning Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl http://<cdsm_host>/api/services/configure/system/id/mpeg

Response

If the request succeeds, the list element is returned in the XML body response containing the MPEG tuning configuration.

Response Example
POST System MPEG Tuning Configuration

Following are the request components:

**Request Type:** HTTP POST

**Request**

```
curl --form upload=@/root/mpeg.xml --form press=OK
"http://<cdsm_host>/api/services/configure/system/id/mpeg"
```

**Request Body:** XML

**Request Example**

Following is the XML body for the POST request to upload an individual system level MPEG Tuning

```
<?xml version="1.0" encoding="UTF-8"?><MPEGTuning
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><MPEGTuningSettings
PrgIDStdize='Disable' PSI='Disable' SeqEndRemove='Disable' RateStdize='Disable'
Md5Checksum='Disable' DualCAS='Disable' /></MPEGTuning>
```

System IP Nicknames

The System IP Nicknames list request-response message returns a list of all configured system IP Nicknames.

GET ALL System IP Nicknames Configuration

Following are the request components:

**Request Type:** HTTP GET

**Request**

```
curl http://<cdsm_host>/api/services/configure/system/id/ipname
```

**Response**

If the request succeeds, the list element is returned in the XML body response containing the list of System IP Nicknames.

**Response Example**

```
<?xml version="1.0" encoding="UTF-8"?><IPNickNames
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><IPNickName
IPAddress='10.197.102.130' NickName='Server-130'></IPNickName></IPNickNames>
```
POST System IP Nicknames Configuration

Following are the request components:

Request Type: HTTP POST

Request

```
curl --form upload=@/root/ipname.xml --form press=OK
'http://<cdsm_host>/api/services/configure/system/id/ipname'
```

Request Body: XML

Request Example

Following is the XML body for the POST request to upload an individual system level IP Nicknames:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<IPNickNames
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
<IPNickName
IPAddress='10.197.102.130' NickName='Server-130'></IPNickName>
</IPNickNames>
```

System Media Importer/Exporter

The System Media Importer/Exporter list request-response message returns Media Importer/Exporter configuration.

GET ALL System Media Importer/Exporter Configuration

Following are the request components:

Request Type: HTTP GET

Request

```
curl http://<cdsm_host>/api/services/configure/system/id/mediasetup
```

Response

If the request succeeds, the list element is returned in the XML body response containing the Media Importer/Exporter configuration.

Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<MediaServiceSettings
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
<MediaImporterSettings
ImporterType='Host' TransformerType='Tribune' AutoImport='Enable'
FTPServerIP='10.197.103.200' FTPPath='/home/ftpout/auto/' UserName='ftpout' RetryCount='3'
RetryInterval='3' Schedule='Daily' ImportTime='10-51' ImportDays='' Protocol='FTP'
Password='ftpout123'></MediaImporterSettings>
</MediaServiceSettings>
```
**POST System Media Importer/Exporter Configuration**

Following are the request components:

- **Request Type:** HTTP POST

- **Request**

  ```
curl --form upload=@/root/mediasetup.xml --form press=OK 
  "http://<cdsm_host>/api/services/configure/system/id/mediasetup"
  
  Request Body: XML
  
  **Request Example**

  Following is the XML body for the POST request to upload an individual system level Media Importer/Exporter:

  ```
  <?xml version="1.0" encoding="UTF-8"?>
  <MediaServiceSettings
  xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
  <MediaImporterSettings ImporterType='Host' TransformerType='Tribune' AutoImport='Enable'
  FTPServerIP='10.197.103.200' FTPPath='/home/ftpout/auto/' UserName='ftpout' RetryCount='3'
  RetryInterval='3' Schedule='Daily' ImportTime='10-51' ImportDays='' Protocol='FTP'
  Password='ftpout123'></MediaImporterSettings>
  </MediaServiceSettings>
  ```

**System Input Channels**

The System Input Channels list request-response message returns a list of all configured system Input Channels.

**GET ALL System Input Channels Configuration**

Following are the request components:

- **Request Type:** HTTP GET

- **Request**

  ```
curl http://<cdsm_host>/api/services/configure/system/id/inputchannels
  
  Response

  If the request succeeds, the list element is returned in the XML body response containing the list of System Input Channels.

  **Response Example**

  ```
  <?xml version="1.0" encoding="UTF-8"?>
  <InputChannels
  xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
  <Channel Name='ASIA' LWindowLengthDay='2' LWindowLengthMonth='2' GenreCategory='aaa'
  NetworkCategory='bbb' MulticastIP='232.2.1.1' Port='6004' ChannelCode='ASIA' ChannelID='9990'
  Product='MOD' ProviderName='cc' ProviderID='9990' PreviewPeriod='12' LWindowStart='0'
  Encryption='Yes' Rating='45' Price='9.0' BillID='450' CopyProtection='Yes' ViewCanBeResumed='Yes'
  ContentType='SD' ShowType='qwert' ClosedCaptioning='Yes' ProviderQAContact='3'
  MaxViewLengthDays='02' MaxViewLengthHours='02' MaxViewLengthMinutes='02'
  AudioType='Mono'></Channel>
  </InputChannels>
  ```
POST System Input Channels Configuration

Following are the request components:

Request Type: HTTP POST

Request

curl --form upload=@/root/inputchannels.xml --form press=OK
'http://<cdsm_host>/api/services/configure/system/id/inputchannels'

Request Body: XML

Request Example

Following is the XML body for the POST request to upload an individual system level Input Channels:

```xml
<?xml version="1.0" encoding="UTF-8"?><InputChannels
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><Channel
Name='ASIA' LWindowLengthDay='2' LWindowLengthMonth='2' GenreCategory='aaa'
NetworkCategory='bbb' MulticastIP='232.2.1.1' Port='6004' ChannelCode='ASIA'
ChannelID='9990' Product='MOD' ProviderName='cc' ProviderID='9990'
PreviewPeriod='12' Encryption='Yes' Rating='45' Price='9.0'
BillID='450' CopyProtection='Yes' ViewCanBeResumed='Yes'
ContentType='SD' ShowType='qwert' ClosedCaptioning='Yes' ProviderQAContact='3'
MaxViewLengthDays='02' MaxViewLengthHours='02' MaxViewLengthMinutes='02'
AudioType='Mono'></Channel></InputChannels>
```

System Source Output Port

The System Source Output Port list request-response message returns a list of all configured system Source Output Port.

GET ALL System Source Output Port Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl http://<cdsm_host>/api/services/configure/system/id/sourceoutput

Response

If the request succeeds, the list element is returned in the XML body response containing the list of System Source Output Port.

Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?><SourceOutputPort
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><SOP
Name='SOP1'
VirtualIP='192.169.102.253'></SOP></SourceOutputPort>
```
POST System Source Output Port Configuration

Following are the request components:

Request Type: HTTP POST

Request

curl --form upload=@/root/sourceoutput.xml --form press=OK
"http://<cdsm_host>/api/services/configure/system/id/sourceoutput"

Request Body: XML

Request Example
Following is the XML body for the POST request to upload an individual system level Source Output Port.

```xml
<?xml version="1.0" encoding="UTF-8"?><SourceOutputPort
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><SOP Name='SOP1' VirtualIP='192.169.102.253'></SOP></SourceOutputPort>
```

System Output Channels

The System Output Channels list request-response message returns a list of all configured system Output Channels.

GET ALL System Output Channels Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl http://<cdsm_host>/api/services/configure/system/id/outputchannels

Response

If the request succeeds, the list element is returned in the XML body response containing the list of System Output Channels.

Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?><OutputChannelConfig
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><OutputChannel
Name='longivity40001' DestIP='192.168.3.2' DestPort='40001'></OutputChannel><OutputChannel
Name='longivity40002' DestIP='192.168.2.3' DestPort='40002'></OutputChannel></OutputChannelConfig>
```
POST System Output Channels Configuration

Following are the request components:

**Request Type: HTTP POST**

**Request**

```bash
curl --form upload=@/root/outputchannels.xml --form press=OK
'http://<cdsm_host>/api/services/configure/system/id/outputchannels'
```

**Request Body: XML**

**Request Example**

Following is the XML body for the POST request to upload an individual system level output channels:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<OutputChannelConfig xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
  <OutputChannel Name='longivity40001' DestIP='192.168.3.2' DestPort='40001'/>
  <OutputChannel Name='longivity40002' DestIP='192.168.2.3' DestPort='40002'/>
</OutputChannelConfig>
```

System Callsign Setup

The System Callsign Setup list request-response message returns a list of all configured system Callsign Setup.

GET ALL System Callsign Setup Configuration

Following are the request components:

**Request Type: HTTP GET**

**Request**

```bash
curl http://<cdsm_host>/api/services/configure/system/id/callsign
```

**Response**

If the request succeeds, the list element is returned in the XML body response containing the list of System Callsign Setup.

**Response Example**

```
<?xml version="1.0" encoding="UTF-8"?>
<CallSignSetup xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
  <CallSign Name='Chan1467457835' IPAddress='232.2.1.4' Port='6000'/>
  <CallSign Name='Chan1467459443' IPAddress='232.2.1.4' Port='6001'/>
</CallSignSetup>
```
POST System Callsign Setup Configuration

Following are the request components:

Request Type: HTTP POST

Request

curl --form upload=@/root/callsign.xml --form press=OK
 "http://<cdsm_host>/api/services/configure/system/id/callsign"

Request Body: XML

Request Example

Following is the XML body for the POST request to upload an individual system level callsign setup:

"<?xml version="1.0" encoding="UTF-8"?>
<CallSignSetup xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><CallSign
Name='Chan1467457835' IPAddress='232.2.1.4' Port='6000'></CallSign><CallSign
Name='Chan1467459443' IPAddress='232.2.1.4' Port='6001'></CallSign></CallSignSetup>"

System Ingest Driver Server

The System Ingest Driver Server list request-response message returns Ingest Driver Server configuration.

GET ALL System Ingest Driver Server Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl http://<cdsm_host>/api/services/configure/system/id/ingestdriverserverconfig

Response

If the request succeeds, the list element is returned in the XML body response containing Ingest Driver Server configuration.

Response Example

"<?xml version="1.0" encoding="UTF-8"?>
<IngestDriverServerSettings xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><IngestDriverServer
Port='50000' NoOfThreads='50'></IngestDriverServer></IngestDriverServerSettings>"
POST System Ingest Driver Server Configuration

Following are the request components:

Request Type: HTTP POST

Request

curl --form upload=@/root/ingestdriver.xml --form press=OK
'http://<cdsm_host>/api/services/configure/system/id/ingestdriver'

Request Body: XML

Request Example

Following is the XML body for the POST request to upload an individual system level Ingest Driver Server.

<?xml version="1.0" encoding="UTF-8"?><IngestDriverServerSettings
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><IngestDriverServer
Port='50000' NoOfThreads='50'></IngestDriverServer></IngestDriverServerSettings>

System Logging

The System Logging list request-response message returns logging configuration done in the system level.

GET ALL System Logging Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl http://<cdsm_host>/api/services/configure/system/id/logging

Response

If the request succeeds, the list element is returned in the XML body response containing the logging configuration done in the system level.

Response Example

<?xml version="1.0" encoding="UTF-8"?><LogConfig
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><Facility name="aim"
local-log="notice" remote-log="DISABLE"><debug-flags><flag
name="general"/></debug-flags></Facility></LogConfig>
POST System Logging Configuration

Following are the request components:

Request Type: HTTP POST

Request

curl --form upload=@/root/logging.xml --form press=OK "http://<cdsm_host>/api/services/configure/system/id/logging"

Request Body: XML

Request Example

Following is the XML body for the POST request to upload an individual system level System logging:

```xml
<?xml version="1.0" encoding="UTF-8"?><LogConfig xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><Facility name="aim" local-log="notice" remote-log="DISABLE"><debug-flags><flag name="general"/></debug-flags></Facility></LogConfig>
```

System Syslog

The System Syslog list request-response message returns Syslog configuration for the System level.

GET ALL System Syslog Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl http://<cdsm_host>/api/services/configure/system/id/syslog

Response

If the request succeeds, the list element is returned in the XML body response containing the Syslog configuration for the System level.

Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?><SyslogConfig xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><SyslogConfigDetails RemoteLogging='Disable'/></SyslogConfigDetails></SyslogConfig>
```

POST System Syslog Configuration

Following are the request components:

Request Type: HTTP POST

Request

curl --form upload=@/root/syslog.xml --form press=OK "http://<cdsm_host>/api/services/configure/system/id/syslog"
Request Body: XML

Request Example
Following is the XML body for the POST request to upload an individual Syslog configuration:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<SyslogConfig
    xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"
    <SyslogConfigDetails
        RemoteLoggings='Disable'></SyslogConfigDetails></SyslogConfig>
```

System VBO Setup

The System VBO Setup list request-response message returns VBO Setup configuration.

GET ALL System VBO Setup Configuration

Following are the request components:

Request Type: HTTP GET

Request
curl http://<cdsm_host>/api/services/configure/system/id/vbo

Response
If the request succeeds, the list element is returned in the XML body response containing the VBO Setup configuration.

Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<VBOList
    xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"
    <VBO VHO="20003"
        IPAddress="10.197.103.138" Port="3400" ReceiveTimeOut="2"
        VBOIMNameSpace="urn:com:cisco:vbo:ingest" VBOIMInterfaceURI="/contentserver/"
    ></VBO></VBOList>
```

POST System VBO Setup Configuration

Following are the request components:

Request Type: HTTP POST

Request
curl --form upload=@/root/vbosetup.xml --form press=OK
"http://<cdsm_host>/api/services/configure/system/id/vbosetup"
System Backup Configurations

The System Backup Configuration list request-response message returns a list of all System level configurations.

GET ALL System Backup Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl http://<cdsm_host>/api/services/configure/system/id/backup

Response

If the request succeeds, the list element is returned in the XML body response containing the a list of all System level configurations.

Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<SystemConfiguration
 xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
<DNS
 xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
<DomainSuffix>hcl</DomainSuffix>
</DNS>
<NTP
 xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
<NTPServer>10.197.86.134</NTPServer>
</NTP>
<HostServiceList
 xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
<ArrayNameConfig
 xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
<ArrayName
 ArrayID='2000' Name='Array_2000' Vendor='ISA-OpenStream'></ArrayName>
</ArrayNameConfig>
<VBOList
 xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
<VBO VHO="20001" DomainName="VHO1" IPAddress="10.197.103.233" Port="6100" ReceiveTimeOut="2"
 ODRMNotifyUrl="/odrm/OdrmApiR2/EventNotify" ODRMNameSpace="urn:net:beaumaris:acR2:xsd:1"
 SetupNameSpace="urn:net:beaumaris:common:xsd:1" SetupInterfaceUrl="/streamingserver/"
 ></VBO>
</VBOList>
<MPEGTuning
 xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
<MPEGTuningSettings
 PTSNPTMode='File' C2ATimeout='2000' PauseBehavior='Only NULL packets' FFResume='Disable'
 RWSkip='Disable' FwdJump='Disable' SkippingStoneMode='Disable' DualCAS='Disable' />
</MPEGTuning>
<IPNickNames
 xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
<IPNickName
 IPAddress='10.197.102.130' NickName='Server-130'></IPNickName>
</IPNickNames>
<LogConfig
 xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
<Facility name="aim"
 local-log="notice" remote-log="DISABLE" debug-flags><flag
 name="general"/></debug-flags></Facility></LogConfig></SystemConfiguration>
```
POST System Backup Configuration

Following are the request components:

**Request Type: HTTP POST**

**Request**

curl --form upload=@/root/restore.xml --form press=OK
"http://<cdsm_host>/api/services/configure/system/id/restore"

**Request Body: XML**

**Request Example**

Following is the XML body for the POST request to upload an individual system level backup configurations:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<SystemConfiguration
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"/>
</SystemConfiguration>
```

Array Level Configuration API

The Array Level Configuration APIs offers the ability to download or upload an XML file that contains the array level configuration settings by way of an API call using any REpresentational State Transfer (REST) client.

Following are the Array level configuration APIs:

- **Array Level DNS**, page 11-34
- **Array Level NTP Server**, page 11-35
- **Streamer for BMS Connectivity Configuration**, page 11-36
- **Vault for BMS Connectivity Configuration**, page 11-37
Array Level DNS

The following is the array level DNS configurations in the system.

GET ALL Array DNS Configuration

Following are the request components:

**Request Type: HTTP GET**

**Request**

curl http://<cdsm_host>/api/services/configure/array/id/dns

**Response**

If the request succeeds, the list element is returned in the XML body response containing the list of array level DNS.
Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?><ArrayDNSConfig xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><ArrayDNS GroupID='1991'><DomainSuffix>new1</DomainSuffix><DNSServer>10.197.86.135</DNSServer></ArrayDNS></ArrayDNSConfig>
```

**POST Array DNS Configuration**

Following are the request components:

**Request Type: HTTP POST**

**Request**

```
curl --form upload=@/root/dns.xml --form press=OK "http://<cdsm_host>/api/services/configure/array/id/dns"
```

**Request Body: XML**

**Request Example**

Following is the XML body for the POST request to upload an individual array DNS configuration settings:

```xml
<?xml version="1.0" encoding="UTF-8"?><ArrayDNSConfig xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><ArrayDNS GroupID='1991'><DomainSuffix>new1</DomainSuffix><DNSServer>10.197.86.135</DNSServer></ArrayDNS></ArrayDNSConfig>
```

**Array Level NTP Server**

The following is the array level NTP configurations in the system.

**GET ALL Array NTP Configuration**

Following are the request components:

**Request Type: HTTP GET**

**Request**

```
curl http://<cdsm_host>/api/services/configure/array/id/ntp
```

**Response**

If the request succeeds, the list element is returned in the XML body response containing the list of array level NTP.

**Response Example**

```xml
<?xml version="1.0" encoding="UTF-8"?><ArrayNTPConfig xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><ArrayNTP GroupID='1991'><NTPServerIP>10.197.86.139</NTPServerIP></ArrayNTP></ArrayNTPConfig>
```
POST Array NTP Configuration

Following are the request components:

**Request Type:** HTTP POST

**Request**

curl --form upload=@/root/ntp.xml --form press=OK
"http://<cdsm_host>/api/services/configure/array/id/ntp"

**Request Body:** XML

**Request Example**

Following is the XML body for the POST request to upload an individual array NTP configuration settings:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ArrayNTPConfig xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
  <ArrayNTP GroupID='1991'>
    <NTPServerIP>10.197.86.139</NTPServerIP>
  </ArrayNTP>
</ArrayNTPConfig>
```

Streamer for BMS Connectivity Configuration

The following is the streamer for BMS configurations in the system.

GET ALL Streamer BMS Configuration

Following are the request components:

**Request Type:** HTTP GET

**Request**

curl http://<cdsm_host>/api/services/configure/array/id/streamerbms

**Response**

If the request succeeds, the list element is returned in the XML body response containing the Streamer for BMS connectivity configurations.

**Response Example**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<StreamerBMSConfig xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
  <OpenStreamSettings StreamMasterIP='10.197.103.225' Port='3300' HeadendID='0.0.0.0' StreamSourceConfig='Control IP' WebServicePort='8080' StreamingMode='GigE' MSASupport='Disabled' TME-SCE='Disabled'/>
  <CORBAServices NameServicePort='2000' NotifyServicePort='2010' NotifyServiceFactory='NotifyEventChannelFactory'/>
  <CORBAEventChannels EventChannelID='EventChannels' EventChannelKind='Context' StreamChannelID='StreamChannel' StreamChannelKind='Factory' FactoriesID='Factories' FactoriesKind='Context' EventChannelFactory='NotifyEventChannelFactory' LoadQueryInterval='3'/>
  <LSCPService StreamServiceID='ArroyoStreamService' StreamServiceKind='Factory' MasterNoOfThreads='16' PlayNoOfThreads='34' LSCPPort='9000' LSCPResponsePad='Disabled' LSCPClientProtocol='TVGuide' AdvancedISASettings='Disabled'/>
</StreamerBMSConfig>
```
POST Streamer BMS Configuration

Following are the request components:

**Request Type:** HTTP POST

**Request**

curl --form upload=@/root/streamerbms.xml --form press=OK
"http://<cdsm_host>/api/services/configure/array/id/streamerbms"

**Request Body:** XML

**Request Example**

Following is the XML body for the POST request to upload an individual streamer for BMS connectivity configuration:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<StreamerBMSConfig
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
<OpenStreamSettings
StreamMasterIP='10.197.103.225' Port='3300' HeadendID='0.0.0.0'
StreamSourceConfig='Control IP' WebServicePort='8080' StreamingMode='GigE'
MSASupport='Disabled' TME-SCE='Disabled'/>
<CORBAServices NameServicePort='2000'
NotifyServicePort='2010'
NotifyServiceFactory='NotifyEventChannelFactory'/>
<CORBAEventChannels
EventChannelID='EventChannels' EventChannelKind='Context'
StreamChannelID='StreamChannel'
StreamChannelKind='Factory'
FactoriesID='Factories'
FactoriesKind='Context'
EventChannelFactory='NotifyEventChannelFactory'
LoadQueryInterval='3'/>
<LSCPService
StreamServiceID='ArroyoStreamService'
StreamServiceKind='Factory'
MasterNoOfThreads='16'
PlayNoOfThreads='34'
LSCPPort='9000'
LSCPResponsePad='Disabled'
LSCPClientProtocol='TVGuide'
AdvancedISASettings='Disabled'/>
<OpenStreamResourceServices
ServiceName='ArroyoResourceManager'
ServicePollTime='3600'
StreamTimeout='80'
StreamSourcePort='8999'/>
<SessionGateways
SessionGateway="N2BBSessionGateway.Factory"/>
</StreamerBMSConfig>
```

Vault for BMS Connectivity Configuration

The following is the vault for BMS configurations in the system.

GET ALL Vault BMS Configuration

Following are the request components:

**Request Type:** HTTP GET

**Request**

curl http://<cdsm_host>/api/services/configure/array/id/vaultbms
Response

If the request succeeds, the list element is returned in the XML body response containing the Vault for BMS connectivity configurations.

Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<VaultBMSConfig
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
<OpenStreamSettings
ContentServiceMasterIP='10.197.86.177' Port='3200' HeadendID='0.0.0.0'
WebServicePort='8080' FTPOutPort='5500'/>
<ContentService
ContentStoreName='ArroyoVideoContentStore-NOS' ContentStoreKind='Factory'
ContentFactoryID='ArroyoContentStoreFactory-NOS' ContentFactoryKind='Factory'
ContentNoOfThreads='32' PreEncryptionSupport='Disabled' FTPClientPort='21'
FTPServerPort='4000' FTPNoOfAttempts='1' FTPTimeout='360000000'/>
<CORBAServices
NameServiceIP='10.197.86.177' NameServicePort='5000' NotifyServiceIP='10.197.86.177'
NotifyServicePort='5010' NotifyServiceFactory='NotifyEventChannelFactory'/>
</VaultBMSConfig>
```

POST Vault BMS Configuration

Following are the request components:

**Request Type:** HTTP POST

**Request**

```
curl --form upload=@/root/vaultbms.xml --form press=OK
"http://<cdsm_host>/api/services/configure/array/id/vaultbms"
```

**Request Body:** XML

**Request Example**

Following is the XML body for the POST request to upload an individual Vault for BMS connectivity configuration:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<VaultBMSConfig
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
<OpenStreamSettings
ContentServiceMasterIP='10.197.86.177' Port='3200' HeadendID='0.0.0.0'
WebServicePort='8080' FTPOutPort='5500'/>
<ContentService
ContentStoreName='ArroyoVideoContentStore-NOS' ContentStoreKind='Factory'
ContentFactoryID='ArroyoContentStoreFactory-NOS' ContentFactoryKind='Factory'
ContentNoOfThreads='32' PreEncryptionSupport='Disabled' FTPClientPort='21'
FTPServerPort='4000' FTPNoOfAttempts='1' FTPTimeout='360000000'/>
<CORBAServices
NameServiceIP='10.197.86.177' NameServicePort='5000' NotifyServiceIP='10.197.86.177'
NotifyServicePort='5010' NotifyServiceFactory='NotifyEventChannelFactory'/>
</VaultBMSConfig>
```

Stream Groups Setup

The following is the stream groups setup configurations in the system.
GET ALL Stream Groups Setup Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl http://<cdsm_host>/api/services/configure/array/id/streamgroupsetup

Response
If the request succeeds, the list element is returned in the XML body response containing the list of stream groups setup configurations.

Response Example


POST Stream Groups Setup Configuration

Following are the request components:

Request Type: HTTP POST

Request

curl --form upload=@/root/streamgroupsetup.xml --form press=OK
'http://<cdsm_host>/api/services/configure/array/id/streamgroupsetup'

Request Body: XML

Request Example

Following is the XML body for the POST request to upload an individual stream groups setup configuration:

SSV Groups Setup

The following is the SSV groups setup configurations in the system.

GET ALL SSV Groups Setup Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl http://<cdsm_host>/api/services/configure/array/id/ssvgroupssetup

Response

If the request succeeds, the list element is returned in the XML body response containing the list of SSV groups setup configurations.

Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?><SSVGroupsConfigList
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><SSVGroupConfig
GroupName='SSV-GROUP1' FadeFrameSupport='Enabled'><SSVGroupMembers><Server
IPAddress='10.197.103.223' ServerID='223' GroupID='2000'/><Server
IPAddress='10.197.103.224' ServerID='224' GroupID='2000'/></SSVGroupMembers></SSVGroupConfig></SSVGroupsConfigList>
```

POST SSV Groups Setup Configuration

Following are the request components:

Request Type: HTTP POST

Request

curl --form upload=@/root/ssvgroupssetup.xml --form press=OK
"http://<cdsm_host>/api/services/configure/array/id/ssvgroupssetup"

Request Body: XML

Request Example

Following is the XML body for the POST request to upload an individual SSV groups setup configurations:

```xml
<?xml version="1.0" encoding="UTF-8"?><SSVGroupsConfigList
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><SSVGroupConfig
GroupName='SSV-GROUP1' FadeFrameSupport='Enabled'><SSVGroupMembers><Server
IPAddress='10.197.103.223' ServerID='223' GroupID='2000'/><Server
IPAddress='10.197.103.224' ServerID='224' GroupID='2000'/></SSVGroupMembers></SSVGroupConfig></SSVGroupsConfigList>
```

VHO Setup

The following is the VHO setup configurations in the system.
GET ALL VHO Setup Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl http://<cdsm_host>/api/services/configure/array/id/vhosetup

Response

If the request succeeds, the list element is returned in the XML body response containing the list of array level VHO setup.

Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<VHOGroupsConfigList
    xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
    <VHOGroupConfig
        GroupName='VHO1'>
        <VHOGroupMembers>
            <StreamGroup Name='SetupControl'/>
            <StreamGroup Name='PlayGroup'/>
        </VHOGroupMembers>
    </VHOGroupConfig>
    <VHOGroupConfig
        GroupName='VHO2'>
        <VHOGroupMembers>
            <StreamGroup Name='SetupControl'/>
            <StreamGroup Name='PlayGroup'/>
        </VHOGroupMembers>
    </VHOGroupConfig>
</VHOGroupsConfigList>
```

POST VHO Setup Configuration

Following are the request components:

Request Type: HTTP POST

Request

```
curl --form upload=@/root/vhosetup.xml --form press=OK
    "http://<cdsm_host>/api/services/configure/array/id/vhosetup"
```

Request Body: XML

Request Example

Following is the XML body for the POST request to upload an individual VHO setup:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<VHOGroupsConfigList
    xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
    <VHOGroupConfig
        GroupName='VHO1'>
        <VHOGroupMembers>
            <StreamGroup Name='SetupControl'/>
            <StreamGroup Name='PlayGroup'/>
        </VHOGroupMembers>
    </VHOGroupConfig>
    <VHOGroupConfig
        GroupName='VHO2'>
        <VHOGroupMembers>
            <StreamGroup Name='SetupControl'/>
            <StreamGroup Name='PlayGroup'/>
        </VHOGroupMembers>
    </VHOGroupConfig>
</VHOGroupsConfigList>
```

Vault Groups Setup

The following is the vault groups setup configurations in the system.
GET ALL Vault Groups Setup Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl http://<cdsm_host>/api/services/configure/array/id/vaultgroupssetup

Response

If the request succeeds, the list element is returned in the XML body response containing the list of Vault
groups setup configurations.

Response Example

<?xml version="1.0" encoding="UTF-8"?><VaultGroupsConfigList
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><VaultGroupConfig
GroupName='SHE1'><VaultGroupMembers><Server IPAddress='10.197.103.53' ServerID='554'
GroupID='1991'/></VaultGroupMembers></VaultGroupConfig><VaultGroupConfig
GroupName='SHE2'><VaultGroupMembers><Server IPAddress='10.197.103.54' ServerID='555'
GroupID='1991'/></VaultGroupMembers></VaultGroupConfig></VaultGroupsConfigList>

POST Vault Groups Setup Configuration

Following are the request components:

Request Type: HTTP POST

Request

curl --form upload=@/root/vaultgroupssetup.xml --form press=OK
"http://<cdsm_host>/api/services/configure/array/id/vaultgroupssetup"

Request Body: XML

Request Example

Following is the XML body for the POST request to upload an individual Vault groups setup:

<?xml version="1.0" encoding="UTF-8"?><VaultGroupsConfigList
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><VaultGroupConfig
GroupName='SHE1'><VaultGroupMembers><Server IPAddress='10.197.103.53' ServerID='554'
GroupID='1991'/></VaultGroupMembers></VaultGroupConfig><VaultGroupConfig
GroupName='SHE2'><VaultGroupMembers><Server IPAddress='10.197.103.54' ServerID='555'
GroupID='1991'/></VaultGroupMembers></VaultGroupConfig></VaultGroupsConfigList>

Ingest Steering

The following is the Ingest steering configurations in the system.
GET ALL Ingest Steering Configuration

Following are the request components:

Request Type: HTTP GET

Request
curl http://<cdsm_host>/api/services/configure/array/id/ingeststeering

Response
If the request succeeds, the list element is returned in the XML body response containing the Ingest Steering configurations.

Response Example
<?xml version="1.0" encoding="UTF-8"?><IngestSteeringConfig xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><VaultGroupList><VaultGroup Name='SHE1'/><VaultGroup Name='SHE2'/></VaultGroupList></IngestSteeringConfig>

POST Ingest Steering Configuration

Following are the request components:

Request Type: HTTP POST

Request
curl --form upload=@/root/ingeststeering.xml --form press=OK "http://<cdsm_host>/api/services/configure/array/id/ingeststeering"

Request Body: XML

Request Example
Following is the XML body for the POST request to upload an individual Ingest steering configuration settings:

<?xml version="1.0" encoding="UTF-8"?><IngestSteeringConfig xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><VaultGroupList><VaultGroup Name='SHE1'/><VaultGroup Name='SHE2'/></VaultGroupList></IngestSteeringConfig>

Cache Groups Setup

The following is the cache groups setup configurations in the system.

GET ALL Cache Groups Setup Configuration

Following are the request components:

Request Type: HTTP GET

Request
curl http://<cdsm_host>/api/services/configure/array/id/cachegroupssetup
Response
If the request succeeds, the list element is returned in the XML body response containing the list of Cache groups setup configurations.

Response Example
<?xml version="1.0" encoding="UTF-8"?>
<CacheGroupsConfigList
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
<CacheGroupConfig
GroupName='CG1'>
<CacheGroupMembers>
<Server IPAddress='10.197.103.137' ServerID='556'
GroupID='1991'/>
</CacheGroupMembers>
</CacheGroupConfig>
<CacheGroupConfig
GroupName='TestCG1'/>
</CacheGroupsConfigList>

POST Cache Groups Setup Configuration

Following are the request components:

Request Type: HTTP POST

Request
curl --form upload=@/root/cachegroupssetup.xml --form press=OK
"http://<cdsm_host>/api/services/configure/array/id/cachegroupssetup"

Request Body: XML

Request Example
Following is the XML body for the POST request to upload an individual Cache groups setup configuration settings:
<?xml version="1.0" encoding="UTF-8"?>
<CacheGroupsConfigList
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
<CacheGroupConfig
GroupName='CG1'>
<CacheGroupMembers>
<Server IPAddress='10.197.103.137' ServerID='556'
GroupID='1991'/>
</CacheGroupMembers>
</CacheGroupConfig>
<CacheGroupConfig
GroupName='TestCG1'/>
</CacheGroupsConfigList>

Mapping Cache Groups to Cache Groups

The following is the mapping of cache groups to cache groups configurations in the system.

GET ALL Mapping Cache Groups to Cache Groups Configuration

Following are the request components:

Request Type: HTTP GET

Request
curl http://<cdsm_host>/api/services/configure/array/id/cachetocachemap

Response
If the request succeeds, the list element is returned in the XML body response containing the list of mapping of cache groups to cache groups.

Response Example
POST Mapping Cache Groups to Cache Groups Configuration

Following are the request components:

**Request Type: HTTP POST**

Request

curl --form upload=@/root/cachetocachemap.xml --form press=OK
"http://<cdsm_host>/api/services/configure/array/id/cachetocachemap"

**Request Body: XML**

**Request Example**

Following is the XML body for the POST request to upload an individual mapping Cache groups to Cache groups configuration settings:

```xml
<?xml version="1.0" encoding="UTF-8"?><CacheToCacheMapConfig
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><CacheGroupPreferences
GroupName='CG1'><CacheGroupPreference GroupName='TestCG1'
Preference='None'/></CacheGroupPreferences><CacheGroupPreferences
GroupName='TestCG1'><CacheGroupPreference GroupName='CG1'
Preference='None'/></CacheGroupPreferences></CacheToCacheMapConfig>
```

Mapping Cache Groups to Vault Groups

The following is the mapping of cache groups to vault groups configurations in the system.

GET ALL Mapping Cache Groups to Vault Groups Configuration

Following are the request components:

**Request Type: HTTP GET**

Request

curl http://<cdsm_host>/api/services/configure/array/id/cachetovaultmap

**Response**

If the request succeeds, the list element is returned in the XML body response containing the list of mapping of cache groups to vault groups.

**Response Example**

```xml
<?xml version="1.0" encoding="UTF-8"?><CacheToVaultMapConfig
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><CacheGroupPreferences
GroupName='CG1'><VaultGroupPreference GroupName='SHE1'
Preference='None'/></CacheGroupPreferences><CacheGroupPreferences
GroupName='SHE1'><VaultGroupPreference GroupName='CG1'
Preference='None'/></CacheGroupPreferences></CacheToVaultMapConfig>
```
POST Mapping Cache Groups to Vault Groups Configuration

Following are the request components:

**Request Type: HTTP POST**

**Request**

curl --form upload=@/root/cachetovaultmap.xml --form press=OK
"http://<cdsm_host>/api/services/configure/array/id/cachetovaultmap"

**Request Body: XML**

**Request Example**

Following is the XML body for the POST request to upload an individual mapping Cache groups to Vault
groups configuration settings:

```xml
<?xml version="1.0" encoding="UTF-8"?><CacheToVaultMapConfig xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><CacheGroupPreferences GroupName='CG1'><VaultGroupPreference GroupName='SHE1' Preference='High'/><VaultGroupPreference GroupName='SHE2' Preference='High'/></CacheGroupPreferences><CacheGroupPreferences GroupName='TestCG1'><VaultGroupPreference GroupName='SHE1' Preference='High'/><VaultGroupPreference GroupName='SHE2' Preference='High'/></CacheGroupPreferences></CacheToVaultMapConfig>
```

**D5 Interface Settings**

The following is the D5 Interface configurations in the system.

**GET ALL D5 Interface Configuration**

Following are the request components:

**Request Type: HTTP GET**

**Request**

curl http://<cdsm_host>/api/services/configure/array/id/d5setup

**Response**

If the request succeeds, the list element is returned in the XML body response containing the list of
mapping of D5 Interface.

**Response Example**
POST D5 Interface Configuration

Following are the request components:

**Request Type:** HTTP POST

**Request**

```
curl --form upload=@/root/d5setup.xml --form press=OK
'http://<cdsm_host>/api/services/configure/array/id/d5setup'
```

**Request Body:** XML

**Request Example**

Following is the XML body for the POST request to upload an individual D5 Interface configuration settings:

```xml
<?xml version="1.0" encoding="UTF-8"?><D5SetupConfigList
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><D5SetupConfig
GroupName='SG2' ModelName='' SignificantResourceUsage='10' ODRMUrl=''
NotifyInterval=''><SOPConfigList><SOPConfig Name='SOP1'
LoopbackAddress=''/></SOPConfigList></D5SetupConfig><D5SetupConfig GroupName='SG'
ModelName='' SignificantResourceUsage='10' ODRMUrl=''
NotifyInterval=''><SOPConfigList><SOPConfig Name='SOP1'
LoopbackAddress=''/></SOPConfigList></D5SetupConfig></D5SetupConfigList>
```

Locating Cache Groups

The following is the cache group locator configurations in the system.

GET ALL Cache Group Locator Configuration

Following are the request components:

**Request Type:** HTTP GET

**Request**

```
curl http://<cdsm_host>/api/services/configure/array/id/cachegrouplocator
```

**Response**

If the request succeeds, the list element is returned in the XML body response containing the list of mapping of cache group locator.

**Response Example**

```xml
<?xml version="1.0" encoding="UTF-8"?><D5SetupConfigList
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><D5SetupConfig
GroupName='SG2' ModelName='' SignificantResourceUsage='10' ODRMUrl=''
NotifyInterval=''><SOPConfigList><SOPConfig Name='SOP1'
LoopbackAddress=''/></SOPConfigList></D5SetupConfig><D5SetupConfig GroupName='SG'
ModelName='' SignificantResourceUsage='10' ODRMUrl=''
NotifyInterval=''><SOPConfigList><SOPConfig Name='SOP1'
LoopbackAddress=''/></SOPConfigList></D5SetupConfig></D5SetupConfigList>
```
POST Cache Group Locator Configuration

Following are the request components:

Request Type: HTTP POST

Request

curl --form upload=@/root/cachegrouplocator.xml --form press=OK
"http://<cdsm_host>/api/services/configure/array/id/cachegrouplocator"

Request Body: XML

Request Example

Following is the XML body for the POST request to upload an individual cache group locator configuration settings:

```xml
<?xml version="1.0" encoding="UTF-8"?><CDNGroupsConfigList
xmlns='http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs'>
<CDNGroupConfig
GroupName='CG1' LocationVirtualIP='192.165.100.240'
LocationSubnetMask='255.255.255.0'></CDNGroupConfig></CDNGroupsConfigList>
```

Locating CDN Groups

The following is the CDN group locator configurations in the system.

GET ALL CDN Group Locator Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl http://<cdsm_host>/api/services/configure/array/id/cdngrouplocator

Response

If the request succeeds, the list element is returned in the XML body response containing the list of mapping of CDN group locator.

Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?><CDNGroupsConfigList
xmlns='http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs'>
<CDNGroupConfig
GroupName='CG1' LocationVirtualIP='192.165.100.240'
LocationSubnetMask='255.255.255.0'></CDNGroupConfig></CDNGroupsConfigList>
```
POST CDN Group Locator Configuration

Following are the request components:

Request Type: HTTP POST

Request

curl --form upload=@/root/cdngrouplocator.xml --form press=OK
'\http://<cdsm_host>/api/services/configure/array/id/cdngrouplocator'

Request Body: XML

Request Example

Following is the XML body for the POST request to upload an individual CDN group locator configuration settings:

<?xml version="1.0" encoding="UTF-8"?><CDNGroupsConfigList
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><CDNGroupConfig
GroupName='CG1' LocationVirtualIP='192.165.100.240'
LocationSubnetMask='255.255.255.0'></CDNGroupConfig></CDNGroupsConfigList>

Mapping Stream to Cache

The following is the mapping of stream to cache configurations in the system.

GET ALL Mapping Stream to Cache Configurations

Following are the request components:

Request Type: HTTP GET

Request

curl http://<cdsm_host>/api/services/configure/array/id/streamtocachemap

Response

If the request succeeds, the list element is returned in the XML body response containing the list of mapping stream to cache.

Response Example

<?xml version="1.0" encoding="UTF-8"?><StreamToCacheMapConfig
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><StreamGroupPreferences
GroupName='SetupControl'><CacheGroupPreference GroupName='CG1'
Preference='High'/><CacheGroupPreference GroupName='TestCG1'
Preference='None'/><VaultGroupPreference GroupName='SHE1'
Preference='None'/><VaultGroupPreference GroupName='SHE2'
Preference='None'/></StreamGroupPreferences><StreamGroupPreferences
GroupName='PlayGroup'><CacheGroupPreference GroupName='CG1'
Preference='High'/><CacheGroupPreference GroupName='TestCG1'
Preference='None'/><VaultGroupPreference GroupName='SHE1'
Preference='None'/><VaultGroupPreference GroupName='SHE2'
Preference='None'/></StreamGroupPreferences></StreamToCacheMapConfig>
POST Mapping Stream to Cache Configuration

Following are the request components:

**Request Type: HTTP POST**

**Request**

curl --form upload=@/root/streamtocachemap.xml --form press=OK
"http://<cdsm_host>/api/services/configure/array/id/streamtocachemap"

**Request Body: XML**

**Request Example**

Following is the XML body for the POST request to upload an individual mapping of stream to cache configuration settings:

```xml
<?xml version="1.0" encoding="UTF-8"?><StreamToCacheMapConfig
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><StreamGroupPreferences
GroupName='SetupControl'><CacheGroupPreference GroupName='CG1'
Preference='High'/><CacheGroupPreference GroupName='TestCG1'
Preference='None'/><VaultGroupPreference GroupName='SHE1'
Preference='None'/><VaultGroupPreference GroupName='SHE2'
Preference='None'/></StreamGroupPreferences><StreamGroupPreferences
GroupName='PlayGroup'><CacheGroupPreference GroupName='CG1'
Preference='High'/><CacheGroupPreference GroupName='TestCG1'
Preference='None'/><VaultGroupPreference GroupName='SHE1'
Preference='None'/><VaultGroupPreference GroupName='SHE2'
Preference='None'/></StreamGroupPreferences></StreamToCacheMapConfig>
```

Mapping Stream to CDN

The following is the mapping of stream to CDN configurations in the system.

GET ALL Mapping Stream to CDN Configurations

Following are the request components:

**Request Type: HTTP GET**

**Request**

curl http://<cdsm_host>/api/services/configure/array/id/streamtocdnmap

**Response**

If the request succeeds, the list element is returned in the XML body response containing the list of mapping stream to CDN.

**Response Example**

```xml
<?xml version="1.0" encoding="UTF-8"?><StreamToCDNMapConfig
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><StreamToCDNGroupConfig
GroupName='SG2'  DefaultCDNGroupName='VG1'><VolumePreferences Name='X-Men'><CDNGroup
GroupName='CG1'
Preference='High'/></VolumePreferences></StreamToCDNGroupConfig></StreamToCDNMapConfig>
```
POST Mapping Stream to CDN Configuration

Following are the request components:

Request Type: HTTP POST

Request

curl --form upload=@/root/streamtocdnmap.xml --form press=OK
"http://<cdsm_host>/api/services/configure/array/id/streamtocdnmap"

Request Body: XML

Request Example

Following is the XML body for the POST request to upload an individual mapping of stream to CDN:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<StreamToCDNMapConfig
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
<StreamToCDNGroupConfig
GroupName='SG2'  DefaultCDNGroupName='VG1'><VolumePreferences Name='X-Men'><CDNGroup
GroupName='CG1'
Preference='High'/></VolumePreferences></StreamToCDNGroupConfig></StreamToCDNMapConfig>
```

Mapping Vault Groups for Redundancy

The following is the mapping vault groups for redundancy configurations in the system.

GET ALL Mapping Vault Groups for Redundancy Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl http://<cdsm_host>/api/services/configure/array/id/vaultredundancymap

Response

If the request succeeds, the list element is returned in the XML body response containing the list of mapping vault groups.

Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<VaultRedundancyMapConfig
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
<VaultRedundancyMap
GroupName='SHE1'><VaultGroup Name='SHE2'
Mirror='Yes'/></VaultRedundancyMap>
<VaultRedundancyMap
GroupName='SHE2'><VaultGroup Name='SHE1'
Mirror='No'/></VaultRedundancyMapConfig>
```
**Array Level Configuration API**

**POST Mapping Vault Groups for Redundancy Configuration**

Following are the request components:

**Request Type:** HTTP POST

**Request**

curl --form upload=@/root/vaultredundancymap.xml --form press=OK "http://<cdsm_host>/api/services/configure/array/id/vaultredundancymap"

**Request Body:** XML

**Request Example**

Following is the XML body for the POST request to upload an individual mapping vault groups for redundancy:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<VaultRedundancyMapConfig xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
  <VaultRedundancyMap GroupName='SHE1'>
    <VaultGroup Name='SHE2' Mirror='Yes'/>
  </VaultRedundancyMap>
  <VaultRedundancyMap GroupName='SHE2'>
    <VaultGroup Name='SHE1' Mirror='No'/>
  </VaultRedundancyMap>
</VaultRedundancyMapConfig>
```

**Master Vault Group**

The following is the master vault group configurations in the system.

**GET ALL Master Vault Group Configuration**

Following are the request components:

**Request Type:** HTTP GET

**Request**

curl http://<cdsm_host>/api/services/configure/array/id/mastervaultgroup

**Response**

If the request succeeds, the list element is returned in the XML body response containing the list of master vault groups.

**Response Example**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<MasterVaultGroupConfig xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
  <GroupConfig GroupName='SHE1' ActAsMaster='Yes'/>
  <GroupConfig GroupName='SHE2' ActAsMaster='No'/>
</MasterVaultGroupConfig>
```
**POST Master Vault Group Configuration**

Following are the request components:

**Request Type:** HTTP POST

**Request**

```bash
curl --form upload=@/root/mastervaultgroup.xml --form press=OK
'http://<cdsm_host>/api/services/configure/array/id/mastervaultgroup'
```

**Request Body:** XML

**Request Example**

Following is the XML body for the POST request to upload an individual master vault group:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<MasterVaultGroupConfig xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
  <GroupConfig GroupName='SHE1' ActAsMaster='Yes'/>
  <GroupConfig GroupName='SHE2' ActAsMaster='No'/>
</MasterVaultGroupConfig>
```

**Control and Setup IPs**

The following is the control and setup IPs configurations in the system.

**GET ALL Control and Setup IPs Configuration**

Following are the request components:

**Request Type:** HTTP GET

**Request**

```bash
curl http://<cdsm_host>/api/services/configure/array/id/controlsetupip
```

**Response**

If the request succeeds, the list element is returned in the XML body response containing the list of control and setup IPs.

**Response Example**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ControlSetupIPConfigList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
  <ControlSetupIPConfig GroupName='SG2' IPAddress='10.197.102.160' SubnetMask='255.255.255.0' IPType='Control/Setup IP' SetupID='1'/>
  <ControlSetupIPConfig GroupName='SG' IPAddress='10.197.102.31' SubnetMask='255.255.255.0' IPType='Control/Setup IP' SetupID='3'/>
</ControlSetupIPConfigList>
```
POST Control and Setup IPs Configuration

Following are the request components:

Request Type: HTTP POST

Request

curl --form upload=@/root/controlsetupip.xml --form press=OK "http://<cdsm_host>/api/services/configure/array/id/controlsetupip"

Request Body: XML

Request Example

Following is the XML body for the POST request to upload an individual control and setup IPs:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ControlSetupIPConfigList
    xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
    <ControlSetupIPConfig GroupName='SG2' IPAddress='10.197.102.160' SubnetMask='255.255.255.0' IPType='Control/Setup IP' SetupID='1'/>
    <ControlSetupIPConfig GroupName='SG' IPAddress='10.197.102.31' SubnetMask='255.255.255.0' IPType='Control/Setup IP' SetupID='3'/>
</ControlSetupIPConfigList>
```

VHO ISA Setup

The following is the VHO ISA setup configurations in the system.

GET ALL VHO ISA Setup Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl http://<cdsm_host>/api/services/configure/array/id/vhoisasetup

Response

If the request succeeds, the list element is returned in the XML body response containing the list of VHO ISA setup.

Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<VHOISAConfigurationSettings
    xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
    <VHOISAConfig GroupName='VHO1'>
        <StreamerBMSSettings StreamMasterPort='3300' HeadendID='0.0.0.0' StreamSourceConfig='Control IP'/>
        <CORBAServices NameServiceIP='10.197.103.9' NameServicePort='2000' NotifyServiceIP='10.197.103.9' NotifyServicePort='2010' NotifyServiceFactory='NotifyEventChannelFactory'/>
        <LSCPService StreamServiceID='ArroyoStreamService' StreamServiceKind='Factory' StreamChannelID='StreamChannel' StreamChannelKind='Factory' MasterNoOfThreads='32' PlayNoOfThreads='34' LSCPPort='9000' LSCPResponsePad='Disabled' LSCPClientProtocol='Cisco(RTSP)'/>
    </VHOISAConfig>
</VHOISAConfigurationSettings>
```
POST VHO ISA Setup Configuration

Following are the request components:

**Request Type: HTTP POST**

**Request**

curl --form upload=@/root/vhoisasetup.xml --form press=OK
'http://<cdsm_host>/api/services/configure/array/id/vhoisasetup'

**Request Body: XML**

**Request Example**

Following is the XML body for the POST request to upload an individual VHO ISA setup:

```xml
<?xml version="1.0" encoding="UTF-8"?><VHOISAConfigurationSettings xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><VHOISAConfig GroupName='VHO1'><StreamerBMSSettings StreamMasterPort='3300' HeadendID='0.0.0.0' StreamSourceConfig='Control IP'/><CORBAServices NameServiceIP='10.197.103.9' NameServicePort='2000' NotifyServiceIP='10.197.103.9' NotifyServicePort='2010' NotifyServiceFactory='NotifyEventChannelFactory'/><LSCPService StreamServiceID='ArroyoStreamService' StreamServiceKind='Factory' StreamChannelID='StreamChannel' StreamChannelKind='Factory' MasterNoOfThreads='32' PlayNoOfThreads='34' LSCPPort='9000' LSCPPort='9000' LSCPResponsePad='Disabled' LSCPClientProtocol='Cisco(RTSP)'
AdvancedISASettings='Disabled'</LSCPService><OpenStreamResourceServices ServiceName='ArroyoResourceManager' ServicePollTime='3600' StreamTimeout='80'
StreamSourcePort='8999'/><SessionGateways><SessionGateway>N2BBSessionGateway.Factory</SessionGateway><SessionGateway>Test20</SessionGateway></SessionGateways></VHOISAConfig><VHOISAConfig GroupName='VHO2'></VHOISAConfig></VHOISAConfigurationSettings>
```

**Site Setup**

The following is the site setup configurations in the system.

**GET ALL Site Setup Configuration**

Following are the request components:

**Request Type: HTTP GET**

**Request**

curl http://<cdsm_host>/api/services/configure/array/id/sitesetup

**Response**

If the request succeeds, the list element is returned in the XML body response containing the list of site setup.
Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?><SiteSetupConfigurationsList
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><SiteSetupConfig
SiteName='vds_site'><AssignedGroups><AssignedGroup GroupName='VG2'/><AssignedGroup
GroupName='CG1'/></AssignedGroups></SiteSetupConfig></SiteSetupConfigurationsList>
```

**POST Site Setup Configuration**

Following are the request components:

**Request Type: HTTP POST**

**Request**

```bash
curl --form upload=@/root/sitesetup.xml --form press=OK
"http://<cdsm_host>/api/services/configure/array/id/sitesetup"
```

**Request Body: XML**

**Request Example**

Following is the XML body for the POST request to upload an individual site setup:

```xml
<?xml version="1.0" encoding="UTF-8"?><SiteSetupConfigurationsList
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><SiteSetupConfig
SiteName='vds_site'><AssignedGroups><AssignedGroup GroupName='VG2'/><AssignedGroup
GroupName='CG1'/></AssignedGroups></SiteSetupConfig></SiteSetupConfigurationsList>
```

**Thin Pipe Mapping**

The following is the thin pipe mapping configurations in the system.

**GET ALL Thin Pipe Mapping Configuration**

Following are the request components:

**Request Type: HTTP GET**

**Request**

```bash
curl http://<cdsm_host>/api/services/configure/array/id/thinpipemap
```

**Response**

If the request succeeds, the list element is returned in the XML body response containing the list of site setup.

**Response Example**

```xml
<?xml version="1.0" encoding="UTF-8"?><ThinPipeMapConfigList
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"></ThinPipeMapConfigList>
```
POST Thin Pipe Mapping Configuration

Following are the request components:

Request Type: HTTP POST

Request

curl --form upload=@/root/thinpipemap.xml --form press=OK
'http://<cdsm_host>/api/services/configure/array/id/thinpipemap'

Request Body: XML

Request Example

Following is the XML body for the POST request to upload an individual thin pipe mapping:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ThinPipeMapConfigList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"></ThinPipeMapConfigList>
```

Ingest Driver Client

The following is the Ingest driver client configurations in the system.

GET ALL Ingest Driver Client Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl http://<cdsm_host>/api/services/configure/array/id/ingestdriverclient

Response

If the request succeeds, the list element is returned in the XML body response containing the Ingest driver client configuration.

Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<IngestDriverClientConfigList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><VHOGroup GroupName='Market3'><IngestDriverClientConfig Timeout='7200' AssetFactoryID='AssetFactory' AssetFactoryKind='Factory'/></VHOGroup></IngestDriverClientConfigList>
```

POST Ingest Driver Client Configuration

Following are the request components:

Request Type: HTTP POST

Request

curl --form upload=@/root/ingestdriverclient.xml --form press=OK
'http://<cdsm_host>/api/services/configure/array/id/ingestdriverclient'
Request Body: XML

Request Example
Following is the XML body for the POST request to upload an individual Ingest driver client:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<IngestDriverClientConfigList
    xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
    <VHOGroup
        GroupName='Market3'>
        <IngestDriverClientConfig
            Timeout='7200'
            AssetFactoryID='AssetFactory'
            AssetFactoryKind='Factory'/>
    </VHOGroup>
</IngestDriverClientConfigList>
```

Array Level Configuration Backup

The following is the backup of Array Level configurations in the system.

GET ALL Array Level Backup Configuration

Following are the request components:

Request Type: HTTP GET

Request

```bash
curl http://<cdsm_host>/api/services/configure/array/id/backup
```

Response

If the request succeeds, the list element is returned in the XML body response containing the list of Array Level configurations.

Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ArrayConfiguration
    xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
    <ArrayDNSConfig
        xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
        <ArrayDNS
            GroupID='1991'>
            <DomainSuffix>new1</DomainSuffix>
            <DNSServer>10.197.86.135</DNSServer>
        </ArrayDNS>
    </ArrayDNSConfig>
    <ArrayNTPConfig
        xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
        <ArrayNTP
            GroupID='1991'>
            <NTPServerIP>10.197.86.139</NTPServerIP>
        </ArrayNTP>
    </ArrayNTPConfig>
    <VaultGroupsConfigList
        xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
        <VaultGroupConfig
            GroupName='SHE1'>
            <VaultGroupMembers>
                <Server
                    IPAddress='10.197.103.53'
                    ServerID='554'
                    GroupID='1991'/>
            </VaultGroupMembers>
        </VaultGroupConfig>
        <VaultGroupConfig
            GroupName='SHE2'>
            <VaultGroupMembers>
                <Server
                    IPAddress='10.197.103.54'
                    ServerID='555'
                    GroupID='1991'/>
            </VaultGroupMembers>
        </VaultGroupConfig>
    </VaultGroupsConfigList>
    <IngestSteeringConfig
        xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
        <VaultGroupList>
            <VaultGroup
                Name='SHE1'/>
            <VaultGroup
                Name='SHE2'/>
        </VaultGroupList>
    </IngestSteeringConfig>
    <CacheGroupsConfigList
        xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
        <CacheGroupConfig
            GroupName='CG1'>
            <CacheGroupMembers>
                <Server
                    IPAddress='10.197.103.137'
                    ServerID='556'
                    GroupID='1991'/>
            </CacheGroupMembers>
        </CacheGroupConfig>
        <CacheGroupConfig
            GroupName='TestCG1'>
            <CacheGroupMembers>
                <Server
                    IPAddress='10.197.103.137'
                    ServerID='556'
                    GroupID='1991'/>
            </CacheGroupMembers>
        </CacheGroupConfig>
    </CacheGroupsConfigList>
    <CacheToCacheMapConfig
        xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
        <CacheToVaultMapConfig
            xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">"
POST Array Level Backup Configuration

Following are the request components:

Request Type: HTTP POST

Request

curl --form upload=@/root/backup.xml --form press=OK
"http://<cdsm_host>/api/services/configure/array/id/restore"

Request Body: XML

Request Example

Following is the XML body for the POST request to upload an individual backup configuration settings:

```
<?xml version="1.0" encoding="UTF-8"?><ArrayConfiguration
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><ArrayDNSConfig
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><ArrayDNS
GroupID='1991'><DomainSuffix>new1</DomainSuffix><DNSServer>10.197.86.135</DNSServer></Arra
yDNS></ArrayDNSConfig><ArrayNTPConfig
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><ArrayNTP
GroupID='1991'><NTPServerIP>10.197.86.139</NTPServerIP></ArrayNTP></ArrayNTPConfig><VaultG
roupsConfigList
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><VaultGroupConfig
GroupName='SHE1'><VaultGroupMembers><Server IPAddress='10.197.103.53' ServerID='554'
GroupID='1991'/></VaultGroupMembers></VaultGroupConfig><VaultGroupConfig
GroupName='SHE2'><VaultGroupMembers><Server IPAddress='10.197.103.54' ServerID='555'
GroupID='1991'/></VaultGroupMembers></VaultGroupConfig></VaultGroupsConfigList><IngestStee
ringConfig
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><VaultGroupList><VaultGr
oup Name='SHE1'/><VaultGroup
Name='SHE2'/></VaultGroupList></IngestSteeringConfig><CacheGroupsConfigList
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><CacheGroupConfig
GroupName='CG1'><CacheGroupMembers><Server IPAddress='10.197.103.137' ServerID='556'
GroupID='1991'/></CacheGroupMembers></CacheGroupConfig><CacheGroupConfig
GroupName='TestCG1'><CacheGroupMembers><Server IPAddress='10.197.103.137' ServerID='556'
GroupID='1991'/></CacheGroupMembers></CacheGroupConfig></CacheGroupsConfigList><CacheToCacheMapConfig
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"></CacheToCacheMapConfig>
```
Server Level Configuration API

The Server Level Configuration APIs offers the ability to download or upload an XML file that contains the server level configuration settings by way of an API call using any REpresentational State Transfer (REST) client.

Following are the Server level configuration APIs:

- Server Interface Setup
- Server Setup
- Recorder Setup
- Routes
- SNMP Agent
- Server Level DNS
- Server Level NTP
- Server Level RTSP
- Server Level FSI Setup
- Server Level Logging
- Server Level Syslog
- Server Level Backup Configuration
Server Interface Setup

GET Server Interface Setup Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl "http://<cdsm_host>/api/services/configure/server/id/interfacesetup/serverid/<server
id>/groupid/<group id>"

Example

curl
"http://10.197.92.11/api/services/configure/server/id/interfacesetup/serverid/123/groupid/
16"

Response Example

<?xml version="1.0" encoding="UTF-8"?>
<InterfaceSetupConfigList
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
  <InterfaceSetupConfig><Server Type="VAULT" PartNumber="CDE250-2A4" GroupID="1000"
ServerID="32"/>
    <InterfaceConfig Type="Management" Number="0" CachePort="" TransportPort=""
SubnetMask="255.255.255.0" IPAddress="10.197.101.132" Name="eth0"/>
    <InterfaceConfig Type="General" Number="1" CachePort="" TransportPort=""
SubnetMask="255.255.255.0" IPAddress="192.169.105.122" Name="eth1"/>
    <InterfaceConfig Type="Cache" Number="2" CachePort="" TransportPort=""
SubnetMask="" IPAddress="10.197.101.157" Name="eth2"/>
    <InterfaceConfig Type="Default Control" Number="3" CachePort="" TransportPort=""
SubnetMask="255.255.255.0" IPAddress="10.197.101.160" Name="eth3"/>
    <InterfaceConfig Type="Ingest" Number="4" CachePort="" TransportPort=""
SubnetMask="255.255.255.192" IPAddress="10.197.101.161" Name="eth4"/>
    <InterfaceConfig Type="Ingest/Cache" Number="5" CachePort="" TransportPort=""
SubnetMask="255.255.255.192" IPAddress="10.197.101.162" Name="eth5"/>
    <InterfaceConfig Type="General" Number="6" CachePort="" TransportPort=""
SubnetMask="255.255.255.192" IPAddress="10.197.101.164" Name="eth6"/>
    <InterfaceConfig Type="Ingest/Cache" Number="7" CachePort="" TransportPort=""
SubnetMask="255.255.255.192" IPAddress="10.197.101.165" Name="eth7"/>
    <InterfaceConfig Type="Not Used" Number="8" CachePort="" TransportPort="" SubnetMask="" IPAddress="" Name="eth8"/>
    <InterfaceConfig Type="Not Used" Number="9" CachePort="" TransportPort="" SubnetMask="" IPAddress="" Name="eth9"/>
  </InterfaceSetupConfig>
</InterfaceSetupConfigList>

GET ALL Server Interface Setup Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl http://<cdsm_host>/api/services/configure/server/id/interfacesetup
Response Example

<?xml version="1.0" encoding="UTF-8"?>
<InterfaceSetupConfigList
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
  <InterfaceSetupConfig><Server Type="VAULT" PartNumber="CDE250-2A4" GroupID="1000"
    ServerID="32"/>
    <InterfaceConfig Type="Management" Number="0" CachePort="" TransportPort=""
      SubnetMask="255.255.255.0" IPAddress="10.197.101.132" Name="eth0"/></InterfaceConfig
    <InterfaceConfig Type="General" Number="1" CachePort="" TransportPort=""
      SubnetMask="255.255.255.0" IPAddress="192.169.105.122" Name="eth1"/>
    <InterfaceConfig Type="Cache" Number="2" CachePort="" TransportPort=""
      SubnetMask="" IPAddress="10.197.101.157" Name="eth2"/></InterfaceConfig
    <InterfaceConfig Type="Default Control" Number="3" CachePort="" TransportPort=""
      SubnetMask="255.255.255.0" IPAddress="10.197.101.160" Name="eth3"/>
    <InterfaceConfig Type="Ingest" Number="4" CachePort="" TransportPort=""
      SubnetMask="255.255.255.192" IPAddress="10.197.101.161" Name="eth4"/>
    <InterfaceConfig Type="Ingest/Cache" Number="5" CachePort="" TransportPort=""
      SubnetMask="255.255.255.192" IPAddress="10.197.101.162" Name="eth5"/>
    <InterfaceConfig Type="General" Number="6" CachePort="" TransportPort=""
      SubnetMask="255.255.255.192" IPAddress="10.197.101.164" Name="eth6"/>
    <InterfaceConfig Type="Ingest/Cache" Number="7" CachePort="" TransportPort=""
      SubnetMask="255.255.255.192" IPAddress="10.197.101.165" Name="eth7"/>
    <InterfaceConfig Type="Not Used" Number="8" CachePort="" TransportPort=""
      SubnetMask="" IPAddress="" Name="eth8"/></InterfaceConfig
    <InterfaceConfig Type="Not Used" Number="9" CachePort="" TransportPort=""
      SubnetMask="" IPAddress="" Name="eth9"/></InterfaceConfig>
  </InterfaceSetupConfig>
</InterfaceSetupConfigList>

POST Server Interface Setup Configurations

Following are the request components:

Request Type: HTTP POST

Request

curl --form upload=@/root/interfacesetup.xml --form press=OK
"http://<cdsm_host>/api/services/configure/server/id/interfacesetup"

Request Body: XML

Request Example

Following is the XML body for the POST request to upload an individual server configuration settings:

<?xml version="1.0" encoding="UTF-8"?>
<InterfaceSetupConfigList
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
  <InterfaceSetupConfig><Server Type="VAULT" PartNumber="CDE250-2A4" GroupID="1000"
    ServerID="32"/>
    <InterfaceConfig Type="Management" Number="0" CachePort="" TransportPort=""
      SubnetMask="255.255.255.0" IPAddress="10.197.101.132" Name="eth0"/></InterfaceConfig
    <InterfaceConfig Type="General" Number="1" CachePort="" TransportPort=""
      SubnetMask="255.255.255.0" IPAddress="192.169.105.122" Name="eth1"/>
    <InterfaceConfig Type="Cache" Number="2" CachePort="" TransportPort=""
      SubnetMask="" IPAddress="10.197.101.157" Name="eth2"/></InterfaceConfig
    <InterfaceConfig Type="Default Control" Number="3" CachePort="" TransportPort=""
      SubnetMask="255.255.255.0" IPAddress="10.197.101.160" Name="eth3"/>
    <InterfaceConfig Type="Ingest" Number="4" CachePort="" TransportPort=""
      SubnetMask="255.255.255.192" IPAddress="10.197.101.161" Name="eth4"/>
    <InterfaceConfig Type="Ingest/Cache" Number="5" CachePort="" TransportPort=""
      SubnetMask="255.255.255.192" IPAddress="10.197.101.162" Name="eth5"/>
    <InterfaceConfig Type="General" Number="6" CachePort="" TransportPort=""
      SubnetMask="255.255.255.192" IPAddress="10.197.101.164" Name="eth6"/>
    <InterfaceConfig Type="Ingest/Cache" Number="7" CachePort="" TransportPort=""
      SubnetMask="255.255.255.192" IPAddress="10.197.101.165" Name="eth7"/>
    <InterfaceConfig Type="Not Used" Number="8" CachePort="" TransportPort=""
      SubnetMask="" IPAddress="" Name="eth8"/>
    <InterfaceConfig Type="Not Used" Number="9" CachePort="" TransportPort=""
      SubnetMask="" IPAddress="" Name="eth9"/></InterfaceConfig>
  </InterfaceSetupConfig>
</InterfaceSetupConfigList>
Server Setup

GET Server Setup Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl "http://<cdsm_host>/api/services/configure/server/id/serversetup/serverid/<server id>/groupid/<group id>"

Example

curl "http://10.197.92.11/api/services/configure/server/id/serversetup/serverid/123/groupid/16"

Response Example

<?xml version="1.0" encoding="UTF-8"?><ServerConfigList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><StreamServerConfig TTL="32" ArrayID="20003" HostName="vz3-s3-151" PartNumber="CDE220-2S3-C"
NullStream="Enable" LivePlayback="Disable" StunPlayErrorDelay="1000" StunPlayTimeout="10"
SourceIP="192.168.207.65" TransportPortStart="48879" TransportPortEnd="48879"
CachePort="48879" TCPEnable="Enable" StreamerIsCache="Disable" JumboSFrames="Disable"
JumboCFrames="Disable" DSCPMarkingMethod="Simple" DualCAS="Disable"
DefaultGateway="10.197.103.1" DefaultGatewayDevice="eth0" MaxNoOfInterfaces="14"<Server ServerID="123" GroupID="16"><SimpleDSCPMarking ControlDSCP="0" DataDSCP="0"
StreamDSCP="0" HTTPDSCP="0" /></ServerConfigName="eth0" Type="Management"
IPAddress="10.197.103.151" SubnetMask="255.255.255.0" TransportPort="" CachePort="" Number="0"/></InterfaceConfigName="eth1" Type="Not Used" IPAddress="" SubnetMask="" TransportPort="" CachePort="" Number="1"/></InterfaceConfigName="eth2" Type="Stream/Cache"
IPAddress="192.161.101.65" SubnetMask="" TransportPort="" CachePort="" Number="2"/></InterfaceConfigName="eth3" Type="Stream/Cache" IPAddress="192.161.101.66"
SubnetMask="" TransportPort="" CachePort="" Number="3"/></InterfaceConfigName="eth4"
Type="Stream/Cache" IPAddress="192.161.101.67" SubnetMask="" TransportPort="" CachePort="" Number="4"/></InterfaceConfigName="eth5" Type="Stream/Cache" IPAddress="192.161.101.68"
SubnetMask="" TransportPort="" CachePort="" Number="5"/></InterfaceConfigName="eth6"
Type="Stream/Cache" IPAddress="192.161.101.69" SubnetMask="" TransportPort="" CachePort="" Number="6"/></InterfaceConfigName="eth7" Type="Stream/Cache" IPAddress="192.161.101.70"
SubnetMask="" TransportPort="" CachePort="" Number="7"/></InterfaceConfigName="eth8"
Type="Stream/Cache" IPAddress="192.161.101.71" SubnetMask="" TransportPort="" CachePort="" Number="8"/></InterfaceConfigName="eth9" Type="Stream/Cache" IPAddress="192.161.101.72"
SubnetMask="" TransportPort="" CachePort="" Number="9"/></InterfaceConfigName="eth10"
Type="Stream/Cache" IPAddress="192.161.101.73" SubnetMask="" TransportPort="" CachePort="" Number="10"/></InterfaceConfigName="eth11" Type="Not Used" IPAddress="" SubnetMask="" TransportPort="" CachePort="" Number="11"/></InterfaceConfigName="eth12" Type="Not Used"
GET ALL Server Setup Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl http://<cdsm_host>/api/services/configure/server/id/serversetup

Response Example

<?xml version="1.0" encoding="UTF-8"?>
<ServerConfigList
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><StreamServerConfig
TTL="32" ArrayID="20003" HostName="vz3-s3-151" PartNumber="CDE220-2S3-C"
NullStream="Enable" LivePlayback="Disable" StunPlayErrorDelay="1000" StunPlayTimeout="10"
SourceIP="192.168.207.65" TransportPortStart="48879" TransportPortEnd="48879"
CachePort="48879" TCPTraffic="Enable" StreamerIsCache="Disable" JumboSFrames="Disable"
JumboCFrames="Disable" DSCPMarkingMethod="Simple" DualCAS="Disable"
DefaultGateway="10.197.103.1" DefaultGatewayDevice="eth0" MaxNoOfInterfaces="14"><Server
ServerID="123" GroupID="16"/></SimpleDSCPMarking ControlDSCP="0" DataDSCP="0"
StreamDSCP="0" HTTPDSCP="0" /><InterfaceConfig Name="eth0" Type="Management"
IPAddress="10.197.103.151" SubnetMask="255.255.255.0" TransportPort="" CachePort=""
Number="0"/></InterfaceConfig><InterfaceConfig Name="eth1" Type="Not Used"
IPAddress="" SubnetMask="" TransportPort="" CachePort="" Number="1"/></InterfaceConfig>

POST Server Setup Configuration

Following are the request components:

Request Type: HTTP POST

Request

curl --form upload=@/root/serversetup.xml --form press=OK
"http://<cdsm_host>/api/services/configure/server/id/serversetup"
Request Body: XML

Request Example
Following is the XML body for the POST request to upload an individual server configuration settings:

```xml
<?xml version="1.0" encoding="UTF-8"?><ServerConfigList
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><StreamServerConfig
TTL="32" ArrayID="20003" HostName="vz3-s3-151" PartNumber="CDE220-2S3-C"
NullStream="Enable" LivePlayback="Disable" StunPlayErrorDelay="1000" StunPlayTimeout="10"
SourceIP="192.168.207.65" TransportPortStart="48879" TransportPortEnd="48879"
CachePort="48879" TCPTraffic="Enable" StreamerIsCache="Disable" JumboCFrames="Disable"
JumboSFrames="Disable" DSCPMarkingMethod="Simple" DualCAS="Disable"
DefaultGateway="10.197.103.1" DefaultGatewayDevice="eth0" MaxNoOfInterfaces="14"><Server
ServerID="123" GroupID="16"/><SimpleDSCPMarking ControlDSCP="0" DataDSCP="0"
StreamDSCP="0" HTTPDSCP="0"/></ServerConfigList>
```

Recorder Setup

The Recorder Configuration API offers the ability to download or upload an XML file that contains the Recorder settings by way of an API call using any REpresentational State Transfer (REST) client. The XML file can be created by the Export or Backup feature and modified as appropriate for the Recorders. Alternatively, the XML file can be originated by using the XML schema to validate the XML file, then uploaded using HTTP POST with a REST client.

Following are the HTTP POST and GET URLs and XML examples for the Recorder Configuration API:

GET Recorder Setup Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl 'http://<cdsm_host>/api/services/configure/server/id/recorder/serverid/id/recorder/deviceid/groupid'
Example

curl 
"http://10.197.92.11/api/services/configure/server/id/recorder/serverid/123/groupid/16"

Response Example

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<RecorderServerConfigList
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
  <RecorderServerConfig ComponentName="CDSRECORDER:16:123" HostName="ISR-102.109"
DefaultGateway="172.22.102.1" HTTPDSCP="36" PartNumber="CDE250-2A4"
ModelName="CISCO:CDE250-2A4" RecorderManagerIP="172.22.102.38" NTPServerIP="10.81.254.202"
HeartbeatInterval="15" RecorderLocation="SJC-BldgFixedTEST" RecordingModify="5"
SCTE35AdMarkers="Retain" RecorderPort="50005" ProtocolVersion="1" ThreadpoolSize="16"
StatusReportInterval="60" HTTPRequestTimeout="5" HTTPResponseTimeout="5"
RecordingBandwidth="1" DeliveryBandwidth="10" StorageCapacity="1"
LocationVirtualIP="172.8.2.1" LocationSubnetMask="255.255.255.0" MaxNoOfInterfaces="10">
  <Server ServerID="123" GroupID="16"></Server>
  <ComplexInterfaceConfig Name="eth0" Type="Management" IPAddress="172.22.102.109"
SubnetMask="255.255.255.0" TransportPort="" Number="0" FTP="Disable"></ComplexInterfaceConfig>
  <ComplexInterfaceConfig Name="eth2" Type="Ingest" IPAddress="192.169.203.1"
SubnetMask="255.255.255.0" TransportPort="" Number="2" FTP="Disable"></ComplexInterfaceConfig>
  <ComplexInterfaceConfig Name="eth3" Type="Ingest/Stream/Cache"
IPAddress="192.169.203.109" SubnetMask="255.255.255.0" TransportPort="" Number="3" FTP="Disable"></ComplexInterfaceConfig>
  <ComplexInterfaceConfig Name="eth4" Type="Stream" IPAddress="192.169.203.106"
TransportPort="" Number="4" FTP="Disable"></ComplexInterfaceConfig>
  <ComplexInterfaceConfig Name="eth5" Type="Stream" IPAddress="192.169.203.107"
TransportPort="" Number="5" FTP="Disable"></ComplexInterfaceConfig>
  <ComplexInterfaceConfig Name="eth6" Type="Locate" IPAddress="192.169.203.104"
TransportPort="" Number="6" FTP="Disable"></ComplexInterfaceConfig>
</RecorderServerConfig>
</RecorderServerConfigList>

GET ALL Recorder Setup Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl http://<<cdam_host>>/api/services/configure/server/id/recorders

Response Example

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<RecorderServerConfigList
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
  <RecorderServerConfig ComponentName="CDSRECORDER:16:123" HostName="ISR-102.109"
DefaultGateway="172.22.102.1" HTTPDSCP="36" PartNumber="CDE250-2A4"
ModelName="CISCO:CDE250-2A4" RecorderManagerIP="172.22.102.38" NTPServerIP="10.81.254.202"
HeartbeatInterval="15" RecorderLocation="SJC-BldgFixedTEST" RecordingModify="5"
SCTE35AdMarkers="Retain" RecorderPort="50005" ProtocolVersion="1" ThreadpoolSize="16"
StatusReportInterval="60" HTTPRequestTimeout="5" HTTPResponseTimeout="5"
RecordingBandwidth="1" DeliveryBandwidth="10" StorageCapacity="1"
LocationVirtualIP="172.8.2.1" LocationSubnetMask="255.255.255.0" MaxNoOfInterfaces="10">
  <Server ServerID="123" GroupID="16"></Server>
Chapter 11 Configuration APIs

Server Level Configuration API

POST Recorder Setup Configuration

Following are the request components:

Request Type: HTTP POST

curl --form upload=@/root/recorders.xml --form press=OK
  "http://<cdsm_host>/api/services/configure/server/id/recorders"

Request Body: XML

Request Example

Following is the XML body for the POST request to upload the server-level ISR configuration settings:

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<RecorderServerConfigList
  xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
  <RecorderServerConfig ComponentName="CDSRECORER:16:123" HostName="ISR-102.109"
    DefaultGateway="172.22.102.1" HTTPDSCP="36" PartNumber="CDE250-2A4"
    ModelName="CISCO:CDE250-2A4" RecorderManagerIP="172.22.102.38"
    NTPServerIP="10.81.254.202" HeartbeatInterval="15"
    RecorderLocation="SJC-BldgFixedTEST" RecordingModify="5"
    SCTE35AdMarkers="Retain" RecorderPort="50005"
    ProtocolVersion="1" ThreadpoolSize="16" StatusReportInterval="60"
    HTTPRequestTimeout="5" HTTPResponseTimeout="5"
    RecordingBandwidth="1" DeliveryBandwidth="10"
    StorageCapacity="1"
    LocationVirtualIP="172.8.2.1" LocationSubnetMask="255.255.255.0"
    MaxNoOfInterfaces="10"/>
  <Server ServerID="123" GroupID="16"></Server>
  <ComplexInterfaceConfig Name="eth0" Type="Management" IPAddress="172.22.102.109"
    SubnetMask="255.255.255.0" TransportPort="" Number="0"
    FTP="Disable"></ComplexInterfaceConfig>
  <ComplexInterfaceConfig Name="eth2" Type="Ingest" IPAddress="192.169.203.1"
    SubnetMask="255.255.255.0" TransportPort="" Number="2"
    FTP="Disable"></ComplexInterfaceConfig>
  <ComplexInterfaceConfig Name="eth3" Type="Ingest/Stream/Cache"
    IPAddress="192.169.203.109" SubnetMask="255.255.255.0"
    TransportPort="" Number="3"
    FTP="Disable"></ComplexInterfaceConfig>
  <ComplexInterfaceConfig Name="eth4" Type="Stream" IPAddress="192.169.203.106"
    TransportPort="" Number="4" FTP="Disable"></ComplexInterfaceConfig>
  <ComplexInterfaceConfig Name="eth5" Type="Stream" IPAddress="192.169.203.107"
    TransportPort="" Number="5" FTP="Disable"></ComplexInterfaceConfig>
  <ComplexInterfaceConfig Name="eth6" Type="Locate" IPAddress="192.169.203.104"
    TransportPort="" Number="6" FTP="Disable"></ComplexInterfaceConfig>
</RecorderServerConfigList>
Chapter 11  Configuration APIs

Server Level Configuration API

<RecorderServerConfigList>

Routes

GET Server Routes Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl "http://<cdsm_host>/api/services/configure/server/id/route/serverid/<server id>/groupid/<group id>"

Example

curl "http://10.197.92.11/api/services/configure/server/id/route/serverid/123/groupid/16"

Response Example

<?xml version="1.0" encoding="UTF-8"?><RouteTableList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><RouteTable><Server ServerID='130' GroupID='2627'></Server><Route Network='192.168.102.0' SubnetMask='255.255.255.0' Gateway='192.168.102.1' RouteType='cServer Source'></Route></RouteTable><RouteTable><Server ServerID='132' GroupID='2627'></Server><Route Network='192.169.102.0' SubnetMask='255.255.255.0' Gateway='192.169.102.1' RouteType='cServer Source'></Route></RouteTable><RouteTable><Server ServerID='135' GroupID='2627'></Server><Route Network='192.169.102.0' SubnetMask='255.255.255.0' Gateway='192.169.102.1' RouteType='cServer Source'></Route></RouteTable><RouteTable><Server ServerID='144' GroupID='2627'></Server><Route Network='192.169.102.0' SubnetMask='255.255.255.0' Gateway='192.169.102.1' RouteType='cServer Source'></Route></RouteTable><RouteTable><Server ServerID='159' GroupID='2627'></Server><Route Network='192.169.102.0' SubnetMask='255.255.255.0' Gateway='192.169.102.1' RouteType='cServer Source'></Route></RouteTable><RouteTableList>

GET ALL Server Routes Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl http://<cdsm_host>/api/services/configure/server/id/routes

Response Example

<?xml version="1.0" encoding="UTF-8"?><RouteTableList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><RouteTable><Server ServerID='130' GroupID='2627'></Server><Route Network='192.168.102.0' SubnetMask='255.255.255.0' Gateway='192.168.102.1' RouteType='cServer Source'></Route></RouteTable><RouteTable><Server ServerID='132' GroupID='2627'></Server><Route Network='192.169.102.0' SubnetMask='255.255.255.0' Gateway='192.169.102.1' RouteType='cServer Source'></Route></RouteTable><RouteTable><Server ServerID='135' GroupID='2627'></Server><Route Network='192.169.102.0' SubnetMask='255.255.255.0' Gateway='192.169.102.1' RouteType='cServer Source'></Route></RouteTable><RouteTable><Server ServerID='144' GroupID='2627'></Server><Route Network='192.169.102.0' SubnetMask='255.255.255.0' Gateway='192.169.102.1' RouteType='cServer Source'></Route></RouteTable><RouteTable><Server ServerID='159' GroupID='2627'></Server><Route Network='192.169.102.0' SubnetMask='255.255.255.0' Gateway='192.169.102.1' RouteType='cServer Source'></Route></RouteTable><RouteTableList>
**POST Server Routes Configuration**

Following are the request components:

- **Request Type:** HTTP POST

  ```bash
curl --form upload=/root/route.xml --form press=OK
  "http://<cdsm_host>/api/services/configure/server/id/route"
  ``

- **Request Body:** XML

  ```xml
  <?xml version="1.0" encoding="UTF-8"?>
  <RouteTableList
    xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
    <RouteTable><Server ServerID='130' GroupID='2627'></Server>
    <Route Network='192.169.102.0' SubnetMask='255.255.255.0' Gateway='192.169.102.1' RouteType='cServer Source'></Route>
  </RouteTable>
  <RouteTable><Server ServerID='132' GroupID='2627'></Server>
  <Route Network='192.169.102.0' SubnetMask='255.255.255.0' Gateway='192.169.102.1' RouteType='cServer Source'></Route>
  </RouteTable>
  <RouteTable><Server ServerID='135' GroupID='2627'></Server>
  <Route Network='192.169.102.0' SubnetMask='255.255.255.0' Gateway='192.169.102.1' RouteType='cServer Source'></Route>
  </RouteTable>
  <RouteTable><Server ServerID='144' GroupID='2627'></Server>
  <Route Network='192.169.102.0' SubnetMask='255.255.255.0' Gateway='192.169.102.1' RouteType='cServer Source'></Route>
  </RouteTable>
  <RouteTable><Server ServerID='159' GroupID='2627'></Server>
  <Route Network='192.169.102.0' SubnetMask='255.255.255.0' Gateway='192.169.102.1' RouteType='cServer Source'></Route>
  </RouteTable>
  </RouteTableList>
  ```

**SNMP Agent**

**GET Server SNMP Agent Configuration**

Following are the request components:

- **Request Type:** HTTP GET

  ```bash
curl "http://<cdsm_host>/api/services/configure/server/id/snmpagent/serverid/<server id>/groupid/<group id>"
  ```
Example

curl
"http://10.197.92.11/api/services/configure/server/id/snmpagent/serverid/123/groupid/16"

Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<SNMPAgentList

xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
<SNMPAgent

Contact='vdstv' Location='location1' DefaultTrapCommunity='trap' >
<Server ServerID='223' GroupID='2000' />
<SNMPCommunity Name='vdstv_cisco' Permissions='Read-Write'/>
<SNMPTrapStation TrapStation='trap_name' Version='v1'

TrapCommunity='trap' />
<SNMPVACM User='user1' Access='Read-Write' Authentication='None'

OID='' />
<SNMPUSM User='user1' Authentication='MD5' AuthenticationPassword='root@123'

Encryption='DES' EncryptionPassword='root@123' />
</SNMPAgent>

</SNMPAgentList>
```

GET ALL Server SNMP Agent Configuration

Following are the request components:

Request Type: HTTP GET

Request
curl http://<cdsm_host>/api/services/configure/server/id/snmpagents

Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<SNMPAgentList

xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
<SNMPAgent

Contact='vdstv' Location='location1' DefaultTrapCommunity='trap' >
<Server ServerID='223' GroupID='2000' />
<SNMPCommunity Name='vdstv_cisco' Permissions='Read-Write'/>
<SNMPTrapStation TrapStation='trap_name' Version='v1'

TrapCommunity='trap' />
<SNMPVACM User='user1' Access='Read-Write' Authentication='None'

OID='' />
<SNMPUSM User='user1' Authentication='MD5' AuthenticationPassword='root@123'

Encryption='DES' EncryptionPassword='root@123' />
</SNMPAgent>

</SNMPAgentList>
```

POST Server SNMP Agent Configuration

Following are the request components:

Request Type: HTTP POST

Request
curl --form upload=@/root/snmp.xml --form press=OK
"http://<cdsm_host>/api/services/configure/server/id/snmp"

Request Body: XML

Request Example

Following is the XML body for the POST request to upload an individual server configuration settings:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<SNMPAgentList

xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
<SNMPAgent

Contact='vdstv' Location='location1' DefaultTrapCommunity='trap' >
<Server ServerID='223' GroupID='2000' />
<SNMPCommunity Name='vdstv_cisco' Permissions='Read-Write'/>
<SNMPTrapStation TrapStation='trap_name' Version='v1'

TrapCommunity='trap' />
<SNMPVACM User='user1' Access='Read-Write' Authentication='None'

OID='' />
<SNMPUSM User='user1' Authentication='MD5' AuthenticationPassword='root@123'

Encryption='DES' EncryptionPassword='root@123' />
</SNMPAgent>

</SNMPAgentList>
```
Server Level DNS

GET Server Level DNS Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl "http://<cdsm_host>/api/services/configure/server/id/dnsserver/serverid/<server id>/groupid/<group id>"

Example

curl "http://10.197.92.11/api/services/configure/server/id/dnsserver/serverid/123/groupid/16"

Response Example

<?xml version="1.0" encoding="UTF-8"?><DNSList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><DNS><Server ServerID='123' GroupID='16'/><DomainSuffix>cisco</DomainSuffix><DNSServer>10.197.92.161</DNSServer></DNS></DNSList>

GET ALL Server Level DNS Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl http://<cdsm_host>/api/services/configure/server/id/dnsservers

Response Example

<?xml version="1.0" encoding="UTF-8"?><DNSList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><DNS><Server ServerID='123' GroupID='16'/><DomainSuffix>cisco</DomainSuffix><DNSServer>10.197.92.161</DNSServer></DNS></DNSList>
POST Server Level DNS Configuration

Following are the request components:

Request Type: HTTP POST

Request

curl --form upload=@/root/dns.xml --form press=OK
"http://<cdsm_host>/api/services/configure/server/id/dns"

Request Body: XML

Request Example

Following is the XML body for the POST request to upload an individual server configuration settings:

<?xml version="1.0" encoding="UTF-8"?><DNSList
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><DNS>
<Server ServerID='123'
GroupID='16'/><DomainSuffix>cisco</DomainSuffix><DNSServer>10.197.92.161</DNSServer></DNS>
</DNSList>

Server Level NTP

GET Server Level NTP Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl "http://<cdsm_host>/api/services/configure/server/id/ntpserver/serverid/<server id>/groupid/<group id>"

Example

curl "http://10.197.92.11/api/services/configure/server/id/ntpserver/serverid/123/groupid/16"

Response Example

<?xml version="1.0" encoding="UTF-8"?><NTPServerList
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><NTPServer><Server ServerID='123'
GroupID='16'/><NTPServerIP>10.197.92.11</NTPServerIP></NTPServer></NTPServerList>

GET ALL Server Level NTP Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl http://<cdsm_host>/api/services/configure/server/id/ntpservers
Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?><NTPServerList
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">  <NTPServer><Server
ServerID='123'
GroupID='16'/><NTPServerIP>10.197.92.11</NTPServerIP></Server></NTPServerList>
```

**POST Server Level NTP Configuration**

Following are the request components:

**Request Type: HTTP POST**

Request : curl --form upload=@/root/ntp.xml --form press=OK
  "http://<cdsm_host>/api/services/configure/server/id/ntp"

**Request Body: XML**

**Request Example**

Following is the XML body for the POST request to upload an individual server configuration settings:

```xml
<?xml version="1.0" encoding="UTF-8"?><NTPServerList
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">  <NTPServer><Server
ServerID='123'
GroupID='16'/><NTPServerIP>10.197.92.11</NTPServerIP></Server></NTPServerList>
```

**Server Level RTSP**

**GET Server Level RTSP Configuration**

Following are the request components:

**Request Type: HTTP GET**

Request

curl 'http://<cdsm_host>/api/services/configure/server/id/rtspserver/serverid/<server
id>/groupid/<group id>''

**Example**

curl
  'http://10.197.92.11/api/services/configure/server/id/rtspserver/serverid/123/groupid/16''

**Response Example**

```xml
<?xml version="1.0" encoding="UTF-8"?><RTSPSetupList
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">  <RTSPSetup
MasterStreamingIP="10.197.102.160" SessionInactivityTimeout="18000000" SessionPauseTimeout="900000" BackofficeTimeout="18000000" RTSPServerIP="10.197.102.160" RTSPServerPort="554" LSCPAddress="10.197.102.160" LSCPPort="9000" LSCPResponsePadding="Off" ComponentName="SOP1" MaintenanceMode="Off" Server ServerID="10" GroupID="2627" />
<RTSPClient ReceivePort="554" SendPort="554" ReceiveBuffer="65535" Model="ngod" Transport="TCP" />
<RTSPClient ReceivePort="5554" SendPort="5554" ReceiveBuffer="65535" Model="ngod" Transport="TCP" />
<RTSPSetup MasterStreamingIP="10.197.102.31" SessionInactivityTimeout="18000000" SessionPauseTimeout="900000" BackofficeTimeout="18000000" RTSPServerIP="10.197.102.31" RTSPServerPort="554" LSCPAddress="10.197.102.31" LSCPPort="9000" LSCPResponsePadding="Off" ComponentName="SOP1" MaintenanceMode="Off" Server ServerID="10" GroupID="2627" />
</RTSPSetupList>
```
GET ALL Server Level RTSP Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl http://<cdsm_host>/api/services/configure/server/id/rtsp

Response Example

<?xml version="1.0" encoding="UTF-8"?>
<RTSPSetupList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
  <RTSPSetup MasterStreamingIP="10.197.102.160" SessionInactivityTimeout="18000000" SessionPauseTimeout="900000" RTSPServerPort="554" LSCPAddress="10.197.102.31" LSCPPort="9000" LSCPResponsePadding="Off" ComponentName="SOP1" MaintenanceMode="Off"><Server ServerID="10" GroupID="2627" /></RTSPSetup>
  <RTSPSetup MasterStreamingIP="10.197.102.31" SessionInactivityTimeout="18000000" SessionPauseTimeout="900000" BackofficeTimeout="18000000" RTSPServerIP="10.197.102.31" RTSPServerPort="554" LSCPAddress="10.197.102.31" LSCPPort="9000" LSCPResponsePadding="Off" ComponentName="SOP1" MaintenanceMode="Off"><Server ServerID="57" GroupID="2627" /></RTSPSetup>
  <RTSPSetup MasterStreamingIP="10.197.102.31" SessionInactivityTimeout="18000000" SessionPauseTimeout="900000" BackofficeTimeout="18000000" RTSPServerIP="10.197.102.31" RTSPServerPort="554" LSCPAddress="10.197.102.31" LSCPPort="9000" LSCPResponsePadding="Off" ComponentName="SOP1" MaintenanceMode="Off"><Server ServerID="152" GroupID="2627" /></RTSPSetup>
  <RTSPSetup MasterStreamingIP="10.197.102.31" SessionInactivityTimeout="18000000" SessionPauseTimeout="900000" BackofficeTimeout="18000000" RTSPServerIP="10.197.102.31" RTSPServerPort="554" LSCPAddress="10.197.102.31" LSCPPort="9000" LSCPResponsePadding="Off" ComponentName="SOP1" MaintenanceMode="Off"><Server ServerID="158" GroupID="2627" /></RTSPSetup>
</RTSPSetupList>
POST Server Level RTSP Configuration

Following are the request components:

**Request Type: HTTP POST**

Request: `curl --form upload=@/root/rtsp.xml --form press=OK 'http://<cdsm_host>/api/services/configure/server/id/rtsp`

**Request Body: XML**

**Request Example**

Following is the XML body for the POST request to upload an individual server configuration settings:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<RTSPSetupList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><RTSPSetup
MasterStreamingIP="10.197.102.160" SessionInactivityTimeout="18000000"
SessionPauseTimeout="900000" BackofficeTimeout="18000000" RTSPServerIP="10.197.102.160"
RTSPServerPort="554" LSCPAddress="10.197.102.160" LSCPPort="9000"
LSCPResponsePadding="Off" ComponentName="SOP1" MaintenanceMode="Off"><Server ServerID="10"
GroupID="2627"></Server></RTSPSetup><RTSPSetup
MasterStreamingIP="10.197.102.31" SessionInactivityTimeout="18000000"
SessionPauseTimeout="900000" BackofficeTimeout="18000000" RTSPServerIP="10.197.102.31"
RTSPServerPort="554" LSCPAddress="10.197.102.31" LSCPPort="9000" LSCPResponsePadding="Off"
ComponentName="SOP1" MaintenanceMode="Off"><Server ServerID="152"
GroupID="2627"></Server></RTSPSetup><RTSPSetup
MasterStreamingIP="10.197.102.31" SessionInactivityTimeout="18000000"
SessionPauseTimeout="900000" BackofficeTimeout="18000000" RTSPServerIP="10.197.102.31"
RTSPServerPort="554" LSCPAddress="10.197.102.31" LSCPPort="9000" LSCPResponsePadding="Off"
ComponentName="SOP1" MaintenanceMode="Off"><Server ServerID="152"
GroupID="2627"></Server></RTSPSetup><RTSPSetup
MasterStreamingIP="10.197.102.160" SessionInactivityTimeout="18000000"
SessionPauseTimeout="900000" BackofficeTimeout="18000000" RTSPServerIP="10.197.102.160"
RTSPServerPort="554" LSCPAddress="10.197.102.160" LSCPPort="9000" LSCPResponsePadding="Off"
ComponentName="SOP1" MaintenanceMode="Off"><Server ServerID="158"
GroupID="2627"></Server></RTSPSetup></RTSPSetupList>
Server Level FSI Setup

GET Server Level FSI Setup Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl "http://<cdsm_host>/api/services/configure/server/id/fsiserver/serverid/<server id>/groupid/<group id>"

Example

curl

"http://10.197.92.11/api/services/configure/server/id/fsiserver/serverid/123/groupid/16"

Response Example

<?xml version="1.0" encoding="UTF-8"?>
<FSISetupList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><FSISetup IPAddress="0.0.0.0" ServerPort="20004" FTPClientPort="21" FTPOutServerPort="21" FTPOutLoginTTL="64" ContentRootPath="/files" AsyncCallbackURL=""><Server ServerID="130" GroupID="2627"></Server>
<FSISetup IPAddress="0.0.0.0" ServerPort="20004" FTPClientPort="21" FTPOutServerPort="21" FTPOutLoginTTL="64" ContentRootPath="/files" AsyncCallbackURL=""><Server ServerID="135" GroupID="2627"></Server></FSISetupList>

GET ALL Server Level FSI Setup Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl http://<cdsm_host>/api/services/configure/server/id/fsi

Response Example

<?xml version="1.0" encoding="UTF-8"?>
<FSISetupList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><FSISetup IPAddress="0.0.0.0" ServerPort="20004" FTPClientPort="21" FTPOutServerPort="21" FTPOutLoginTTL="64" ContentRootPath="/files" AsyncCallbackURL=""><Server ServerID="130" GroupID="2627"></Server>
<FSISetup IPAddress="0.0.0.0" ServerPort="20004" FTPClientPort="21" FTPOutServerPort="21" FTPOutLoginTTL="64" ContentRootPath="/files" AsyncCallbackURL=""><Server ServerID="135" GroupID="2627"></Server></FSISetupList>
POST Server Level FSI Setup Configuration

Following are the request components:

Request Type: HTTP POST

Request

curl --form upload=@/root/fsi.xml --form press=OK
'http://<cdsm_host>/api/services/configure/server/id/fsi

Request Body: XML

Request Example

Following is the XML body for the POST request to upload an individual server configuration settings:

```xml
<?xml version="1.0" encoding="UTF-8"?><FSISetupList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><FSISetup IPAddress="0.0.0.0" ServerPort="20004" FTPClientPort="21" FTPOutServerPort="21"
 FTPOutLoginTTL="64" ContentRootPath="/files" AsyncCallbackURL=""><Server ServerID="130"
 GroupID="2627"/></FSISetup><FSISetup IPAddress="0.0.0.0" ServerPort="20004"
 FTPClientPort="21" FTPOutServerPort="21" FTPOutLoginTTL="64" ContentRootPath="/files"
 AsyncCallbackURL=""><Server ServerID="135" GroupID="2627"/></FSISetup></FSISetupList>
```

Server Level Logging

GET Server Level Logging Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl 'http://<cdsm_host>/api/services/configure/server/id/loggingserver/serverid/<server id>/groupid/<group id>

Example

curl
"http://10.197.92.11/api/services/configure/server/id/loggingserver//serverid/123/groupid/16"

Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?><LogConfigList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><LogConfig><Server ServerID="130" GroupID="2627"/><Facility name="aim" local-log="notice"
 remote-log="DISABLE"><debug-flags><flag name="general"></debug-flags></Facility></LogConfig></LogConfigList>
```
**GET ALL Server Level Logging Configuration**

Following are the request components:

**Request Type: HTTP GET**

**Request**

curl http://<cdsm_host>/api/services/configure/server/id/logging

**Response Example**

```xml
<?xml version="1.0" encoding="UTF-8"?><LogConfigList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><LogConfig><Server ServerID="130" GroupID="2627"><Facility name="aim" local-log="notice" remote-log="DISABLE"><debug-flags><flag name="general"></debug-flags></Facility></LogConfig></LogConfigList>
```

**POST Server Level Logging Configuration**

Following are the request components:

**Request Type: HTTP POST**

**Request**

curl --form upload=@/root/logging.xml --form press=OK "http://<cdsm_host>/api/services/configure/server/id/logging

**Request Body: XML**

**Request Example**

Following is the XML body for the POST request to upload an individual server configuration settings:

```xml
<?xml version="1.0" encoding="UTF-8"?><LogConfigList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><LogConfig><Server ServerID="130" GroupID="2627"><Facility name="aim" local-log="notice" remote-log="DISABLE"><debug-flags><flag name="general"></debug-flags></Facility></LogConfig></LogConfigList>
```

**Server Level Syslog**

**GET Server Level Syslog Configuration**

Following are the request components:

**Request Type: HTTP GET**

**Request**

curl "http://<cdsm_host>/api/services/configure/server/id/syslogserver/serverid/<server id>/groupid/<group id>"
Example

curl
'http://10.197.92.11/api/services/configure/server/id/syslogserver/serverid/123/groupid/16
`

Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?><SyslogConfigList
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><SyslogConfig
RemoteLogging='Disable' ><Server ServerID='135' GroupID='2627'
/></SyslogConfig></SyslogConfigList>
```

GET ALL Server Level Syslog Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl http://<cdsm_host>/api/services/configure/server/id/syslog

Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?><SyslogConfigList
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><SyslogConfig
RemoteLogging='Disable' ><Server ServerID='135' GroupID='2627'
/></SyslogConfig></SyslogConfigList>
```

POST Server Level Syslog Configuration

Following are the request components:

Request Type: HTTP POST

Request

curl --form upload=@/root/syslog.xml --form press=OK
'http://<cdsm_host>/api/services/configure/server/id/syslog

Request Body: XML

Request Example

Following is the XML body for the POST request to upload an individual server configuration settings:

```xml
<?xml version="1.0" encoding="UTF-8"?><SyslogConfigList
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><SyslogConfig
RemoteLogging='Disable' ><Server ServerID='135' GroupID='2627'
/></SyslogConfig></SyslogConfigList>
```
Server Level Backup Configuration

GET ALL Server Level Backup Configuration

Following are the request components:

Request Type: HTTP GET

Request:
curl http://<cdsm_host>/api/services/configure/server/id/backup

Response Example:

```xml
<?xml version="1.0" encoding="UTF-8"?><ServerConfiguration xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><RouteTableList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><RouteTable><Server ServerID='556' GroupID='1991'></Server><Route Network='10.197.103.0' SubnetMask='255.255.255.0' Gateway='10.197.103.1' RouteType='cServer Destination'></Route><Route Network='192.161.105.0' SubnetMask='255.255.255.0' Gateway='192.161.105.1' RouteType='cServer Source'></Route></RouteTable><RouteTable><Server ServerID='554' GroupID='1991'></Server><Route Network='10.197.103.0' SubnetMask='255.255.255.0' Gateway='10.197.103.1' RouteType='cServer Destination'></Route><Route Network='192.161.105.0' SubnetMask='255.255.255.0' Gateway='192.161.105.1' RouteType='cServer Source'></Route></RouteTable><RouteTable><Server ServerID='555' GroupID='1991'></Server><Route Network='10.197.103.0' SubnetMask='255.255.255.0' Gateway='10.197.103.1' RouteType='cServer Destination'></Route><Route Network='192.161.105.0' SubnetMask='255.255.255.0' Gateway='192.161.105.1' RouteType='cServer Source'></Route></RouteTable><SNMPAgentList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><SNMPAgent Contact='SNMPcontact' Location='snmplocation' DefaultTrapCommunity='default trap community'><Server ServerID='556' GroupID='1991'/></SNMPAgent><SNMPAgent Contact='snmp' Location='snmp' DefaultTrapCommunity='snmp'><Server ServerID='554' GroupID='1991'/></SNMPAgent><DNSList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><DNS><Server ServerID='554' GroupID='1991'/><DomainSuffix>S1</DomainSuffix><DomainSuffix>S2</DomainSuffix><DNSServer>3.3.3.3</DNSServer><DNSServer>2.2.2.2</DNSServer></DNS><NTPServerList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><NTPServer><Server ServerID='556' GroupID='1991'/><NTPServerIP>10.197.86.134</NTPServerIP></NTPServer><NTPServer><Server ServerID='554' GroupID='1991'/><NTPServerIP>10.197.86.134</NTPServerIP></NTPServer><NTPServer><Server ServerID='555' GroupID='1991'/><NTPServerIP>10.197.86.134</NTPServerIP></NTPServer></NTPServerList><ServerConfigList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><VaultServerConfig ArrayID="1" SourceIP='192.168.207.65' CachePort='48879' TCPTraffic="Enable" VaultLocalCopies=1 VaultMirrorCopies=2 TTL=32 FTPOutSession=0 FTPOutBandwidth=0 FTPOutInterface='0' HostName='VZ3-V1-53' PartNumber='CDE250-2A4' JumboFrames='disable' DSCPMarkingMethod='Simple' DualCAS='disable' DefaultGateway='10.197.103.1' DefaultGatewayDevice="eth0" MaxNOOfInterfaces=10><Server ServerID='554' GroupID='1991'/><SimpleDSCPMarking ControlDSCP=0 DataDSCP=0 /></ServerConfigList></ServerConfiguration>
```
Server Level Configuration API

Type="Cache" IPAddress="192.161.105.56" SubnetMask="" TransportPort="" CachePort="" Number="7"/>
<InterfaceConfig Name="eth8" Type="Cache" IPAddress="192.161.105.57" SubnetMask="" TransportPort="" CachePort="" Number="8"/>
<InterfaceConfig Name="eth9" Type="Cache" IPAddress="192.161.105.58" SubnetMask="" TransportPort="" CachePort="" Number="9"/>
</VaultServerConfig>
</ServerConfigList>

<LogConfigList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
<LogConfig>
<Server ServerID="556" GroupID="1991"/>
<Facility name="avsdb" local-log="" remote-log=""/>
<Facility name="bwm" local-log="" remote-log=""/>
<Facility name="c2k" local-log="" remote-log=""/>
<Facility name="collectd" local-log="" remote-log=""/>
<Facility name="dnsresolver" local-log="" remote-log=""/>
<Facility name="http" local-log="" remote-log=""/>
<Facility name="ns" local-log="" remote-log=""/>
<Facility name="prodassert" local-log="" remote-log=""/>
<Facility name="statsd" local-log="" remote-log=""/>
<Facility name="utils" local-log="" remote-log=""/>
</LogConfig>
</LogConfigList>
POST Server Level Backup Configuration

Following are the request components:

**Request Type:** HTTP POST

**Request**

curl --form upload=@/root/restore.xml --form press=OK
"http://<cdsm_host>/api/services/configure/server/id/restore"

**Request Body:** XML

**Request Example**

Following is the XML body for the POST request to upload an individual server configuration settings:

```xml
<?xml version="1.0" encoding="UTF-8"?><ServerConfiguration xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><RouteTableList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><RouteTable><Server ServerID='556' GroupID='1991'></Server><Route Network='10.197.103.0'
```
SubnetMask='255.255.255.0' Gateway='10.197.103.1' RouteType='cServer Destination'></Route><Route Network='192.161.105.0' SubnetMask='255.255.255.0' Gateway='10.197.103.1' RouteType='cServer Source'></Route><Route Table='Server ID=554' GroupID='1991'></Route><Route Table='Server ID=555' GroupID='1991'></Route></RouteTable></RouteTableList><SNMPAgentList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><SNMPAgent Contact='SNMPcontact' Location='snmplocation' DefaultTrapCommunity='default trap community'><Server ServerID='554' GroupID='1991'/></SNMPAgent><SNMPAgent Contact='snmp' Location='snmp' DefaultTrapCommunity='snmp'><Server ServerID='554' GroupID='1991'/></SNMPAgent></SNMPAgentList><DNSList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><DNS><Server ServerID='554' GroupID='1991'></Server><DomainSuffix>S1</DomainSuffix><DomainSuffix>S2</DomainSuffix><DNSServer>3.3.3.3</DNSServer><DNSServer>2.2.2.2</DNSServer></DNS></DNSList><NTPServerList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><NTPServer><Server ServerID='556' GroupID='1991'></Server><NTPServerIP>10.197.86.134</NTPServerIP></NTPServer><NTPServer><Server ServerID='554' GroupID='1991'></Server><NTPServerIP>10.197.86.134</NTPServerIP></NTPServer><NTPServer><Server ServerID='555' GroupID='1991'></Server></NTPServer></NTPServerList><ServerConfigList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><VaultServerConfig ArrayID="1" SourceIP="192.168.207.65" CachePort="48879" TCPTraffic="Enable" VaultLocalCopies="1" VaultMirrorCopies="2" TTL="32" FTPOutSession="0" FTPOutBandwidth="0" FTPOutInterface="0" HostName="VZ3-V1-53" PartNumber="CDE250-2A4" JumboFrames="Disable" DSCPBandwidth="0" Simple DualCAS="Disable" DefaultGateway='10.197.103.1' DefaultGatewayDevice='eth0' MaxNoOfInterfaces='10'><Server ServerID='554' GroupID='1991'/></Server></VaultServerConfig><VaultServerConfig ArrayID="2" SourceIP="192.168.207.65" CachePort="48879" TCPTraffic="Enable" VaultLocalCopies="1" VaultMirrorCopies="2" TTL="32" FTPOutSession="0" FTPOutBandwidth="0" FTPOutInterface="0" HostName="VZ3-V2-54" PartNumber="CDE250-2A4" JumboFrames="Disable" DSCPBandwidth="0" Simple DualCAS="Disable" DefaultGateway='10.197.103.1' DefaultGatewayDevice='eth0' MaxNoOfInterfaces='10'><Server ServerID='555' GroupID='1991'/></Server></VaultServerConfig>
SubnetMask="" TransportPort="" CachePort="" Number="6"/>
</InterfaceConfig>

Type="Cache" IPAddress="192.161.101.65" SubnetMask="" TransportPort="" CachePort="" Number="7"/>
</InterfaceConfig>

Type="Not Used" IPAddress="" SubnetMask="" TransportPort="" CachePort="" Number="8"/>
</InterfaceConfig>

Type="Not Used" IPAddress="" SubnetMask="" TransportPort="" CachePort="" Number="9"/>
</InterfaceConfig>

<InterfaceConfig Name="eth7" Type="Cache" IPAddress="192.161.105.76" SubnetMask="" TransportPort="" CachePort="" Number="7"/>
</InterfaceConfig>

<InterfaceConfig Name="eth8" Type="Cache" IPAddress="192.161.105.77" SubnetMask="" TransportPort="" CachePort="" Number="8"/>
</InterfaceConfig>

<InterfaceConfig Name="eth9" Type="Cache" IPAddress="192.161.105.78" SubnetMask="" TransportPort="" CachePort="" Number="9"/>
</InterfaceConfig>

<InterfaceConfig Name="eth10" Type="Cache" IPAddress="192.161.105.80" SubnetMask="" TransportPort="" CachePort="" Number="10"/>
</InterfaceConfig>

<InterfaceConfig Name="eth11" Type="Not Used" IPAddress="" SubnetMask="" TransportPort="" CachePort="" Number="11"/>
</InterfaceConfig>

<InterfaceConfig Name="eth12" Type="Not Used" IPAddress="" SubnetMask="" TransportPort="" CachePort="" Number="12"/>
</InterfaceConfig>

<InterfaceConfig Name="eth13" Type="Not Used" IPAddress="" SubnetMask="" TransportPort="" CachePort="" Number="13"/>
</InterfaceConfig>

<LogConfigList xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><LogConfig>
  <Server ServerID="556" GroupID="1991"/>
  <Facility name="avsdb" local-log="" remote-log=""/>
  <Facility name="bwm" local-log="" remote-log=""/>
  <Facility name="c2k" local-log="" remote-log=""/>
  <Facility name="collectd" local-log="" remote-log=""/>
  <Facility name="dnsresolver" local-log="" remote-log=""/>
  <Facility name="http" local-log="" remote-log=""/>
  <Facility name="ns" local-log="" remote-log=""/>
  <Facility name="prodassert" local-log="" remote-log=""/>
  <Facility name="statsd" local-log="" remote-log=""/>
  <Facility name="utils" local-log="" remote-log=""/>
</LogConfig>

<LogConfig>
  <Server ServerID="554" GroupID="1991"/>
  <Facility name="aim" local-log="notice" remote-log="DISABLE"/>
  <Facility name="avsdb" local-log="" remote-log=""/>
  <Facility name="bwm" local-log="" remote-log=""/>
  <Facility name="c2k" local-log="" remote-log=""/>
  <Facility name="collectd" local-log="" remote-log=""/>
  <Facility name="dnsresolver" local-log="" remote-log=""/>
  <Facility name="http" local-log="" remote-log=""/>
  <Facility name="ns" local-log="" remote-log=""/>
  <Facility name="prodassert" local-log="" remote-log=""/>
  <Facility name="statsd" local-log="" remote-log=""/>
  <Facility name="utils" local-log="" remote-log=""/>
</LogConfig>

<LogConfig>
  <Server ServerID="555" GroupID="1991"/>
  <Facility name="aim" local-log="" remote-log=""/>
  <Facility name="avsdb" local-log="" remote-log=""/>
  <Facility name="bwm" local-log="" remote-log=""/>
  <Facility name="c2k" local-log="" remote-log=""/>
  <Facility name="collectd" local-log="" remote-log=""/>
  <Facility name="dnsresolver" local-log="" remote-log=""/>
  <Facility name="http" local-log="" remote-log=""/>
  <Facility name="ns" local-log="" remote-log=""/>
  <Facility name="prodassert" local-log="" remote-log=""/>
  <Facility name="pvr_asset_manager" local-log="" remote-log=""/>
  <Facility name="statsd" local-log="" remote-log=""/>
  <Facility name="tvod_manager" local-log="" remote-log=""/>
  <Facility name="utils" local-log="" remote-log=""/>
</LogConfig>

<LogConfig>
  <Server ServerID="556" GroupID="1991"/>
  <Facility name="avsdb" local-log="" remote-log=""/>
  <Facility name="bwm" local-log="" remote-log=""/>
  <Facility name="c2k" local-log="" remote-log=""/>
  <Facility name="collectd" local-log="" remote-log=""/>
  <Facility name="dnsresolver" local-log="" remote-log=""/>
  <Facility name="http" local-log="" remote-log=""/>
  <Facility name="ns" local-log="" remote-log=""/>
  <Facility name="prodassert" local-log="" remote-log=""/>
  <Facility name="pvr_asset_manager" local-log="" remote-log=""/>
  <Facility name="statsd" local-log="" remote-log=""/>
  <Facility name="tvod_manager" local-log="" remote-log=""/>
  <Facility name="utils" local-log="" remote-log=""/>
</LogConfig>

<LogConfig>
  <Server ServerID="557" GroupID="1991"/>
  <Facility name="avsdb" local-log="" remote-log=""/>
  <Facility name="bwm" local-log="" remote-log=""/>
  <Facility name="c2k" local-log="" remote-log=""/>
  <Facility name="collectd" local-log="" remote-log=""/>
  <Facility name="dnsresolver" local-log="" remote-log=""/>
  <Facility name="http" local-log="" remote-log=""/>
  <Facility name="ns" local-log="" remote-log=""/>
  <Facility name="prodassert" local-log="" remote-log=""/>
  <Facility name="pvr_asset_manager" local-log="" remote-log=""/>
  <Facility name="statsd" local-log="" remote-log=""/>
  <Facility name="tvod_manager" local-log="" remote-log=""/>
  <Facility name="utils" local-log="" remote-log=""/>
</LogConfig>

<LogConfig>
  <Server ServerID="558" GroupID="1991"/>
  <Facility name="avsdb" local-log="" remote-log=""/>
  <Facility name="bwm" local-log="" remote-log=""/>
  <Facility name="c2k" local-log="" remote-log=""/>
  <Facility name="collectd" local-log="" remote-log=""/>
  <Facility name="dnsresolver" local-log="" remote-log=""/>
  <Facility name="http" local-log="" remote-log=""/>
  <Facility name="ns" local-log="" remote-log=""/>
  <Facility name="prodassert" local-log="" remote-log=""/>
  <Facility name="pvr_asset_manager" local-log="" remote-log=""/>
  <Facility name="statsd" local-log="" remote-log=""/>
  <Facility name="tvod_manager" local-log="" remote-log=""/>
  <Facility name="utils" local-log="" remote-log=""/>
</LogConfig>

<LogConfig>
  <Server ServerID="559" GroupID="1991"/>
  <Facility name="avsdb" local-log="" remote-log=""/>
  <Facility name="bwm" local-log="" remote-log=""/>
  <Facility name="c2k" local-log="" remote-log=""/>
  <Facility name="collectd" local-log="" remote-log=""/>
  <Facility name="dnsresolver" local-log="" remote-log=""/>
  <Facility name="http" local-log="" remote-log=""/>
  <Facility name="ns" local-log="" remote-log=""/>
  <Facility name="prodassert" local-log="" remote-log=""/>
  <Facility name="pvr_asset_manager" local-log="" remote-log=""/>
  <Facility name="statsd" local-log="" remote-log=""/>
  <Facility name="tvod_manager" local-log="" remote-log=""/>
  <Facility name="utils" local-log="" remote-log=""/>
</LogConfig>

<LogConfig>
  <Server ServerID="560" GroupID="1991"/>
  <Facility name="avsdb" local-log="" remote-log=""/>
  <Facility name="bwm" local-log="" remote-log=""/>
  <Facility name="c2k" local-log="" remote-log=""/>
  <Facility name="collectd" local-log="" remote-log=""/>
  <Facility name="dnsresolver" local-log="" remote-log=""/>
  <Facility name="http" local-log="" remote-log=""/>
  <Facility name="ns" local-log="" remote-log=""/>
  <Facility name="prodassert" local-log="" remote-log=""/>
  <Facility name="pvr_asset_manager" local-log="" remote-log=""/>
  <Facility name="statsd" local-log="" remote-log=""/>
  <Facility name="tvod_manager" local-log="" remote-log=""/>
  <Facility name="utils" local-log="" remote-log=""/>
</LogConfig>

<LogConfig>
  <Server ServerID="561" GroupID="1991"/>
  <Facility name="avsdb" local-log="" remote-log=""/>
  <Facility name="bwm" local-log="" remote-log=""/>
  <Facility name="c2k" local-log="" remote-log=""/>
  <Facility name="collectd" local-log="" remote-log=""/>
  <Facility name="dnsresolver" local-log="" remote-log=""/>
  <Facility name="http" local-log="" remote-log=""/>
  <Facility name="ns" local-log="" remote-log=""/>
  <Facility name="prodassert" local-log="" remote-log=""/>
  <Facility name="pvr_asset_manager" local-log="" remote-log=""/>
  <Facility name="statsd" local-log="" remote-log=""/>
  <Facility name="tvod_manager" local-log="" remote-log=""/>
  <Facility name="utils" local-log="" remote-log=""/>
</LogConfig>

<LogConfig>
  <Server ServerID="562" GroupID="1991"/>
  <Facility name="avsdb" local-log="" remote-log=""/>
  <Facility name="bwm" local-log="" remote-log=""/>
  <Facility name="c2k" local-log="" remote-log=""/>
  <Facility name="collectd" local-log="" remote-log=""/>
Maintain Section Configuration API

The Maintain Section Configuration APIs offers the ability to download or upload an XML file that contains the maintain section configuration settings by way of an API call using any REpresentational State Transfer (REST) client.

Following are the Maintain section configuration APIs:

- System Thresholds
- Application Configuration
- CDSM/VVIM Setup
- System Configuration
- Database Configuration
- Configuration Generator
- ID Management
- Stream Monitor Listener
- Maintain Section Backup
System Thresholds

The maintain section System Thresholds list request-response message returns system Thresholds configuration.

GET ALL System Thresholds Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl http://<cdsm_host>/api/services/maintain/software/id/systemthresholds

Response Example

```xml
```

POST System Thresholds Configuration

Following are the request components:

Request Type: HTTP POST

Request

curl --form upload=@/root/systemthresholds.xml --form press=OK "http://<cdsm_host>/api/services/maintain/software/id/systemthresholds"

Request Body: XML

Request Example

Following is the XML body for the POST request to upload an individual system Thresholds configuration settings:

```xml
```
Application Configuration

The maintain section Application Configuration list request-response message returns a list of all application configurations.

GET ALL Application Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl http://<cdsm_host>/api/services/maintain/software/id/applicationconfigs

Response

If the request succeeds, the list element is returned in the XML body response containing the list of Application Configuration.

Response Example

```xml
```

POST System Application Configuration

Following are the request components:

Request Type: HTTP POST

Request

curl --form upload=@/root/applicationconfigs.xml --form press=OK "http://<cdsm_host>/api/services/maintain/software/id/applicationconfigs"

Request Body: XML

Request Example

Following is the XML body for the POST request to upload an individual maintain section application configuration settings:

```xml
<?xml version="1.0" encoding="UTF-8"?><ApplicationConfigurations xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><ApplicationConfig Application='Barker' StreamDeliveryMode='Active-Active'><DeliveryServers><Server
```
CDSM/VVIM Setup

The maintain section CDSM/VVIM Setup list request-response message returns a list of all CDSM/VVIM Setup configurations.

GET ALL CDSM/VVIM Setup Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl http://<cdsm_host>/api/services/maintain/software/id/cdsmsetup

Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ASMSetupConfigList
```
POST System CDSM/VVIM Configuration

Following are the request components:

Request Type: HTTP POST

Request

curl --form upload=@/root/cdsmsetup.xml --form press=OK
"http://<cdsm_host>/api/services/maintain/software/id/cdsmsetup"

Request Body: XML

Request Example

Following is the XML body for the POST request to upload an individual maintain section CDSM/VVIM Setup configuration settings:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ASMSetupConfigList
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><ASMSetupConfig>
<DeployedServerVersion CServerVersion='4.x'/>
<RoleFailoverOnDBConnectivityLoss FailoverSupport='On'/>
<RetryCount='30'/>
<FailoverOnBMSConnectivityLoss FailoverSupport='On'/>
<StreamSteering Mode='Silo'/>
<DeploymentNetworkConfig Networked='L3'/>
<NICBonding Support='Disabled'/>
<StreamFailureMode InstallationType='ISA 4.x'/>
<StreamDestination Type='IPTV' Options='NAT'/>
<SplunkSupport Support='Disabled'/>
<ParentChildServiceGroups Mode='Disabled'/>
<RedirectServer Support='Off'/>
<VBOConfiguration Support='Enabled'/>
<RedirectServerSupport='Off'/>
<AssetScaleCoexistenceSupport Support='Enabled'/>
<vBulkUploadExportConfig Support='Enabled'/>
<TrickModeCapture Support='Disabled'/>
<StreamReportCapture='Enabled'/>
<FailIngestTuning Support='Disabled'/>
<FailoverSupport='On'/>
<RedirectServer Support='Off'/>
<PassiveStorageSupport 'Disabled'/>
<ManageRecorders Support='Disabled'/>
<ATISC2Setup Mode='None'/>
<TVODSetup Support='Disabled'/>
<IngestResiliency Support='Enabled'/>
<VODErrorRepair Support='Disabled' ActivationKey=''/>
<MediaScheduler Support='Off' ActivationKey=''/>
<RealTimeCapture Type='Real Time Capture OFF'/>
<VirtualVideoInfrastructure VVIOptions='Enabled' ManagementSystemRole='Stream Manager' CacheFillProtocol='CCP'
VaultCacheMgrIP='10.197.103.8'
StreamDomainName='VHO1'/>
<ContentStorage ContentStorage='Shared'/>
<RemoteSetupControlService Support='Enabled'
GlobalSourceIP='192.161.101.200'
GlobalSourcePort='4000'
MinimumStreamsPerPort='0'
SingleSessionTimeout='10'
SessionsPerControl='1000'/>
<PlayoutProcessor Support='Disabled'/>
<PlayoutScheduler Support='Disabled' 'LocalizedEPGExtensions='Off'/>
<IngestManager Support='Off' UpStreamIngestSystem='Generic' ActivationKey=''/>
<IngestSteering Support='On'/>
<HealthMonitoring Support='On'/></ASMSetupConfigList>
```
System Configuration

The maintain section System Configuration list request-response message returns a list of all system configurations.

GET ALL System Configuration

Following are the request components:

Request Type: HTTP GET

Request

curl http://<cdsm_host>/api/services/maintain/software/id/systemconfigs

Response Example
<?xml version="1.0" encoding="UTF-8"?><SystemConfigs xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><SystemConfig GroupMap='RTSP/FSI' ServerGroupMap='RTSP/FSI'><PopularityBasedCaching PopularityHalfLife='12'/><ContentListingConfig Value='100'/></SystemConfig></SystemConfigs>

POST System Configuration

Following are the request components:

Request Type: HTTP POST

Request

curl --form upload=@/root/systemconfigs.xml --form press=OK "http://<cdsm_host>/api/services/maintain/software/id/systemconfigs"

Request Body: XML

Request Example
Following is the XML body for the POST request to upload an individual maintain section System configuration settings:

<?xml version="1.0" encoding="UTF-8"?><SystemConfigs xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><SystemConfig GroupMap='RTSP/FSI' ServerGroupMap='RTSP/FSI'><PopularityBasedCaching PopularityHalfLife='12'/><ContentListingConfig Value='100'/></SystemConfig></SystemConfigs>
Database Configuration

The maintain section Database Configuration list request-response message returns a list of all database configurations.

GET ALL Database Configuration

Following are the request components:

Request Type: HTTP GET

Request

```
curl http://<cdsm_host>/api/services/maintain/software/id/databaseconfigs
```

Response Example

```
```

POST Database Configuration

Following are the request components:

Request Type: HTTP POST

Request

```
curl --form upload=@/root/databaseconfigs.xml --form press=OK "http://<cdsm_host>/api/services/maintain/software/id/databaseconfigs"
```

Request Body: XML

Request Example

Following is the XML body for the POST request to upload an individual maintain section Database configuration settings:

```
```
Configuration Generator

The maintain section Configuration Generator list request-response message returns a list of all configurations Generator.

GET ALL Configuration Generator

Following are the request components:

Request Type: HTTP GET

Request

curl http://<cdsm_host>/api/services/maintain/software/id/configgenerator

Response

If the request succeeds, the list element is returned in the XML body response containing the list of Configuration Generator.

Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ConfigurationGeneratorConfigList
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
<ConfigurationGeneratorConfig StreamDomainName='Market3' StreamManagerIP='10.197.86.202'
StreamGroupIDStart='20001' StreamGroupIDEnd='30000' ServerIDStart='1251'
ServerIDEnd='1500' SetupIDStart='3' SetupIDEnd='4'/>
<ConfigurationGeneratorConfig StreamDomainName='Market4' StreamManagerIP='10.197.102.12'
StreamGroupIDStart='40001' StreamGroupIDEnd='50000' ServerIDStart='1751'
ServerIDEnd='2000' SetupIDStart='7' SetupIDEnd='8'/>
<ConfigurationGeneratorConfig StreamDomainName='Market5' StreamManagerIP='10.197.102.25'
StreamGroupIDStart='50001' StreamGroupIDEnd='60000' ServerIDStart='2001'
ServerIDEnd='2250' SetupIDStart='9' SetupIDEnd='10'/>
<ConfigurationGeneratorConfig StreamDomainName='Market6' StreamManagerIP='10.197.102.26'
StreamGroupIDStart='70001' StreamGroupIDEnd='80000' ServerIDStart='2501'
ServerIDEnd='2750' SetupIDStart='13' SetupIDEnd='14'/>
<ConfigurationGeneratorConfig StreamDomainName='Market7' StreamManagerIP='10.197.102.29'
StreamGroupIDStart='60001' StreamGroupIDEnd='70000' ServerIDStart='2251'
ServerIDEnd='2500' SetupIDStart='11' SetupIDEnd='12'/>
</ConfigurationGeneratorConfigList>
```

POST Configuration Generator

Following are the request components:

Request Type: HTTP POST

Request

curl --form upload=@/root/configgenerator.xml --form press=OK
"http://<cdsm_host>/api/services/maintain/software/id/configgenerator"
Request Body: XML

Request Example
Following is the XML body for the POST request to upload an individual maintain section configuration generator settings:

```xml
<?xml version="1.0" encoding="UTF-8"?><ConfigurationGeneratorConfigList
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
<ConfigurationGeneratorConfig StreamDomainName='Market3' StreamManagerIP='10.197.86.202'
StreamGroupIDStart='20001' StreamGroupIDEnd='30000' ServerIDStart='1251'
ServerIDEnd='1500' SetupIDStart='3' SetupIDEnd='4'/>
<ConfigurationGeneratorConfig StreamDomainName='Market4' StreamManagerIP='10.197.102.12'
StreamGroupIDStart='40001' StreamGroupIDEnd='50000' ServerIDStart='1751'
ServerIDEnd='2000' SetupIDStart='7' SetupIDEnd='8'/>
<ConfigurationGeneratorConfig StreamDomainName='Market5' StreamManagerIP='10.197.102.25'
StreamGroupIDStart='50001' StreamGroupIDEnd='60000' ServerIDStart='2001'
ServerIDEnd='2250' SetupIDStart='9' SetupIDEnd='10'/>
<ConfigurationGeneratorConfig StreamDomainName='Market6' StreamManagerIP='10.197.102.26'
StreamGroupIDStart='70001' StreamGroupIDEnd='80000' ServerIDStart='2501'
ServerIDEnd='2750' SetupIDStart='13' SetupIDEnd='14'/>
<ConfigurationGeneratorConfig StreamDomainName='Market7' StreamManagerIP='10.197.102.29'
StreamGroupIDStart='60001' StreamGroupIDEnd='70000' ServerIDStart='2251'
ServerIDEnd='2500' SetupIDStart='11' SetupIDEnd='12'/>
</ConfigurationGeneratorConfigList>
```

ID Management

The maintain section ID Management list request-response message returns a list of all configurations ID management.

GET ALL ID Management

Following are the request components:

Request Type: HTTP GET

Request

```bash
curl http://<cdsm_host>/api/services/maintain/software/id/idmanagement
```

Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?><IDManagementConfigs
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><IDManagementConfig
GroupIDStart='20001' ServerIDStart='1251' SetupIDStart='3'/></IDManagementConfigs>
```

POST ID Management

Following are the request components:

Request Type: HTTP POST

Request

```bash
curl --form upload=@/root/idmanagement.xml --form press=OK
http://<cdsm_host>/api/services/maintain/software/id/idmanagement
```
Request Body: XML

Request Example
Following is the XML body for the POST request to upload an individual maintain section configuration
ID Management settings:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<IDManagementConfigs
  xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
  <IDManagementConfig
    GroupIDStart='20001' ServerIDStart='1251' SetupIDStart='3'/>
</IDManagementConfigs>
```

Stream Monitor Listener

The maintain section Stream Monitor Listener list request-response message returns a Stream Monitor
Listener configurations.

GET ALL Stream Monitor Listener

Following are the request components:

Request Type: HTTP GET

Request

```bash
curl http://<cdsm_host>/api/services/maintain/software/id/streammonitorlistener
```

Response Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<StreamMonitorListenerConfigs
  xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
  <StreamMonitorListenerConfig
    Support='Enabled'
    UDP IP='10.197.103.153' Port='6001'/>
  <StreamMonitorListenerConfig
    UDP IP='10.197.103.144' Port='6002'/>
</StreamMonitorListenerConfigs>
```

POST Stream Monitor Listener

Following are the request components:

Request Type: HTTP POST

Request

```bash
curl --form upload=@/root/streammonitorlistener.xml --form press=OK
  "http://<cdsm_host>/api/services/maintain/software/id/streammonitorlistener"
```

Request Body: XML

Request Example

Following is the XML body for the POST request to upload an individual maintain section configuration
Stream Monitor Listener settings:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<StreamMonitorListenerConfigs
  xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs">
  <StreamMonitorListenerConfig
    Support='Enabled'
    UDP IP='10.197.103.153' Port='6001'/>
  <StreamMonitorListenerConfig
    UDP IP='10.197.103.144' Port='6002'/>
</StreamMonitorListenerConfigs>
```
Maintain Section Backup

The maintain section Backup list request-response message returns a list of all maintain section configurations.

GET ALL Maintain Section Backup

Following are the request components:

Request Type: HTTP GET

Request
curl http://<cdsm_host>/api/services/maintain/software/id/backup

Response Example

  <ASMSSetupConfigList>
    <ASMSSetupConfig>
      <DeployedCServerVersion CServerVersion='4.x'/>
      <StreamFailoverSupport FailoverSupport='On'/>
      <RoleFailoverOnDBConnectivityLoss FailoverSupport='On'/>
      <RetryCount='30'/>
    </ASMSSetupConfig>
  </ASMSSetupConfigList>
</MaintainConfiguration>
POST ID Maintain Section Backup

Following are the request components:

Request Type: HTTP POST

Request

curl --form upload=@/root/restore.xml --form press=OK
"http://<cdsm_host>/api/services/maintain/software/id/restore"

Request Body: XML

Request Example

Following is the XML body for the POST request to upload an individual maintain section backup configuration settings:

```xml
<?xml version="1.0" encoding="UTF-8"?><MaintainConfiguration
xmlns="http://www.cisco.com/schemas/VCPBU/CDS-TV/R0/ciscowebsvcs"><ASMSetupConfigList
```
Maintain Section Configuration API

Support='On'"/>
</ASMSetupConfig>
</ASMSetupConfigList>
</SystemConfigs>

<
</SystemConfig>
</SystemThresholds>
</SystemThresholds>
</IDManagementConfig>
</IDManagementConfig>
</DatabaseConfigs>
</DatabaseConfigs>
</StreamMonitorListenerConfigs>
</StreamMonitorListenerConfigs>
</MaintainConfiguration>

Support='Disabled'></StreamMonitorListenerConfig>
</StreamMonitorListenerConfigs>
</DatabaseConfigs>
</DatabaseConfigs>
</StreamMonitorListenerConfigs>
</StreamMonitorListenerConfigs>
</MaintainConfiguration>