Cisco Internet Streamer CDS 2.6 API Guide

November 2011

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

Text Part Number: OL-23532-01

Cisco Internet Streamer CDS 2.6 API Guide

CONTENTS

Preface

This preface describes who should read the *Cisco Internet Streamer CDS 2.6 API Guide*, how it is organized, and its document conventions. It contains the following sections:

Document Revision History

This section records technical changes to this document. The table shows the document revision number for the change, the date of the change, and a brief summary of the change.

Table 1 Document Revision History

Revision	Date	Change Summary
OL-23532-01	November 2011	Initial release.

Audience

This application program interface (API) guide is written for the knowledgeable application programmer who understands the basic architecture of the CDS software product and Java servlets. The user should be fluent in the Java programming language and have prior practical experience developing content networking solutions. This guide is not intended to direct the user in how to program in the Java language and limits itself to describing how related CDS software servlets are used.

Document Organization

This API guide includes the following chapters:

Chapter or Appendix	Title	Description
Chapter 1	Introduction to Cisco CDS Software APIs	Provides an introduction to the CDS software application program interfaces.
Chapter 2	Replication Status APIs	Describes the Replication Status API and the servlet actions it performs.
Chapter 3	Provisioning APIs	Describes the Delivery Service Provisioning API, Location Provisioning API, Service Engine Provisioning API, and Program API and the servlet actions they perform.
Chapter 4	Listing APIs	Describes the Listing API and the servlet actions it performs.
Chapter 5	Statistics APIs	Describes the Monitoring Statistics API and Streaming Statistics API and the servlet actions they perform.
Chapter 6	File Management APIs	Describes the File Management API and the servlet actions it performs.
Chapter 7	Request Routing Engine API	Describes the Request Routing Engine API.
Chapter 8	Proximity Engine SOAP APIs	Describes the Proximity Engine APIs.
Appendix A	Program Files in the CDS Software	Describes the attributes of program files and examples for different program types.

Document Conventions

This API guide uses basic conventions to represent text and table information.

Convention	Description
boldface font	Commands, keywords, and button names are in **boldface**.
italic font	Variables for which you supply values are in *italics*. Directory names and filenames are also in italics.
`screen` font	Terminal sessions and information the system displays are printed in `screen` font.
`boldface screen` font	Information you must enter is in **`boldface screen`** font.
`italic screen` font	Variables you enter are printed in *`italic screen`* font.
string	Defined as a nonquoted set of characters. For example, when setting a community string for SNMP to “public,” do not use quotation marks around the string, or the string will include the quotation marks.

Convention	Description
vertical bars (|)	Vertical bars separate alternative, mutually exclusive, elements.
< >	Variable for which you supply a value.
{ }	Elements in braces are required elements.
[ ]	Elements in square brackets are optional.
{**x** | **y** | **z**}	Required keywords are grouped in braces and separated by vertical bars.
[**x** | **y** | **z**]	Optional keywords are grouped in brackets and separated by vertical bars.
[{ }]	Braces within square brackets indicate a required choice within an optional element.

Note Means *reader take note*. Notes contain helpful suggestions or references to materials not contained in the manual.

Tip Means *the following information will help you solve a problem*. The tips information might not be troubleshooting or even an action, but could be useful information, similar to a Timesaver.

Related Publications

These documents provide complete information about the CDS and are available from the Cisco.com site:

- *Cisco Internet Streamer CDS 2.6 Software Configuration Guide*
- *Cisco Internet Streamer CDS 2.6 Software Upgrade Guide*
- *Cisco Internet Streamer CDS 2.6 Quick Start Guide*
- *Cisco Internet Streamer CDS 2.6 Command Reference*
- *Cisco Internet Streamer CDS 2.6 Alarms and Error Messages Guide*
- *Release Notes for Cisco Internet Streamer CDS 2.6*
- *Cisco Content Delivery Engine 205/220/250/420 Hardware Installation Guide*
- *Cisco Content Delivery System 2.x Documentation Roadmap*
- *Regulatory Compliance and Safety Information for Cisco Content Delivery Engines*
- *Open Sources Used in CDS IS Release 2.6*

You can access the software documents at the following URL:

http://www.cisco.com/en/US/products/ps7127/tsd_products_support_series_home.html

You can access the hardware documents at the following URL:

http://www.cisco.com/en/US/products/ps7126/tsd_products_support_series_home.html

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:

http://www.cisco.com/en/US/docs/general/whatsnew/whatsnew.html

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.

CHAPTER 1

Introduction to Cisco CDS Software APIs

Cisco Content Delivery System (CDS) software provides HyperText Transport Protocol Secure (HTTPS) application program interfaces (APIs) for monitoring and managing the acquisition and distribution of content.

The Request Routing Engine on the Service Router implements an API that allows another platform's software client to make queries, in the form of an HTTP request, to the Request Routing Engine about which Service Engine the Request Routing Engine selects.

The Proximity Engine on the Service Router implements a rating API on its Simple Object Access Protocol (SOAP) interface. The rating API calculates the proximity of a group of proximity target addresses (PTAs) to a proximity source address (PSA).

This chapter contains the following sections:

HTTPS APIs

The CDS software provides ten sets of HTTPS APIs:

- Replication Status
- Delivery Service Provisioning
- Location Provisioning
- Service Engine Provisioning
- Program
- URL Management
- Listing
- Monitoring Statistics
- Streaming Statistics
- File Management

These HTTPS APIs are Java servlets whose return outputs are generated in XML format. CDS software uses these servlets to monitor and modify specified content acquisition and distribution parameters. Table 1-1 describes these APIs. For most API actions, a unique website and delivery service name must be provided to the API so that the delivery service can be located.

Table 1-1 Cisco CDS Software HTTPS APIs

API	Description
Replication Status	Returns a list of delivery services, Service Engines, or contents, and for each delivery service, an indication whether replication of content for the specified delivery service is complete or not.
Provisioning APIs	Provides the CDSM[1] with CDS delivery service, location, and Service Engine information. • Delivery Service Provisioning API—Monitors and modifies CDS network delivery services. • Location Provisioning API—Creates, modifies, or deletes a CDS network location object. • Service Engine Provisioning API—Activates, locates, or deletes a specified Service Engine. • Program API—Creates, modifies, validates, or deletes programs, and assigns or unassigns Service Engines and delivery services to programs. • URL Management API—Deletes single or multiple content objects.
Listing API	Obtains object information from the local embedded database.
Monitoring Statistics API	Obtains monitoring statistics data about a single Service Engine or all the Service Engines in the CDS network.
Streaming Statistics API	Reports WMT[2], HTTP, Movie Streamer, and Flash Media data collected from the Service Engines or device groups and sends this data to the CDSM. Data obtained with the Streaming Statistics API can be saved and a customized report generated.
File Management API	Performs file management functions on Coverage Zone, NAS[3], Service Rules, and CDN Selector files, including registering, validating, refetching, modifying, and deleting the files. Applies Coverage Zone and CDN Selector files to SRs[4]. **Note** NAS is only supported in lab integrations as proof of concept.

1. CDSM = Content Delivery System Manager.
2. WMT = Windows Media Technology.
3. NAS = network attached storage.
4. SRs = Service Routers.

CDS software also provides authentication, authorization, and accounting (AAA) functions to support users who access external servers and local databases. Authentication verifies the identity and IP address of a user, authorization permits or denies access privileges for authenticated users in the CDS network, and accounting logs authorized usage of network services. These AAA functions are enforced by the APIs so that user credentials must be validated before an API can be executed.

Calling the HTTPS APIs

You can execute the CDS software APIs interactively or through a caller program. API calls must follow the correct syntax. If the user credential is invalid or the syntax is incorrect, the API is not executed. If a user error occurs, a warning is returned that explains the nature of the error along with the syntax of the particular API.

Note All API parameters are case sensitive.

Interactive Calls

Use a browser or Lynx command to execute the API interactively. The user is prompted to enter a username and password for authentication and authorization. Once the user is validated, the API is executed. If the execution is successful and an output is to be returned as a result, the output is displayed in the browser if a browser was used to make the API call, or the output can be redirected to a file if a Lynx command was used to make the API call. If the execution is unsuccessful, an error message is returned.

Programmed Calls

To make an API call, write a caller program using an HTTPS request. The username and password are set in the HTTPS request for AAA validation. If validation and execution are successful and an output is to be returned as a result, the output or a success code is returned. If the execution is unsuccessful, a failure code is returned.

Sample Java Program

The following is a sample Java client program that requires two Simple API for XML (SAX) parsing APIs. This sample code requires the “org.xml.sax.*” API and “org.xml.sax.helpers.*” API for the parser and the HTTPS URL package for the connection.

```
package testing.download.client;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.MalformedURLException;
import java.net.URL;
import javax.net.ssl.HostnameVerifier;
import javax.net.ssl.HttpsURLConnection;
import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLSession;
import javax.net.ssl.TrustManager;

import javax.net.ssl.X509TrustManager;


public class Client
{

    public static void main (String[] args)
    {
        /**
```

```
         * Setting parameters for the API call
         */
        String cdmAddress_ = "cds-demo-cdsm.cds.cisco.com";
        String cdmPort_ = "8443";
        String taskAPI_ = "com.cisco.unicorn.ui.ListApiServlet";
        String action_ = "getDeliveryServices";
        String channelId_ = "all";
        String urlString_ = "https://" + cdmAddress_ + ":" + cdmPort_ + "/servlet/" +
                    taskAPI_+"?action=" + action_ + "&param=" + channelId_;
        String userName_ = "admin";
        String password_ = "default";

        /**
          * Install the all-trusting trust manager
          */
        try {
            SSLContext sc = SSLContext.getInstance("SSL");
            sc.init(null, trustAllCerts, new java.security.SecureRandom());
            HttpsURLConnection.setDefaultSSLSocketFactory(sc.getSocketFactory());
        }
        catch (Exception e) {
            System.out.println("Printing Exception Message "+e);
        }

        /**
         * Insert the credentials
         */
        String sAuth = userName_+":"+password_;
        String sEncodedAuth = new sun.misc.BASE64Encoder().encode(sAuth.getBytes());
        /**
          * Create the HTTPS Connection
         */
        HttpsURLConnection conn = null;
        try {
            URL url = new URL(null, urlString_ );

            System.out.println(url.toString());

            conn = (HttpsURLConnection)url.openConnection();
            conn.setRequestProperty("Authorization", "Basic " + sEncodedAuth);
            conn.setHostnameVerifier(new newHostNameVerifier());
            conn.setDoInput(true);
            conn.setDoOutput(true);
            conn.setUseCaches(false);
            conn.setRequestProperty("Connection", "Keep-Alive");
            conn.setRequestMethod("GET");

        }
        catch (MalformedURLException ex)
        {
            System.out.println("Printing Exception Message "+ex);
        }
        catch (IOException ioexception)
        {
            System.out.println("Printing Exception Message "+ioexception);
        }

        /**
         * Handling the response from CDSM
         */
        try
        {
            BufferedReader inStreamReader = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
```

```
            String str;
            while (( str = inStreamReader.readLine())!= null)
            {
                System.out.println("Response from CDSM : ");
                System.out.println(str);
            }
            inStreamReader.close();
        }
        catch (IOException ioexception)
        {
            System.out.println("Printing Exception Message "+ioexception);
        }
    }

/**
 * Create a trust manager that does not validate certificate chains
 */
    private static TrustManager[] trustAllCerts = new TrustManager[]{
            new X509TrustManager() {
            public java.security.cert.X509Certificate[] getAcceptedIssuers() {
                    return null;
                }
                public void checkClientTrusted(
                    java.security.cert.X509Certificate[] certs, String authType) {

                }
                public void checkServerTrusted(
                    java.security.cert.X509Certificate[] certs, String authType) {

                }
            }
    };

    private static class newHostNameVerifier implements HostnameVerifier {

        /**
         * ignore hostname checking
         */
        public boolean verify(String hostname, SSLSession session) {
          return true;
        }
    }

}
```

API Error Messages

When a server error occurs while the APIs are invoked, an XML-formatted message is returned. For example, when Internal Server Error—500 occurs, the client sees the following output:

```
<?xml version="1.0"?>
<Error>
<message status="fail" message="Internal Server Error —5 00"/>
</Error>
```

The following common errors are supported in the message syntax:

- Bad Request—400
- Authorization Required—401

- Forbidden—403
- File Not Found—404
- Request Timeout—408
- Internal Server Error—500

Typically, APIs return error messages when API execution fails. If the execution is successful, APIs do not return any error messages. However, APIs may return warning messages even when the execution is successful.

APIs use numeric error and warning codes. Table 1-2 describes the generic numeric codes used for errors and warnings. Table 1-3 describes some of the numeric codes used by the Program and File Management in API errors and warnings.

Table 1-2 Numeric Codes for Errors and Warnings in APIs

Error or Warning Code	Description
0	None
1	Syntax error
2	Input error
3	Constraint error
4	Input warnings

Table 1-3 Numeric Codes for Errors and Warnings in the Program and File Management APIs

Error or Warning Code	Description
101	Unable to fetch the file
102	File syntax error
103	Invalid value in the file
104	Related system error
105	Program file unused input - Warning

For example, when you enter the following URL to execute an API to delete a selected type of program:

https://<cdsm:port>/servlet/com.cisco.unicorn.ui.ProgramApiServlet?action=deletePrograms&program=type=wmt

and no programs of that type exist, the API returns the subsequent warning.

```
<?xml version="1.0" ?>
<programApi action="deletePrograms">
<message status="success" message="The program(s) are deleted." />
<warning code="4" message="No Program(s) that matched the request were found" />
</programApi>
```

Similarly, when you enter the following URL to execute an API to delete a delivery service:

https://<cdm:port>/servlet/com.cisco.unicorn.ui.ChannelApiServlet?action=deleteDeliveryServices&deliveryService=Channel_333

with an invalid delivery service ID, the API returns the subsequent error.

```
<?xml version="1.0" ?>
```

```
<channelProvisioning action="deleteDeliveryServices">
<message status="fail" message="Input Error: Cannot locate delivery service using delivery
service ID Channel_333" />
<error code="2" message="Input Error: Cannot locate delivery service using delivery
service ID Channel_333" />
</channelProvisioning>
```

API Tasks

The following sections provide a brief list of tasks performed by the Replication Status, Provisioning, Listing, and Statistics APIs.

Replication Status API

The Replication Status API performs one or more of the following tasks when executed:

- Obtains the replication status of content on specified delivery services
- Obtains the replication status of content for all Service Engines assigned to the specified delivery service
- Obtains the replication status of content for all delivery services assigned to the specified Service Engine
- Lists all replicated items of a specified Service Engine on a specified delivery service, with or without search criteria
- Lists all nonreplicated items of a specified Service Engine on a specified delivery service, with or without search criteria
- Lists all content items of a Service Engine on a specified delivery service, with or without search criteria

Provisioning APIs

The Provisioning APIs include the Delivery Service Provisioning API, Location Provisioning API, Service Engine Provisioning API, and Program API.

Delivery Service Provisioning API

The Delivery Service Provisioning API performs one or more of the following tasks when executed:

- Creates delivery services
- Adds a Manifest file to a specified delivery service
- Assigns Service Engines to a specified delivery service
- Assigns an IP address of a Service Engine to a specified delivery service
- Immediately fetches the Manifest file
- Modifies delivery service settings
- Modifies Manifest file settings
- Removes Service Engines from a specified delivery service
- Removes IP addresses of a Service Engine from a specified delivery service

- Removes device groups from a specified delivery service
- Deletes delivery services
- Creates content origins
- Modifies content origin settings
- Deletes content origins
- Applies Service Rule files to delivery services

Location Provisioning API

The Location Provisioning API performs one or more of the following tasks when executed:

- Creates a specified location
- Modifies a specified location
- Deletes a specified location

Service Engine Provisioning API

The Service Engine Provisioning API performs one or more of the following tasks when executed:

- Activates a specified Service Engine
- Changes the location of a specified Service Engine
- Deletes a specified Service Engine

Program API

The Program API performs one or more of the following tasks when executed:

- Creates a program file
- Validates a program file
- Assigns delivery services to a specified program
- Assigns Service Engines to a specified program
- Fetches a program file
- Modifies a program file
- Removes delivery services from a specified program
- Removes Service Engines from a specified program

URL Management API

The URL Management API performs one or more of the following tasks when executed:

- Removes content items of delivery service by single URL
- Removes content items or delivery service by URL batch file, which contains a set of URLs

Listing API

The Listing API performs one or more of the following tasks when executed:

- Lists selected content origin names or lists every content origin
- Lists selected delivery service names and related content origin IDs or lists every delivery service
- Lists selected Service Engine names or lists every Service Engine
- Lists the location of the specified Service Engines
- Lists selected cluster names or lists every cluster
- Lists selected device group names or lists every device group
- Lists the status of a device or device group
- Lists an object, based on its string ID
- Lists an object, based on its name
- Lists all programs specified
- Lists all multicast addresses currently in use by programs
- Lists all multicast addresses currently in use
- Lists the multicast address range reserved for programs

Statistics API

The Statistics APIs include the Monitoring Statistics API and the Streaming Statistics API.

Monitoring Statistics API

The Monitoring Statistics API performs one or more of the following tasks when executed:

- Obtains monitoring statistics for each Service Engine
- Obtains monitoring statistics for all the Service Engines in a location
- Obtains monitoring statistics for all the Service Engines in the CDS network

Streaming Statistics API

The Streaming Statistics API performs one or more of the following tasks when executed:

- Reports HTTP statistics for each Service Engine or device group
- Reports Movie Streamer statistics for each Service Engine or device group
- Reports WMT statistics for each Service Engine or device group

File Management API

The File Management API performs one or more of the following tasks when executed:

- Displays a list of all the file types that can be registered with the CDSM
- Registers an external file with the CDSM by either uploading a file from any location that is accessible from your PC or by importing a file from an external server
- Validates a file before or after registering it with the CDSM
- Modifies the metadata associated with a registered file
- Immediately refetches a registered file from an external server
- Deletes a registered file from the CDSM
- Lists the details of a specific file or lists all files of a specific file type
- Assigns a Coverage Zone file to an SR or unassigns a Coverage Zone file from an SR
- Associates a CDN Selector file with an SR or disassociates a CDN Selector file from an SR

Request Routing Engine API

The Request Routing Engine API returns the name of the Service Engine which the Request Routing Engine selects as the best Service Engine on the basis of a Client IP address and a URL provided in the calling API.

Note The Request Routing Engine API does not support service-aware routing.

Proximity Engine SOAP API

The Proximity Engine exposes a rating API on its SOAP interface to a proximity client (SR). SOAP is an XML-based messaging protocol for invoking remote procedures by sending XML messages over application layer protocols (for example, HTTP). The SR leverages the rate API to determine the network difference between a PSA and a PTA in order to choose the PTA within closest proximity to the PSA. PTAs returned by the rate API are ranked in ascending order based on the rating each has received. If an error occurs while the rating API is invoked, an XML-formatted fault message is returned. When the Proximity Engine does not consider itself the most appropriate Proximity Engine to service the request, an XML-formatted redirect fault messages is returned. In this message, the Proximity Engine redirects the proximity client to a set of Proximity Engines it considers more appropriate.

Note The Proximity Engine API is available on the CDE205 and CDE220-2G2 platforms.

CHAPTER 2

Replication Status APIs

This chapter describes the Replication Status API and the servlet actions it performs. The Replication Status API returns a list of delivery services, Service Engines, or contents, and for each delivery service, an indication whether replication of content for the specified delivery service is complete or not.

Replication Status API Actions

The Replication Status API is the ReplicationStatusApiServlet.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.RepStatusApiServlet…

This servlet performs one or more of the following actions:

- getDeliveryServices
- getSEsOfDeliveryService
- getDeliveryServicesOfSE
- getReplicatedContent
- getNonReplicatedContent
- getContent
- getStatusOfContentItems
- getStatusOfContentItemInDeliveryService

getDeliveryServices

Obtains the status of content replication of specified delivery services.

Parameter

Either a list of delivery service IDs or the keyword **all** is required.

Return

A list of the delivery services, and for each delivery service, a flag indicating whether replication for the specified delivery service is complete or incomplete.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.RepStatusApiServlet?
action=getDeliveryServices&deliveryService=[all | <deliveryService_ID>,<deliveryService_ID>,…]

getSEsOfDeliveryService

Obtains the status of content replication for all Service Engines assigned to the specified delivery service.

Parameter

- Delivery service ID (required)
- Refetch (optional)—The default is false.

  If refetch is set to true, a background request to obtain a newly updated status is sent to all Service Engines assigned to this delivery service. To view the newly available information, the user must call the API again after several minutes without a refetch.

Return

A list of all Service Engines assigned to a specified delivery service and, for each specified Service Engine, a flag whether replication for the specified Service Engine is complete or incomplete.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.RepStatusApiServlet?
action=getSEsOfDeliveryService&deliveryService=<deliveryService_ID>[&refetch=<true | false>]

getDeliveryServicesOfSE

Obtains the status of content replication for all delivery services assigned to the specified Service Engine.

Parameter

- Service Engine ID (required)
- Refetch (optional)—The default is false.

  If refetch is set to true, a background request to obtain a newly updated status is sent to all Service Engines assigned to this delivery service. To view the newly available information, the user must call the API again after several minutes without a refetch.

Return

A list of all delivery services assigned to a specified Service Engine and, for each delivery service, a flag whether replication for the specified delivery service is complete or incomplete.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.RepStatusApiServlet?
action=getDeliveryServicesOfSE&se=<SE_ID>[&refetch=<true | false>]

getReplicatedContent

Lists all replicated items of a specified Service Engine on a specified delivery service, with or without search criteria.

Parameter

- Service Engine ID (required)
- Delivery service ID (required)
- Search criteria (optional)

  One or more content names or patterns must each be separated by a comma. Patterns can contain the wildcards * or ?.
- Refetch (optional)—The default is false.

  If refetch is set to true, a background request to retrieve the content is issued. The updated information is cached on the CDSM and can be retrieved in the next call.

Return

A list of all replicated content items on a specified Service Engine for a specified delivery service that matches the search criteria, if the search criteria have been specified.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.RepStatusApiServlet?action=getReplicatedContent&se=<SE_ID>&deliveryService=<deliveryService_ID>[&criteria=<criteria>][&refetch=<true | false>]

getNonReplicatedContent

Lists all nonreplicated items of a specified Service Engine on a specified delivery service, with or without search criteria.

Parameter

- Service Engine ID (required)
- Delivery service ID (required)
- Search criteria (optional)

  One or more content names or patterns must each be separated by a comma. Patterns can contain the wildcards * or ?.
- Refetch (optional)—The default is false.

  If refetch is set to true, a background request to retrieve the content is issued. The updated information is cached in the CDSM and can be retrieved in the next call.

Return

A list of all content items that are not replicated on a specified Service Engine of a specified delivery service that matches the search criteria, if search criteria have been specified. The list includes content items that are yet to be replicated, are in the process of being replicated, or have failed to be replicated.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.RepStatusApiServlet?action=getNonReplicatedContent&se=<SE_ID>&deliveryService=<deliveryService_ID>[&criteria=<criteria>][&refetch=<true | false>]

getContent

Lists all content items of a Service Engine on a specified delivery service, with or without search criteria.

Parameter

- Service Engine ID (required)
- Delivery service ID (required)
- Search criteria (optional)

  One or more content names or patterns must each be separated by a comma. Patterns can contain the wildcards * or ?.
- Refetch (optional)—The default is false.

  If refetch is set to true, a background request to retrieve the content is issued. The updated information is cached on the CDSM and can be retrieved in the next call.

Return

A list of all content items on the Service Engine of a specified delivery service that matches the specified criteria, if search criteria have been specified.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.RepStatusApiServlet?action=getContent&se=<SE_ID>&deliveryService=<deliveryService_ID>[&criteria=<criteria>][&refetch=<true | false>]

getStatusOfContentItems

Lists content items of a delivery service, with or without search criteria, in all the Service Engines assigned to that delivery service.

Parameter

- Delivery service ID (required)
- Search criteria (optional)

  One or more content names or patterns must each be separated by a comma. Patterns can contain the wildcards * or ?.
- Refetch (optional)—The default is false.

  If refetch is set to true, a background request to retrieve the content is issued. The updated information is cached in the CDSM and can be retrieved in the next call.

Note When refetch is set to true, the request is sent to the Service Engines assigned to the delivery service to obtain new information. This is a processor-intensive operation.

Return

A list of all content items in the delivery service and their status across Service Engines, or a list of content items that matches the specified criteria and their status across Service Engines, if search criteria have been specified.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.RepStatusApiServlet?action=getStatusOfContentItems&deliveryService=<deliveryService_ID>[&criteria=<criteria>][&refetch=<true | false>]

getStatusOfContentItemInDeliveryService

Lists the status of a specified content item in the delivery service on all the Service Engines assigned to the delivery service.

Parameter

- Delivery service ID (required)
- Complete URL of the content item (required)

Return

The status of the specified content item on all the Service Engines assigned to the delivery service.

Note This action must be called after the getStatusOfContentItems action; otherwise, unexpected output results. The URL must be one of the URLs listed in the output of the getStatusOfContentItems action.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.RepStatusApiServlet?action=getStatusOfContentItemInDeliveryService&deliveryService=<deliveryService_ID>&criteria=<complete URL of the delivery service content item>

XML-Formatted Output for Replication Status

The following is the Document Type Definition (DTD) of the XML-formatted output for the replication status:

```
<?xml version="1.0"?>
<!DOCTYPE replicationStatus[
  <!ELEMENT replicationStatus (message, CeStatus*, Delivery-Service-Status*)>
  <!ATTLIST replicationStatus
        action         CDATA #REQUIRED
        count          CDATA #REQUIRED
  >
  <!ELEMENT message EMPTY>
  <!ATTLIST message
    status         (success | failure) "success"
        message        CDATA #REQUIRED
  >
  <!ELEMENT CeStatus EMPTY>
  <!ATTLIST CeStatus
```

```
      ceId           CDATA #REQUIRED
      ceName         CDATA #IMPLIED
      channelId      CDATA #REQUIRED
      channelName    CDATA #IMPLIED
      state          CDATA #IMPLIED
      filesDone      CDATA #IMPLIED
      filesToDo      CDATA #IMPLIED
      filesFailed    CDATA #IMPLIED
      filesUpdateFailed  CDATA #IMPLIED
      totalFiles         CDATA #IMPLIED
      updateTime         CDATA #IMPLIED
 >
<!ELEMENT Delivery servicestatus EMPTY>
<!ATTLIST Delivery servicestatus
      id                 CDATA #REQUIRED
      totalNumCes        CDATA #REQUIRED
      numCesComplete     CDATA #REQUIRED
      numCesInProcess    CDATA #REQUIRED
      numCesFailed       CDATA #REQUIRED
      numCesUnknownState CDATA #REQUIRED
      rootCeState        CDATA #REQUIRED
      manifestError      CDATA #IMPLIED
      usedDiskQuota      CDATA #IMPLIED
      validAsOf          CDATA #IMPLIED
>
```

CHAPTER 3

Provisioning APIs

This chapter describes the Delivery Service Provisioning APIs, Location Provisioning APIs, Service Engine Provisioning APIs, Program APIs, and URL Management APIs and the servlet actions they perform. This chapter contains the following sections:

Delivery Service Provisioning API Actions

The Delivery Service Provisioning API is the ChannelApiServlets.

Some of the output fields are not used for the following actions:

- createDeliveryService
- modifyDeliveryService
- createContentOrigin
- modifyContentOrigin

Table 3-1 lists the unused output fields.

Table 3-1 Output Fields Not Used in the CDS

Schema Object	Unused Field	Comment
CeConfig	TftpDirectoryListingId	"CeConfig" is mapped to the"Service Engine" schema object. TFTP and WCCP are not used. Although "TftpDirectoryListingId," "TftpProxyList," and "WccpRouterListsPerCeForDg" can be queried by API, they are not used in the CDS.
	WccpConfig	
	TftpProxyList: <list name="TftpProxyList" type="TftpProxy" size="0"/>	
	WccpRouterListsPerCeForDg : <list name="WccpRouterListsPerCeForDg" type="WccpRouterListPerCeForDg" size="0" />	

Table 3-1 Output Fields Not Used in the CDS (continued)

Schema Object	Unused Field	Comment
Website	ContentProvidId	"Website" is mapped to the "content origin" schema object. Content Provider and CIFS configurations are not used. Although "ContentProvidId" and "CifsWebsites" can be queried by API, they are not used in the CDS.
	CifsWebsites: <list name="CifsWebsites" type="CifsWebsites" size="0" />	
Channel	MCastEnabled	"Channel" is mapped to the "delivery service"schema object. Content Provider and multicast configurations are not used. Although "MCastEnabled," and "ChannelMCasts" can be queried by API, they are not used in the CDS.
	ChannelMCasts: <list name="ChannelMCasts" type="ChannelMCast" size="0" />	

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.ChannelApiServlet…

This servlet performs one or more of the following actions:

- createDeliveryService
- addManifest
- assignSEs
- assignDeliveryServiceIp
- fetchNow
- modifyDeliveryService
- modifyManifest
- unassignSEs
- unassignDeliveryServiceIp
- deleteDeliveryServices
- createContentOrigin
- modifyContentOrigin
- deleteContentOrigin
- applyRuleFile

createDeliveryService

Creates a delivery service.

Parameter

- Delivery service name (required)
- Content origin ID associated with the specified delivery service (required)
- Weak certification (optional)—The default is false.
- Skip encryption (optional)—The default is false.
- Delivery service priority (optional)—The default is medium.

  The settings are:
  - High
  - Medium
  - Low
- Delivery service description (optional)—The default is null.
- FailoverIntvl (optional)—The default is 120.
- Never (optional)—The default is false.
- Live (optional)—The default is false.

Return

The newly created delivery service ID.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.ChannelApiServlet?action=createDeliveryService&deliveryService=<deliveryService_name>&contentOrigin=<contentorigin_ID>[&weakCert=<true | false>][&skipEncrypt= <true | false>][&priority=<high | medium | low>][&failoverIntvl=<20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 | 110 | 120>][&never=<true | false>][&qos=<system|0-63>][&desc=<description>][&live=<true | false>]

addManifest

Adds a Manifest file to a specified delivery service.

Parameter

- Delivery service ID (required)
- Manifest URL (required)
- Quota (required)
- TTL (required)—In minutes
- User ID (optional)
- User password (optional)
- User domain name (optional)
- Not basic authentication (optional)—The default is false.

- No proxy (optional)—The default is false.
- Proxy IP address or host name (optional)
- Proxy port (optional)
- Proxy username (optional)
- Proxy password (optional)
- Proxy NTLM user domain name (optional)
- Proxy not basic authentication (optional)—The default is false.

Return

The updated delivery service record.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.ChannelApiServlet?action=addManifest&deliveryService=<deliveryService_ID>&manifest=<manifest_URL>&quota=<quota>&ttl=<ttl>[&user=<user_name>][&password=<password>][&userDomainName=<user_domain_name>][&notBasicAuth=<truelfalse>][&noProxy=<true | false>][&proxyIpHostname=<proxy_ip_hostname>][&proxyPort=<proxy_port>][&proxyUser=<proxy_user>][&proxyPassword=<proxy_password>][&proxyNtlmUserDomainName=<proxy_ntlm_user_domain_name>][&proxyNotBasicAuth=<truelfalse>]

assignSEs

Assigns Service Engines to a specified delivery service.

This action need not be used if the assignDeliveryService action has already been used. If a delivery service has already been assigned to a program, the assignSEs action executes successfully but returns a warning message.

Parameter

- Delivery service ID (required)
- Content Acquirer ID (required if no Content Acquirer is assigned; otherwise, this parameter is optional)
- Either a list of Service Engines or the keyword **all** is required (see the following rules).
- Either a list of clusters or the keyword **all** is required (see the following rules).

Rules

- If a Service Engine list is set to **all**, a cluster list cannot be specified.
- If the cluster list is set to **all**, a Service Engine list cannot be specified.
- Both a Service Engine list and a cluster list cannot be set to **all** at the same time.

If these rules are violated, an error message is returned.

Return

None.

Note The Service Engine and cluster form a one-to-one relationship. A cluster is considered a wrapper around the Service Engine.

When assigning the Service Engine, specify one of the following options:

- List of Service Engines
- All Service Engines
- List of clusters
- All clusters
- List of Service Engines and clusters

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.ChannelApiServlet?action=assignSEs&deliveryService=<deliveryService_ID>[&contentAcquirer=<contentAcquirer_ID>][&se=all | <SE_ID>, <SE_ID>, …][&cluster=all | <Cluster_ID>, <Cluster_ID>, …]

assignDeliveryServiceIp

Assigns an IP address of a Service Engine to a single delivery service, a group of delivery services, or all delivery services to which the Service Engine belongs.

This action allows a delivery service to stream from an IP address configured on an interface of a Service Engine, while another delivery service streams from another IP address configured on the same interface of the Service Engine.

Parameter

- List of delivery service IDs or keyword "**all**" (required)
- IP address (required)
- Service Engine ID (required)

Rules

- IP address can be assigned to multiple delivery services, as long as the delivery services share the same content origin.
- IP address must be configured on an interface of the specified Service Engine.
- Service Engine must belong to the delivery services specified.

If these rules are violated, an error message is returned.

Return

None.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.ChannelApiServlet?action=assignDeliveryServiceIp&deliveryService=<all|deliveryService_ID,...>&ip=<IP Address>&se= <se_ID>

fetchNow

Immediately fetches the Manifest file.

Generally, the TTL (time-to-live) value of the Manifest is set to a reasonable value, such as 30 minutes. This servlet forces a freshness check of the Manifest file before the normal time-to-live interval expires on the delivery service specified. If the freshness check indicates that changes to the Manifest file have occurred, the Manifest file is parsed and the content processed. If you want the changes to the Manifest file to be processed immediately, use the fetchNow action.

Parameter

Delivery service ID (required)

Return

None.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.ChannelApiServlet?action=fetchNow&deliveryService=<deliveryService_ID>

modifyDeliveryService

Modifies delivery service settings.

Parameter

- Delivery service ID (required)
- Name of the delivery service (optional)
- Weak certification (optional)
- Skip encryption (optional)
- Delivery service priority (optional)—The default is medium.

  The settings are:
  - High
  - Medium
  - Low
- FailoverIntvl (optional)
- Never (optional)
- Description (optional)

Note If a parameter is not specified, no change is made to the original delivery service settings.

Return

The updated delivery service record.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.ChannelApiServlet?action=modifyDeliveryService&deliveryService=<deliveryService_ID>[&deliveryServiceName=<deliveryService_name>][&weakCert=<true | false>][&skipEncrypt=<true | false>][&priority=<high | medium | low>][&failoverIntvl=<20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 | 110 | 120>][&never=<true | false>][&desc=<description>]

modifyManifest

Modifies Manifest file settings.

Parameter

- Delivery service ID (required)
- Manifest URL (optional)
- Quota (optional)
- TTL (optional)
- User ID (optional)
- User password (optional)
- NTLM user domain name (optional)
- Not basic authentication (optional)—The default is false.
- No proxy (optional)—The default is false.
- Proxy IP address or host name (optional)
- Proxy port (optional)
- Proxy username (optional)
- Proxy password (optional)
- Proxy NTLM user domain name (optional)
- Proxy not basic authentication (optional)—The default is false.

Note If a parameter value is not specified, no change is made to the original Manifest file setting. If the parameter values need to be removed, use the "empty string" mechanism to delete an existing setting. For example, if a manifest was originally set for a delivery service and you now want to remove that manifest from the delivery service, set the manifest parameter to an empty string (manifest="") when using the modifyManifest action.

Setting a Manifest URL to null removes all the other settings.

Return

The updated delivery service record.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.ChannelApiServlet?action=modifyManifest&deliveryService=<deliveryService_ID>[&manifest=<manifest_URL>][&quota=<quota>][&ttl=<ttl>][&user=<user_name>][&password=<password>][&userDomainName=<user_domain_name>][&notBasicAuth=<true | false>][&noProxy=<true | false>]

[&proxyIpHostname=<proxy_ip_hostname>][&proxyPort=<proxy_port>][&proxyUser=<proxy_user>][&proxyPassword=<proxy_password>][&proxyNtlmUserDomainName=<proxy_ntlm_user_domain_name>][&proxyNotBasicAuth=<true | false>]

unassignSEs

Removes Service Engines from a specified delivery service.

This action need not be used if the unassignDeliveryService action has already been used. If a delivery service has already been assigned to a program, the unassignSEs action executes successfully but returns a warning message.

Parameter

- Delivery service ID (required)
- Either a list of Service Engines or the keyword **all** is required (see the following rules).
- Either a list of clusters or the keyword **all** is required (see the following rules).

Rules

- If a Service Engine list is set to **all**, a cluster list cannot be specified.
- If a cluster list is set to **all**, a Service Engine list cannot be specified.
- Both a Service Engine list and a cluster list cannot be set to **all** at the same time.

If these rules are violated, an error message is returned.

Return

None.

Note The Service Engine and cluster form a one-to-one relationship. A cluster is considered a wrapper around the Service Engine.

When removing the Service Engine from the delivery service, specify one of the following options:

- List of Service Engines
- All Service Engines
- List of clusters
- All clusters
- List of Service Engines and clusters

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.ChannelApiServlet?action=unassignSEs&deliveryService=<deliveryService_ID>[&se=all | <SE_ID>, <SE_ID>, …][&cluster=all | <Cluster_ID>, <Cluster_ID>, …]

unassignDeliveryServiceIp

Unassigns IP addresses of a Service Engine from a single delivery service or a group of delivery services. When an IP address of a Service Engine is unassigned from delivery services, any delivery service streaming on the IP address is interrupted.

Parameter

- List of delivery service IDs (required)
- Service Engine ID (required)

Rules

- All delivery services specified must share the same content origin.
- The Service Engine must belong to the delivery services specified.

If these rules are violated, an error message is returned.

Return

None.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.ChannelApiServlet?action=unassignDeliveryServiceIp&deliveryService=<deliveryService_ID,...>&se= <se_ID>

deleteDeliveryServices

Deletes delivery services.

Parameter

Either a list of delivery services or the keyword **all** is required.

Return

None.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.ChannelApiServlet?action=deleteDeliveryServices&deliveryService=all | <deliveryService_ID>, <deliveryService_ID>, …

createContentOrigin

Creates a content origin.

Parameter

- Content origin name (required)
- Origin server (required)
- Fully qualified domain name (FQDN) (required)

Note This is the FQDN used by the Service Router to route the requests to a Service Engine. For example, while processing a request for http://www.cnn.com (origin server FQDN), the Service Router may route the request to a Service Engine using the FQDN http://cdn.cnn.com.

- Enable content-based routing (optional)—The default is true.
- Network Attached Storage (NAS) file ID—The format is FileInfo_*xxx,* where *xxx* is the file ID. The other option is to enter "none," for example, [&nasFile=none].

Note NAS is only supported in lab integrations as proof of concept.

- WMT authentication (optional)—The default is none.
  - None
  - Basic
  - NTLM
  - Digest
  - Negotiate
- Description (optional)

Return

A confirmation that the new content origin has been created and the newly created content origin object has been saved.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.ChannelApiServlet?action=createContentOrigin&name=<contentorigin_name>&origin=<origin_server_IP_or_domain>&fqdn=<fqdn>[&contentBasedRouting=<true | false>][&nasFile=<FileInfo_id | none>][&wmtAuth=<basic | ntlm | digest | negotiate>][&description=<description>]

modifyContentOrigin

Modifies content origin settings.

Parameter

- Content origin ID (required)
- Content origin name (optional)
- Origin server (optional)
- FQDN (optional)
- Enable content-based routing (optional)—The default is true.
- NAS file ID. The format is FileInfo_*xxx,* where *xxx* is the file ID. The other option is to enter "none," for example, [&nasFile=none]

Note NAS is only supported in lab integrations as proof of concept.

- WMT authentication (optional)
  - None
  - Basic
  - NTLM
  - Digest
  - Negotiate
- Description (optional)

Return

A confirmation that content origin attributes have been modified and an updated record for the content origin object.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.ChannelApiServlet?action=modifyContentOrigin&contentOrigin=<contentorigin_ID>[&name=<contentorigin_name>][&origin=<origin=<origin_server_IP_or_domain>][&fqdn=<fqdn>][&contentBasedRouting=<true|false>] [&nasFile=<<FileInfo_id | none>][&wmtAuth=<none | basic | ntlm | digest | negotiate>][&description=<description>]

deleteContentOrigin

Deletes content origins.

Parameter

Either a list of content origin IDs or the keyword **all** is required.

Return

A confirmation that the content origins have been deleted.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.ChannelApiServlet?action=deleteContentOrigins&contentOrigin=all | <contentorigin_ID>, <contentorigin_ID>, …

applyRuleFile

Assigns a Service Rule file to a delivery service or unassigns a Service Rule file from a delivery service.

Parameter

- Delivery service ID (required)—The format is Channel_*xxx*, where *xxx* is the ID of the delivery service.
- Rule file ID (required). Valid values are:
  - None—Unassigns the Rule file from the delivery service.
  - File ID—The format is FileInfo_*xxx,* where *xxx* is the file ID.

Return

Confirmation that the Service Rule file has been assigned to the delivery service or unassigned from the delivery service.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.ChannelApiServlet?action=applyRuleFile&deliveryService=<Channel_xxx>&ruleFile=<none | FileInfo_ID>

Note The applyRuleFile API expects the deliveryService parameter to be in the form Channel_*xxx*, where *xxx* is the ID of the delivery service.

Location Provisioning API Actions

The Location Provisioning API is the LocationApiServlet.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.LocationApiServlet...

This servlet performs one or more of the following actions:

- createLocation
- modifyLocation
- deleteLocation

createLocation

Creates a specified location.

Parameter

- Location name (required)
- Parent location ID (optional)
- Description (optional)

Return

The newly created location object.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.LocationApiServlet?action=createLocation&location=<location_name>[&parent=<parent_ID>][&desc=<description>]

modifyLocation

Modifies a specified location.

Parameter

- Location ID (required)
- Location name (optional)
- Parent location ID (optional)
- Description (optional)

Return

The modified location object.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.LocationApiServlet?action=modifyLocation&location=<location_ID>[&name=<location_name>][&parent=<parent_ID>][&desc=<description>]

deleteLocation

Deletes a specified location.

Parameter

Location ID (required)

Return

A message that the specified location has been deleted.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.LocationApiServlet?action=deleteLocation&location=<location_ID>

Service Engine Provisioning API Actions

The Service Engine Provisioning API is the CeApiServlet.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.CeApiServlet...

This servlet performs one or more of the following actions:

- activateSe
- changeSeLocation
- deleteSe

activateSe

Activates a specified Service Engine.

Parameter

- Service Engine ID (required)
- Location ID (required)

Return

The modified Service Engine object.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.CeApiServlet?action=activateSe&se=<SE_ID>&location=<location_ID>

changeSeLocation

Changes the location of a specified Service Engine.

Parameter

- Service Engine ID (required)
- Location ID (required)

Return

The modified Service Engine object.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.CeApiServlet?action=changeSeLocation&se=<SE_ID>&location=<location_ID>

deleteSe

Deletes a specified Service Engine.

Parameter

Service Engine ID (required)

Return

A message that the specified Service Engine has been deleted.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.CeApiServlet?action=deleteSe&se=<SE_ID>

Program API Actions

The Program API is the ProgramApiServlet.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.ProgramApiServlet…

Note You must have administrator-level access privileges to execute Program API actions.

This servlet performs one or more of the following actions:

- createProgram
- validateProgramFile
- assignDeliveryService
- assignSEs
- fetchNow
- modifyProgramFile
- unassignDeliveryService
- unassignSEs
- deletePrograms

createProgram

Fetches a program file using HTTP, validates it, and creates a program based on the input. This action also reserves a multicast address, if the program requires one. The multicast address reserved for the program is not released until the program is deleted.

Parameter

- Program file URL (required)
- Update interval (required)—Interval (in minutes) at which to access the program file to check for updates
- User ID (optional)
- User password (optional)

Return

The newly created program record with the program ID. If the program file fails validation, an error message is returned.

Appendix A, "Program Files in the CDS Software," provides a DTD for the information that is returned.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.ProgramApiServlet?action=createProgram&file=<program_file_URL>&updateInterval=<update_interval_minutes>[&user=<user_name>][&password=<password>]

validateProgramFile

Fetches a program file using HTTP and validates it.

Parameter

Program file URL (required)

Return

None, if there are no errors. If there are errors, returns a list of errors.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.ProgramApiServlet?action=validateProgramFile&file=<program_file_URL>

assignDeliveryService

Assigns a delivery service to a program. When you assign a delivery service to a program, all Service Engines associated with the delivery service are associated with the program. Any modification to the Service Engine delivery service assignment also updates the program.

This action should not be used if the assignSEs action has already been used. If a Service Engine has already been assigned to a program, the assignDeliveryService action fails and returns the following error message:

```
<?xml version="1.0" ?>
- <programApi action="assignDeliveryService">
  <message status="fail" message="Constraint Error: Can not associate a delivery service with the playlist. Service Engines are already assigned to the playlist." />
  <error code="3" message="Constraint Error: Can not associate a delivery service with the playlist. Service Engines are already assigned to the playlist." />
  </programApi>
```

Parameter

- Program ID (required)
- Delivery service ID (required)

Return

The updated program record.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.ProgramApiServlet?action=assignDeliveryService&program=<program_ID>&deliveryService=<deliveryService_ID>

assignSEs

Assigns Service Engines to a program.

This action should not be used if the assignDeliveryService action has already been used. If a delivery service has already been assigned to a program, the assignSEs action fails and returns the following error message:

```
<?xml version="1.0" ?>
- <programApi action="assignSEs">
  <message status="fail" message="Constraint Error: Can not assign Service Engines to the
playlist. The playlist is already associated with a delivery service." />
  <error code="3" message="Constraint Error: Can not assign Service Engines to the
playlist. The playlist is already associated with a delivery service." />
  </programApi>
```

Note This action fails if the program represents a live event, because live programs must be assigned to a live delivery service.

Parameter

- Program ID (required)
- Either a list of Service Engines or the keyword **all** is required.

Return

The updated program record.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.ProgramApiServlet?action=assignSEs&program=<program_ID>&se=all | <SE_ID>, <SE_ID>, ...

fetchNow

Fetches a program file immediately using HTTP and updates the program.

Parameter

Program ID (required)

Return

None, if there are no errors. Displays an error message if the program file fails validation.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.ProgramApiServlet?action=fetchNow&program=<program_ID>

modifyProgramFile

Modifies program file settings.

Parameter

- Program ID (required)
- Program file URL (optional)
- Update interval (optional)
- User ID (optional)
- User password (optional)

Note If a parameter value is not specified, no change is made to the original program file setting. If the parameter values need to be removed, use the "empty string" mechanism to delete an existing setting. For example, if you now want to remove the user ID from the program file, set the user ID parameter to an empty string (user="") when using the modifyProgramFile action.

Note You cannot set the program file URL to an empty string. Setting the program file URL to null removes all the other settings.

Return

The updated program record.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.ProgramApiServlet?action=modifyProgramFile&program=<program_ID>[&file=<program_file_URL>][&updateInterval=<update_interval>][&user=<user_name>][&password=<password>]

unassignDeliveryService

Removes a delivery service from the specified program.

This action should not be used if the unassignSEs action has already been used. The unassignDeliveryService action executes successfully even if a Service Engine has already been unassigned from a program, but displays a warning that the delivery service is not assigned to the program.

Parameter

- Program ID (required)
- Delivery service ID (required)

Return

The updated program record.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.ProgramApiServlet?action=unassignDeliveryService&program=<program_ID>&deliveryService=<deliveryService_ID>

unassignSEs

Removes Service Engines from the specified program.

This action need not be used if the unassignDeliveryService action has already been used. The unassignSEs action executes successfully even if a delivery service has been already unassigned from a program, but displays a warning that the Service Engines are not assigned to the program.

Parameter

- Program ID (required)
- Either a list of Service Engines or the keyword **all** is required.

Return

The updated program record.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.ProgramApiServlet?action=unassignSEs&program=<program_ID>&se=all | <SE_ID>, <SE_ID>, …

deletePrograms

Deletes programs.

Parameter

A list of programs by service type (such as WMT or Movie Streamer) or program ID, or the keyword **all** is required.

Return

None.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.ProgramApiServlet?action=deletePrograms&program=all | id=<program_ID>, <program_ID>, ... | type=<wmt | MovieStreamer>

URL Management API Actions

The URL Management API is the UrlManagementApiServlet.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.UrlManagementApiServlet...

This servlet performs one or more of the following actions:

- singleURLRemoval

- batchURLRemoval

singleURLRemoval

Removes content items from delivery service based on a specified URL. The details for each content removal request are displayed.

Parameter

Single URL (required)

Return

200 Ok—Content URL removal is successful on all Service Engines.

500 Failed to communicate with SE at IP: <*SE IP addr*>—Please ensure the SE is online and the Centralized Management System (CMS) processes are running. The CLI **show cms processes** command can be used for viewing the status of the CMS processes and **cms enable** for enabling the CMS.

500 Failed to remove the content from the SE at IP: <*SE IP addr*> | 200 Ok—Content URL(s) removal is successful on the Service Engines with the following IPs: <*SE IP addr1, SE IP addr2, ...*>

Syntax

https://<cdsmIpAdress>:8443/servlet/com.cisco.unicorn.ui.UrlManagementApiServlet?action=singleURLRemoval&singleUrl=<url>

batchURLRemoval

Removes content items from the delivery service based on a specified set of URLs. The details for each content removal request are displayed.

Parameter

Batch URL (required)

Return

None.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.UrlManagementApiServlet?action=batchURLRemoval<Only programmed API call allowed>

Note The batchURLRemoval requires a programmed API call; it does not work as an interactive API call.

Following is an example of Java code that can be used to call the batchURLRemoval API. Java Development Kit (JDK) 1.6 or higher is required to compile and use this Java code example.

```
import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.MalformedURLException;
```

```
import java.net.URL;
import javax.net.ssl.HostnameVerifier;
import javax.net.ssl.HttpsURLConnection;
import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLSession;
import javax.net.ssl.TrustManager;
import javax.net.ssl.X509TrustManager;

public class BatchURLDemo {

    public static class newHostNameVerifier implements HostnameVerifier {
        /**
         * ignore hostname checking
         */
        public boolean verify(String hostname, SSLSession session) {
            return true;
        }
    }

    public static void main(String args[]) {
        try {

            String userName_ = "admin";  /* CDSM user name*/
            String password_ = "default";/* CDSM password name*/
            String cdsmAddress_ = "10.77.153.98";/* CDSM IP address OR hostname */
            String apiServlet = "UrlManagementApiServlet";  /* API servlet name to call */
            String action = "batchURLRemoval";/* API action name to call */
            String urlsFile = "C:\\batchremoval.xml";/* The path for URLs XML file */
            int cdsmPort_ = 8443;/* CDSM https port number */

            /**
             * Create a trust manager that does not validate certificate chains
             */
            TrustManager[] trustAllCerts = new TrustManager[] { new X509TrustManager() {
                public java.security.cert.X509Certificate[] getAcceptedIssuers() {
                    return null;
                }

                public void checkClientTrusted(
                        java.security.cert.X509Certificate[] certs,
                        String authType) {
                    /**
                     * do any special handling here, or re-throw exception.
                     */
                }

                public void checkServerTrusted(
                        java.security.cert.X509Certificate[] certs,
                        String authType) {
                    /**
                     * Possibly pop up a dialog box asking whether to trust the cert chain
                     */
                }
            } };

            /**
             * Install the all-trusting trust manager
             */
            SSLContext sc = SSLContext.getInstance("SSL");
            sc.init(null, trustAllCerts, new java.security.SecureRandom());
            HttpsURLConnection.setDefaultSSLSocketFactory(sc.getSocketFactory());

            String sAuth = userName_+":"+password_;
            String sEncodedAuth = new sun.misc.BASE64Encoder().encode(sAuth.getBytes());
```

```
                URL url = new URL(null,
"https://"+cdsmAddress_+":"+cdsmPort_+"/servlet/com.cisco.unicorn.ui." +
                    apiServlet +"?action="+action);

            HttpsURLConnection conn = null;
            DataOutputStream dos = null;


            String lineEnd = "\r\n";
            String hyphenLiteral = "--";
            String mPartBoundary = "*****";

            int maxBufferSize = 1024 * 1024;
            int bytesRead, bytesAvailable, bufferSize;
            byte[] buffer;

            try {
                /**
                 * initialize the HTTPS connection with post method
                 */
                FileInputStream fileInputStream = new FileInputStream(new File(
                        urlsFile));
                conn = (HttpsURLConnection) url.openConnection();
                conn.setRequestProperty("Authorization", "Basic "
                        + sEncodedAuth);
                conn.setHostnameVerifier(new newHostNameVerifier());
                conn.setDoInput(true);
                conn.setDoOutput(true);
                conn.setUseCaches(false);
                conn.setRequestMethod("POST");
                conn.setRequestProperty("Connection", "Keep-Alive");
                conn.setRequestProperty("Content-Type",
                        "multipart/form-data;boundary=" + mPartBoundary);
                dos = new DataOutputStream(conn.getOutputStream());
                dos.writeBytes(hyphenLiteral + mPartBoundary + lineEnd);
                dos
                        .writeBytes("Content-Disposition: form-data; name=\"upload\";"
                                + " filename=\""
                                + urlsFile
                                + "\""
                                + lineEnd);
                dos.writeBytes(lineEnd);

                /**
                 * load the URL xml file and upload it to server
                 */
                bytesAvailable = fileInputStream.available();
                bufferSize = Math.min(bytesAvailable, maxBufferSize);
                buffer = new byte[bufferSize];

                bytesRead = fileInputStream.read(buffer, 0, bufferSize); // write

                while (bytesRead > 0) {
                    dos.write(buffer, 0, bufferSize);
                    bytesAvailable = fileInputStream.available();
                    bufferSize = bytesAvailable;
                    bytesRead = fileInputStream.read(buffer, 0, bufferSize);
                }

                dos.writeBytes(lineEnd);
                dos.writeBytes(hyphenLiteral + mPartBoundary + hyphenLiteral
                        + lineEnd);
```

```
                    fileInputStream.close();
                    dos.flush();
                    dos.close();
                     catch (MalformedURLException ex) {
                    System.out.println("Printing Exception Message " + ex);
                     catch (IOException ioexception) {
                    System.out.println("Printing Exception Message " + ioexception);
                }

                /**
                 * Handling the response from CDSM
                 */
                try {
                    BufferedReader inStreamReader = new BufferedReader(
                            new InputStreamReader(conn.getInputStream()));
                        String str;
                        while ((str = inStreamReader.readLine()) != null) {
                        System.out.println("Response from CDSM : ");
                        System.out.println(str);
                    }
                    inStreamReader.close();
                     catch (IOException ioexception) {
                    System.out.println("Printing Exception Message " + ioexception);
                }

                 catch (Exception e) {
                System.out.println("Printing Exception Message " + e);
                e.printStackTrace();
            }

        }
}
```

If the above Java code was saved in a file called “BatchRULDemo.java,” then to compile the code you would enter the **javac BatchURLDemo.java** command, and to run the script you would enter the **java BatchURLDemo** command.

```
javac BatchURLDemo.java
java BatchURLDemo
```

The following is an example of the XML file that is used in the Java code:

```
<?xml version="1.0" encoding="UTF-8"?>
<URLRemovalList xmlns='http://cisco.com/unicorn/cds/urlmgmt'
xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'>
<url-entry>http://2.2.23.32/Thursday.html</url-entry>
<url-entry>http://2.2.23.32/Hello.html</url-entry>
</URLRemovalList>
```

The following is an example of the output returned for the above Java code:

```
<?xml version="1.0"?><URLManagement action="batchURLRemoval"><message status="success"
message="200 OK - Content URL(s) removal is successful on all streaming
engines."/></URLManagement>
```

The details for each content removal request is displayed.

CHAPTER 4

Listing APIs

This chapter describes the Listing APIs and the servlet actions they perform.

Listing API Actions

The Listing API is the ListingApiServlet. If there is a list inside the object, the listed items are printed as elements of the object.

Some of the output fields are not used for the following actions:

- getSEs
- getDeliveryServices
- getContentOrigins

Table 4-1 lists the unused output fields.

Table 4-1 Output Fields Not Used in the CDS

Schema Object	Unused Field	Comment
CeConfig	TftpDirectoryListingId	"CeConfig" is mapped to the "Service Engine" schema object. TFTP and WCCP are not used. Although "TftpDirectoryListingId," "TftpProxyList," and "WccpRouterListsPerCeForDg" can be queried by API, they are not used in the CDS.
	WccpConfig	
	TftpProxyList: <list name="TftpProxyList" type="TftpProxy" size="0"/>	
	WccpRouterListsPerCeForDg : <list name="WccpRouterListsPerCeForDg" type="WccpRouterListPerCeForDg" size="0" />	
Website	ContentProvidId	"Website" is mapped to the "content origin" schema object. Content Provider and CIFS configurations are not used. Although "ContentProvidId" and "CifsWebsites" can be queried by API, they are not used in the CDS.
	CifsWebsites: <list name="ChannelMCasts" type="ChannelMCast" size="0" />	

Table 4-1 Output Fields Not Used in the CDS (continued)

Schema Object	Unused Field	Comment
Channel	MCastEnabled	"Channel" is mapped to the "delivery service"schema object. Content Provider and multicast configurations are not used. Although "MCastEnabled," and "ChannelMCasts" can be queried by API, they are not used in the CDS.
	ChannelMCasts: <list name="ChannelMCasts" type="ChannelMCast" size="0" />	

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.ListApiServlet…

This servlet performs one or more of the following actions:

- getContentOrigins
- getDeliveryServices
- getSEs
- getClusters
- getLocations
- getDeviceGroups
- getDeviceStatus
- getObjectById
- getObjectByName
- getPrograms
- getPgmMcastAddrInUse
- getMcastAddrInUse

getContentOrigins

Lists selected content origin names or lists every content origin.

Parameter

Either a list of content origin names or the keyword **all** is required.

Return

A list of all content origins specified and their details.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.ListApiServlet?action=getContentOrigins&param=all | <contentOrigin_name>, <contentOrigin_name>, …

getDeliveryServices

Lists selected delivery service names and related content origin ID or lists all delivery services.

Parameter

A list of delivery service names with related content origin IDs, a Service Engine ID, a program ID, or the keyword **all** is required.

Return

A list of all delivery services specified and their details.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.ListApiServlet?action= getDeliveryServices&param=all | [name=]<contentOrigin_ID>:(all | <deliveryService_name>) ... | se=<seConfig_ID> | program=<playlist_ID>

getSEs

Lists selected Service Engines by Service Engine name, delivery service, or location, or lists all Service Engines. When Service Engines are listed by location, all Service Engines in the given location and all Service Engines (child, grandchild, and so forth) in the subordinate locations are listed.

Parameter

A list of Service Engine names, a delivery service ID, a location ID, or the keyword **all** is required.

Return

A list of all Service Engines specified and their details.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.ListApiServlet?action=getSEs&param= all | [name=]<se_name>, <se_name>, ... | deliveryService=<deliveryService_ID> | location=<location_ID>

getClusters

Lists selected cluster names or lists every cluster.

Parameter

Either a list of cluster IDs or the keyword **all** is required.

Return

A list of all clusters specified and their details.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.ListApiServlet?action= getClusters& param=all | <Cluster_ID>, <Cluster_ID>, …

getLocations

Lists the location of the specified Service Engines or the locations of all Service Engines.

Parameter

Either the Service Engine ID or the keyword **all** is required.

Return

The requested location record.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.ListApiServlet?action=getLocations&param=all | bySeId=<SE_ID>

getDeviceGroups

Lists selected device group names or lists all device groups.

Parameter

Either a list of device group names or the keyword **all** is required.

Return

A list of all device groups specified and their details.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.ListApiServlet?action=getDeviceGroups&param=all | <device_group_name>, <device_group_name>, …

getDeviceStatus

Lists the status of a device by name.

Parameter

Name of the device that contains the ID of the device or device group.

Note The name of the device is case sensitive.

Return

A list of devices and their status.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.DeviceApiServlet?action=getDeviceStatus[&name=<device_ID> | <deviceGroup_ID>]

getObjectById

Lists an object, based on its string ID.

Parameter

Object string ID

The following are the object types:

- Service Engine
- Delivery service
- Cluster
- Device group
- Content origin
- Program

Return

The requested object.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.ListApiServlet?action= getObjectById&param=<SE_ID | DeliveryService_ID | Cluster_ID | DeviceGroup_ID | ContentOrigin_ID | Playlist_ID>

Note This API is restricted based on permissions granted to the specified user requesting the API. The CDSM allows assignment of API access rights to any user. A user with administrator's privileges bypasses the authentication. For other users, this API is accessed by granting particular rights in the CDSM AAA system.

getObjectByName

Lists an object, based on its name.

Parameter

Object type:object name

The following are the object types:

- Service Engine
- Delivery service
- Device group
- Content origin
- Program

The delivery service name format is content origin name:delivery service name.

Return

The requested object.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.ListApiServlet?action=getObjectByName&param=SE:<seName>|DeliveryService:<deliveryServiceName>|DG:<dgName>|ContentOrigin:<contentOriginName>|Program:<programName>

Note If the type of object is a program, you must have administrator-level access privileges to execute this action, or have user-specific access rights granted for this API.

getPrograms

Lists all programs specified or all programs and their details.

Parameter

A list of program types, program names, delivery service ID, or program ID, or the keyword **all** is required.

Return

A list of all programs specified and their details.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.ListApiServlet?action=getPrograms&param=all | type=<wmt | movieStreamer> | name=<program_name>,<program_name>, ... | deliveryService=<deliveryServiceID> | id=<playlist_ID>,<playlist_ID>...

Note You must have administrator-level access privileges to execute this action, or have user-specific access rights granted for this API.

getPgmMcastAddrInUse

Lists all multicast addresses currently in use by programs.

Parameter

None.

Return

A list of all the multicast addresses currently in use by programs.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.ListApiServlet?action=getPgmMcastAddrInUse

Note You must have administrator-level access privileges to execute this action, or have user-specific access rights granted for this API.

getMcastAddrInUse

Lists all multicast addresses currently in use.

Parameter

None.

Return

A list of all the multicast addresses currently in use.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.ListApiServlet?action=getMcastAddrInUse

Note You must have administrator-level access privileges to execute this action, or have user-specific access rights granted for this API.

CHAPTER 5

Statistics APIs

This chapter describes the Monitoring Statistics API and Streaming Statistics API, and the servlet actions they perform. This chapter contains the following sections:

Monitoring Statistics API Actions

This section describes the Monitoring Statistics API and the servlet actions it performs. The Monitoring Statistics API gets monitoring statistics data about a single Service Engine or all the Service Engines in a CDS network. The Monitoring Statistics API is the MonitoringApiServlet.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.MonitoringApiServlet…

This servlet performs one or more of the following actions:

getSeStats

Obtains monitoring statistics information for the specified Service Engine.

Parameter

- Service Engine ID (required)
- Monitoring statistics type (required)

  The monitoring statistics types are:

  - bytes_served
  - bandwidth_efficiency_gain
  - streaming_sessions
  - cpu_utilization

- Time frame (optional)—The time period over which monitoring statistics are obtained. The default is last_hour.

  The options are:

  - last_hour
  - last_day
  - last_week
  - last_month
  - custom

  Note If you choose custom, you must specify the time frame using the End time from and End time to options.

- End time from (optional)—The date and time that collection of monitoring statistics data should start. You can specify only the date or the date and time. The date format is mm/dd/yyyy, and the time format is hh:mm. Optionally, you can specify the time in hh:mm:ss.
- End time to (optional)—The date and time that collection of monitoring statistics data should end. You can specify only the date or the date and time. The date format is mm/dd/yyyy, and the time format is hh:mm. Optionally, you can specify the time in hh:mm:ss.
- Time zone (optional)—The time zone used to generate the monitoring statistics data. The default is utc.

  The time zones are:

  - utc—Time in UTC (Coordinated Universal Time)
  - se_local_time—Time zone specified on the Service Engine
  - cdsm_local_time—Time zone specified on the CDSM

Return

Requested monitoring statistics data for the selected time period for the Service Engine.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.MonitoringApiServlet?action=getSEstats&id=<SE_id>&type=<bytes_served | bandwidth_efficiency_gain | streaming_sessions | cpu_utilization>[&time_frame=<last_hour | last_day | last_week | last_month | custom>][&end_time_from=<mm/dd/yyyy [hh:mm[:ss]]>][&end_time_to=<mm/dd/yyyy [hh:mm[:ss]]>][&time_zone=<utc | se_local_time | cdsm_local_time>]

getLocationStats

Obtains monitoring statistics information for all the Service Engines in the specified location.

Parameter

- Location ID (required)
- Monitoring statistics type (required)

The monitoring statistics types are:

- bytes_served
- bandwidth_efficiency_gain
- streaming_sessions

- Time frame (optional)—The time period over which monitoring statistics are obtained. The default is last_hour.

  The options are:

  - last_hour
  - last_day
  - last_week
  - last_month
  - custom

Note If you choose custom, you must specify the time frame using the End time from and End time to options.

- End time from (optional)—The date and time that collection of monitoring statistics data should start. You can specify only the date or the date and time. The date format is mm/dd/yyyy, and the time format is hh:mm. Optionally, you can specify the time in hh:mm:ss.
- End time to (optional)—The date and time that collection of monitoring statistics data should end. You can specify only the date or the date and time. The date format is mm/dd/yyyy, and the time format is hh:mm. Optionally, you can specify the time in hh:mm:ss.
- Time zone (optional)—The time zone used to generate the monitoring statistics data. The default is utc.

  The time zones are:

  - utc—Time in UTC (Coordinated Universal Time)
  - cdsm_local_time—Time zone specified on the CDSM

Return

Requested monitoring statistics data for the selected time period for the Service Engines in the specified location.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.MonitoringApiServlet?action=getLocationStats&id=<Location_ID>&type=<bytes_served | bandwidth_efficiency_gain | streaming_sessions>[&time_frame=<last_hour | last_day | last_week | last_month | custom>][&end_time_from=<mm/dd/yyyy [hh:mm[:ss]]>][&end_time_to=<mm/dd/yyyy [hh:mm[:ss]]>][&time_zone=<utc | cdsm_local_time>]

getCdnStats

Obtains monitoring statistics information for the entire CDS network.

Parameter

- Monitoring statistics type (required)

  The monitoring statistics types are:

  - bytes_served
  - bandwidth_efficiency_gain
  - streaming_sessions

- Time frame (optional)—The time period over which monitoring statistics are obtained. The default is last_hour.

  The options are:

  - last_hour
  - last_day
  - last_week
  - last_month
  - custom

Note If you choose custom, you must specify the time frame using the End time from and End time to options.

- End time from (optional)—The date and time that collection of monitoring statistics data should start. You can specify the date or the date and time. The date format is mm/dd/yyyy, and the time format is hh:mm. Optionally, you can specify the time in hh:mm:ss.
- End time to (optional)—The date and time that collection of monitoring statistics data should end. You can specify the date or the date and time. The date format is mm/dd/yyyy, and the time format is hh:mm. Optionally, you can specify the time in hh:mm:ss.
- Time zone (optional)—The time zone used to generate the monitoring statistics data. The default is utc.

  The time zones are:

  - utc—Time in UTC (Coordinated Universal Time)
  - cdsm_local_time—Time zone specified on the CDSM

Return

Requested monitoring statistics data for the selected time period for all the Service Engines in the CDS network.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.MonitoringApiServlet?action=getCdnStats&type=<bytes_served | bandwidth_efficiency_gain | streaming_sessions>[&time_frame=<last_hour | last_day | last_week | last_month | custom>][&end_time_from=<mm/dd/yyyy [hh:mm[:ss]]>] [&end_time_to=<mm/dd/yyyy [hh:mm[:ss]]>][&time_zone=<utc | cdsm_local_time>]

Streaming Statistics API Actions

This section describes the Streaming Statistics API and the servlet actions it performs. The streaming statistics are collected from the CDS network Service Engines and device groups and sent to the CDSM. The HTTP, Movie Streamer,and WMT streaming statistical data is monitored and displayed in the CDSM for all Service Engines, all device groups, or all the Service Engines within a selected device group.

The Streaming Statistics API is the SprayerApiServlet. The CDSM must be running for the servlet to function.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.SprayerApiServlet...

This servlet performs one or more of the following actions to collect statistics for each Service Engine (SE), device group (DG), or device group name for all the Service Engines in the specified device group:

- getHttp
- getMovieStreamer
- getWmt

getHttp

Collects HTTP streaming statistics data from the Service Engines and device groups and sends it to the CDSM.

Parameter

The SE keyword, DG keyword, or the name of the device group is required.

Return

Requested HTTP streaming statistics data for the Service Engines or device groups.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.SprayerApiServlet?action=getHttp&param=SE | DG | <DG_name>

getMovieStreamer

Collects Movie Streamer streaming statistics data from the Service Engines and device groups and sends it to the CDSM.

Parameter

The SE keyword, DG keyword, or the name of the device group is required.

Return

Requested Movie Streamer streaming statistics data for the Service Engines or device groups.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.SprayerApiServlet?action=getMovieStreamer&param=SE | DG | <DG_name>

getWmt

Collects WMT streaming statistics data from the Service Engines and device groups and sends it to the CDSM.

Parameter

The SE keyword, DG keyword, or the name of the device group is required.

Return

Requested WMT streaming statistics data for the Service Engines or device groups.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.SprayerApiServlet?action=getWmt&param=SE | DG | <DG_name>

XML-Formatted Output for Streaming Statistics

The following is the Document Type Definition (DTD) of the XML-formatted output for streaming statistics:

```
<?xml version="1.0" ?>
DOCTYPE <!DOCTYPE sprayerStats [

<!ELEMENT sprayerStats (message, (HttpStats* | MovieStreamerStats* | WmtStats* |
FmsStats*) )>
<!ATTLIST sprayerStats
      action (getHttp | getMovieStreamer | getWmt | getFms )#REQUIRED
      count CDATA #REQUIRED>

<!ELEMENT message EMPTY>
<!ATTLIST message
      status (success | fail) success
      message CDATA #IMPLIED>

<!ELEMENT HttpStats EMPTY>
<!ATTLIST HttpStats
      name CDATA #REQUIRED
      requestsPerSec CDATA #REQUIRED
      bytesPerSec CDATA #REQUIRED
      hitRate CDATA #REQUIRED>

<!ELEMENT MovieStreamerStats EMPTY>
<!ATTLIST MovieStreamerStats
      name CDATA #REQUIRED
      totalBytes CDATA #REQUIRED
      totalPackets CDATA #REQUIRED
      rtspConnections CDATA #REQUIRED
      allConnections CDATA #REQUIRED>

<!ELEMENT WmtStats EMPTY>
<!ATTLIST WmtStats
```

```
      name CDATA #REQUIRED
      requestsPerSec CDATA #REQUIRED
      bytesPerSec CDATA #REQUIRED
      hitRate CDATA #REQUIRED>

<!ELEMENT FmsStats EMPTY>
<!ATTLIST FmtStats
      name CDATA #REQUIRED
      allConnections CDATA #REQUIRED
      bytesPerSec CDATA #REQUIRED
      hitRate CDATA #REQUIRED>

]>
```

CHAPTER 6

File Management APIs

This chapter describes the File Management API and the servlet actions it performs. File Management API actions are used to manage external XML files registered with the CDSM. These external files include Coverage Zone files, Network Attached Storage (NAS) files, Service Rule files, and CDN Selector files.

Note NAS is only supported in lab integrations as proof of concept.

Coverage Zone files are registered with the CDSM and associated with a specific Service Router (SR) or applied globally to the CDN network using the File Management API.

NAS files are registered with the CDSM using the File Management API. The following Delivery Service Provisioning API actions are used to associate a NAS file with a Content Origin, which, through delivery services, makes the NAS file settings available to all root devices located in the same tier as the Content Acquirer:

- createContentOrigin, page 3-9
- modifyContentOrigin, page 3-10

Service Rules files are registered with the CDSM using the File Management API. The Delivery Service Provisioning API action, applyRuleFile, page 3-11, is used to apply a Service Rule file to devices associated with a delivery service.

CDN Selector files are registered with the CDSM and applied to an SR using the File Management API.

Using Multipart/Form-Data Request to Upload a File

There are two import methods for the FileMgmtApiServlet actions:

- Import-imports a file from an external HTTP, HTTPS, or FTP server
- Upload-uploads a file from any location that is accessible from your PC

For the “upload” import method, a multipart/form-data request is used. Following is an example of the upload import method for the registerFile action of the FileMgmtApiServlet that uses the curl utility to upload a file for the HTTPS root CA:

```
curl -k -u admin:default -F "rootfile=@rootc.crt"
https://10.74.61.199:8443/servlet/com.cisco.unicorn.ui.FileMgmtApiServlet?action=registerF
ile&importMethod=upload&fileType=26&destName=rootc.pem
```

In this example, the curl utility uploads the file and the URL sets the parameters; specifically the destination filename is rootc.pem.

If the curl utility is used, another way to upload the file is to use the option -F "file=!sourceFile.xml" can upload the original file sourceFile.xml as a multipart/form-data request.

The following actions of the FileMgmtApiServlet use a multipart/form-data request for the importMethod=upload:

- registerFile
- validateFile
- modifyFile

The refetchFile only uses importMethod=import.

File Management API Actions

The File Management API uses the FileMgmtApiServlet. Some of the output fields are not used for the following actions:

- registerFile
- modifyFile
- listFile

Table 6-1 lists the unused output fields.

Table 6-1 Output Fields Not Used in the CDS

Schema Object	Unused Field	Comment
Record	PacInfos: <list name="PacInfos" type="PacInfo" size="0"/>	Although PacInfos and DsvcLocations can be queried by API, they are not used in the CDS.
	DsvcLocations: <list name="DsvcLocations" type="DsvcLocation" size="0"/>	

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.FileMgmtApiServlet...

The servlet performs one or more of the following actions:

listTypes

List all of the file types supported by the File Management API.

Parameter

None.

Return

The list of file types supported by the File Management API.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.FileMgmtApiServlet?action=listTypes

registerFile

Registers a file with the CDSM using either the import or upload method. The import method allows you to import a supported file from an external HTTP, HTTPs, FTP, or CIFS server. The upload method allows you to upload a supported file from any location that is accessible from your PC.

Parameter

- File type (required)

  The settings are:

  - 1—Coverage Zone file
  - 19—CDN Selector file
  - 20—Rule file
  - 22—NAS file
- Destination file name (required)

- Import method (required)

  The settings are:

  - Import
  - Upload

If the import method is set to import, the following parameters also apply:

- URL of the origin file (required)—For example, //myserver/folder/myfile.txt or <*protocol*>://myhost/myfile.txt. The protocols supported are:
  - http://
  - https://
  - ftp://
- Time-to-live (TTL) (optional)—Frequency, in minutes, with which the CDSM looks for changes in the source file. The default is 10. The range is from 1 to 1440.
- NT LAN Manager (NTLM) user domain name (optional)
- Username (optional)
- Password (optional)
- Disable Basic Authentication (optional)—The default is false.

  When set to true, NTLM headers cannot be stripped off to allow fallback to the basic authentication method.

Rule

When the import method is set to upload, the source file is required when posting a multi-part form-data request.

Return

A confirmation that the file has been registered.

Note In the XML file returned, an internal reference is assigned to the file in the format of FileInfo_*xxx,* where *xxx* is the ID of the file.

Syntax

To register a file using the import method, use the following syntax:

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.FileMgmtApiServlet?action=registerFile&fileType=<1 | 19 | 20 | 22>&destName=<destination_filename>&importMethod=import&originUrl=<file_url> [&ttl=<update_interval>][&username=<username>][&password=<password>][&domain=<ntlm_domain>][&disableBasicAuth=<false | true>]

To register a file using the upload method, use the following syntax:

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.FileMgmtApiServlet?action=registerFile&fileType=<1 | 19 | 20 | 22>&destName=<destination_filename>&importMethod=upload

validateFile

Validates a registered file, or uploads or imports a file into the CDSM and validates the file.

Parameter

- File type (required)

  The settings are:

  - 1—Coverage Zone file
  - 19—CDN Selector file
  - 20—Rule file
  - 22—NAS file
- File ID (required if the destination filename is not specified)—The format is FileInfo_*xxx,* where *xxx* is the ID of the file.
- Import method (required if ID is not specified)

  The settings are:

  - Import
  - Upload
- Destination filename (required if ID is not specified)

If the import method is set to import, the following parameters also apply:

- URL of the origin file (required)—For example, //myserver/folder/myfile.txt or <*protocol*>://myhost/myfile.txt. The protocols supported are:
  - http://
  - https://
  - ftp://
- TTL (optional)—Frequency, in minutes, with which the CDSM looks for changes in the source file. The default is 10. The range is from 1 to 1440.
- NTLM user domain name (optional)
- Username (optional)
- Password (optional)
- Disable Basic Authentication (optional)—The default is false.

  When set to true, NTLM headers cannot be stripped off to allow fallback to the basic authentication method.

Rules

- If the file ID is specified, all optional parameters are ignored.
- If the file ID is not specified, the destination filename and import method must be specified.

Return

A confirmation that the file is valid.

Syntax

To validate a registered file, use the following syntax:

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.FileMgmtApiServlet?action=validateFile&fileType=<1 | 19 | 20 | 22><&id=<FileInfo_id>

To import a file from an external HTTP, HTTPs, FTP, or CIFS server into the CDSM and validate it, use the following syntax:

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.FileMgmtApiServlet?action=validateFile&fileType=<1 | 19 | 20 | 22>&destName=<destination_filename>&importMethod=import&originUrl=<file_url> [&ttl=<update_interval>][&username=<username>][&password=<password>][&domain=<ntlm_domain>][&disableBasicAuth=<false | true>]

To upload a file from a location accessible from your PC and validate it, use the following syntax:

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.FileMgmtApiServlet?action=validateFile&fileType=<1 | 19 | 20 | 22>&destName=<destination_filename>&importMethod=upload

Note Regardless of whether you use the upload or import method to validate the file, the validate action does not register the file with the CDSM.

refetchFile

Notifies the CDSM to refetch a registered file immediately.

Note The refetchFile action applies to files registered using the import method only.

Parameter

- File type (required)

  The settings are:

  - 1—Coverage Zone file
  - 19—CDN Selector file
  - 20—Rule file
  - 22—NAS file
- File ID (required)—The format is FileInfo_*xxx,* where *xxx* is the ID of the file.

Return

A confirmation that the file will be refetched shortly.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.FileMgmtApiServlet?action=refetchFile&fileType=<1 | 19 | 20 | 22 >&id=<FileInfo_id>

modifyFile

Modifies the metadata of a file or modifies either the credentials or TTL settings that control the updates to the file.

Parameter

- File type (required)

  The settings are:

  - 1—Coverage Zone file
  - 19—CDN Selector file
  - 20—Rule file
  - 22—NAS file
- File ID (required)—The format is FileInfo_*xxx,* where *xxx* is the ID of the file.
- Import method (required if ID is not specified)

  The settings are:

  - Import
  - Upload
- Destination filename

If the import method is set to import, the following parameters also apply:

- URL of the origin file (required)—For example, //myserver/folder/myfile.txt or <*protocol*>://myhost/myfile.txt. The protocols supported are:
  - http://
  - https://
  - ftp://
- TTL (optional)—Frequency, in minutes, with which the CDSM looks for changes in the source file. The default is 10. The range is from 1 to 1440.
- NT LAN Manager (NTLM) user domain name (optional)
- Username (optional)
- Password (optional)
- Disable Basic Authentication (optional)—The default is false.

  When set to true, NTMLM headers cannot be stripped off to allow fallback to the basic authentication method.

Return

A confirmation that the file was successfully modified.

Syntax

To modify the settings of a file that was imported into the CDSM from an external server, use the following syntax:

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.FileMgmtApiServlet?action=modifyFile&fileType=<1 | 19 | 20 | 22>&id=<FileInfo_id>&destName=<destination_filename>&importMethod= import&originUrl=<file_url>[&ttl=<update_interval>][&username=<username>][&password=<password>][&domain=<ntlm_domain>][&disableBasicAuth=<false | true>]

To modify the settings of a file that was uploaded from a location accessible from your PC into the CDSM, use the following syntax:

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.FileMgmtApiServlet?action=modifyFile&fileType=<1 | 19 | 20 | 22>&id=<FileInfo_id>&destName=<destination_filename>&importMethod=upload

deleteFile

Removes a registered file from the CDSM.

Parameter

- File type (required)

  The settings are:

  - 1—Coverage Zone file
  - 19—CDN Selector file
  - 20—Rule file
  - 22—NAS file
- File ID (required)—The format is FileInfo_*xxx,* where *xxx* is the ID of the file.

Rule

You can only delete files that are not currently in use.

Return

A confirmation that the file has been deleted.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.FileMgmtApiServlet?action=deleteFile&fileType=<1 | 19 | 20 | 22>&id=<FileInfo_id>

listFile

Displays the details of a specified registered file or displays the details of all registered files of a specified file type.

Parameter

- File type (required)

  The settings are:

  - 1—Coverage Zone file
  - 19—CDN Selector file
  - 20—Rule file
  - 22—NAS file
- File ID (optional)—The format is FileInfo_*xxx,* where *xxx* is the ID of the file.

Return

If a file ID was specified, the details of the requested file are listed. If no file ID was provided, the message lists all files of the specified type and their details. If no files of the specified type exist in the CDSM, the message returned indicates that the request was successful but warns that no files of the specified type exist.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.FileMgmtApiServlet?action=listFile&fileType=<1 | 19 | 20 | 22>[&id=<FileInfo_id>]

applyCZ

Applies a Coverage Zone file to an SR, removes a Coverage Zone file from an SR, or configures global routing.

Parameter

- Target—Deploy a Coverage Zone file globally to the entire CDS network or deploy a Coverage Zone file on the specified SR only. Valid values are:
  - Global
  - SR ID —The format is CrConfig_*xxx*, where *xxx* is the ID of the active SR.
- Coverage Zone file ID. Valid values are:
  - None—Removes the association of the Coverage Zone file with the target.
  - File ID (required)—The format is FileInfo_*xxx,* where *xxx* is the ID of the file.

If the target is set to global, the following parameter applies:

- DNS TTL (optional)—Time period (in seconds) for caching DNS replies. The default is 60 seconds. The range is from 0 to 60.

Return

Confirmation that the Coverage Zone file has been applied to the SR or that global routing has been set.

Syntax

To set global routing for a Coverage Zone file, use the following syntax:

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.FileMgmtApiServlet?action=applyCZ&target=global&czId=<FileInfo_id>[&dnsTtl=<dns_ttl>]

To reset global routing for a Coverage Zone file, use the following syntax:

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.FileMgmtApiServlet?action=applyCZ&target=global&czId=none

To apply a Coverage Zone file to an SR, use the following syntax:

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.FileMgmtApiServlet?action=applyCZ&target=<CrConfig_id>&czId=<FileInfo_id>

To remove a Coverage Zone file from an SR, use the following syntax:

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.FileMgmtApiServlet?action=applyCZ&target=<CrConfig_id>&czId=none

applyCdnSelector

Assigns a CDN Selector file to an SR or unassigns a CDN Selector file from an SR.

Parameter

- SR ID— The format is CrConfig_*xxx,* where *xxx* is the ID of the SR.
- CDN Selector file. Valid values are:
  - None—Unassigns the CDN Selector file from the SR.
  - File ID (required)—The format is FileInfo_*xxx,* where *xxx* is the ID of the file.

Return

Confirmation that the CDN Selector file has been assigned to the SR or unassigned from the SR.

Syntax

https://<cdsmIpAddress>:8443/servlet/com.cisco.unicorn.ui.FileMgmtApiServlet?action=applyCdnSelector&SR=<CrConfig_id>&cdnSelector=<none | FileInfo_id>

CHAPTER 7

Request Routing Engine API

This chapter describes the Request Routing Engine API. The Request Routing Engine contains all the functionality that was part of the Service Router in Release 2.4 and previous releases. Because the Service Router now contains the Proximity Engine, the original Service Router functionality has been encapsulated and renamed the Request Routing Engine.

Request Routing Engine API

The Request Routing Engine API allows another platform's software client to make queries, in the form of an HTTP request, to the Request Routing Engine about which Service Engine the Request Routing Engine selects.

Note The Request Routing Engine API is a Release 2.5.9 and later releases feature and does not support service-aware routing.

Parameter

The Request Routing Engine determines the best Service Engine based on the client IP address and requested URL. Therefore, the platform's software client must include these two parameters in the API query.

- Service Router IP address (required)
- CDNURL (required)
- ClientIP (required)

Request Syntax

http://*ServiceRouterIP*/routeURL?CDNURL=<*Requested URL*>&ClientIP=<*ClientIP*>

Request Example

The following is an example of a client's API HTTP request:

```
http://10.252.250.118/routeURL?CDNURL=http://cds.mov/smooth_linear/cnn.isml/victory&Client
IP=165.137.13.117
```

In this example, the request parameters have the following values:

- Service Router IP address—10.252.250.118
- CDNURL—http://cds.mov/smooth_linear/cnn.isml/victory

- ClientIP—192.0.2.121

Return

The Request Routing Engine returns an HTTP response with an XML payload with the following information:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<RoutedURLResponse>
<primaryContentRoutedURL>SRresponse</primaryContentRoutedURL>
</RoutedURLResponse>
```

The response identifies the name of the Service Engine the Request Routing Engine selected.

Return Example

For the example HTTP request described above, the HTTP response would be the following:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<RoutedURLResponse>
<primaryContentRoutedURL>http://str01-hub03.se.cds.mov/smooth_linear/cnn.isml/victory</pri
maryContentRoutedURL>
</RoutedURLResponse>
```

In this example, str-01-hub03 is the Service Engine the Request Routing Engine selected.

Note When there are API query requests for clientaccesspolicy.xml or crossdomain.xml, the requests are treated like normal API request and the XML payload is sent in the HTTP response as described above.

If requests for clientaccesspolicy.xml or crossdomain.xml are sent directly to the Request Routing Engine (non-API request), then these files are served. For more information about the Cross-Domain Policy, refer to *Cisco Internet Streamer CDS 2.6 Software Configuration Guide*. See the "Related Publications" section on page ix for links to documentation online.

CHAPTER 8

Proximity Engine SOAP APIs

This chapter provides an overview of the Proximity Engine, describes the SOAP (Simple Object Access Protocol) APIs exposed by the Proximity Engine, and presents the Web Service Definition Language (WSDL) file used by Proximity Engine to define proximity services. This chapter contains the following sections:

Routing Concepts and Overview

You should be familiar with the basics of IP routing and routing protocols, such as Open Shortest Path First (OSPF), Intermediate System-to-Intermediate System (IS-IS), and Border Gateway Protocol (BGP). Each Proximity Engine operates in an IP routing domain where the Interior Gateway Protocol (IGP) or BGP is used to distribute routing information across the domain.

Routers running OSPF or IS-IS establish adjacencies with their directly connected neighbors and exchange their connectivity view (that is, each router advertises its visibility about its adjacencies). Advertisements are flooded throughout the whole routing area and each router stores each received advertisement in a link-state database (LSDB).

The LSDB contains the topology of the whole network and each router uses it in order to compute the Shortest Path Tree and the Routing Information Base (RIB) that contains each known IP prefix in the network and its corresponding next-hop.

Each Proximity Engine leverages the LSDB in order to deliver a proximity service to its clients (Service Routers [SRs]). In order to build the LSDB, the Proximity Engine establishes adjacencies with routers running IGP. In the absence of Proximity Engine IGP peering and an LSDB, the Proximity Engine can still leverage the BGP attributes to deliver a proximity service.

Terminology

An SR sends a proximity request to a Proximity Engine. The proximity request specifies a source IP address and a set of one or more target IP addresses. The following terminology is used for these items:

- *Proximity source address* (PSA)—IP address from which the proximity needs to be computed.
- *Proximity target address* (PTA)—IP address to which (from the PSA) proximity has to be computed.
- *Proximity target list* (PTL)—List of PTAs that need to be evaluated (that is, the proximity from each of these proximity target addresses to the proximity source address needs to be computed).
- *Ranking depth*—Integer number that determines the length of the ranking list. For example, an SR can request the proximity of 10 nodes out of a PTL of 20 IP addresses.

The scope of the proximity service is to determine the distance between two IP addresses in a routing area. The SR requests the Proximity Engine to rank a list of IP addresses (PTL) based on the individual distance of each PTA from the PSA.

The Proximity function takes into account:

- Routing topology
- Inter-Autonomous System (AS) reachability
- Optimal path taken by the requested data

Proximity SOAP API Actions

Note The Proximity Engine APIs are available on the CDE205 and CDE220-2G2 platforms.

The Proximity Engine exposes a SOAP interface on port 7003. The SOAP interface of the Proximity Engine implements a rate operation to calculate the proximity rating of a group of PTAs. For additional information on the services of the SOAP interface, see the "Proximity Engine WSDL File" section on page 8-8.

Note The SOAP API and associated port 7003 are only available when proximity-based routing is enabled on the SR.

Note The Proximity Engine implements a legacy group operation to return grouping information for a PSA. This operation may be required in future release, but is not required in the current release as the rate operation returns grouping information for the PSA and the PTAs. The group operation is not specified in this document.

rate API

Requests the Proximity Engine to calculate the proximity of a list of PTAs to a PSA.

Request

The following are the input parameters of the rate request:

- _client—String representing the IP address of the PSA in the format of A.B.C.D.
- _destinations—PTL representing the list of PTAs.
- _category—Type of proximity service requested. Currently, the Proximity Engine only supports network-routing-based proximity which is represented by a value of 2.
- _resultCount—Number of results (that is, PTAs) the client requests to be returned. It refers to Ranking Depth concept described in the "Routing Concepts and Overview" section on page 8-1.

The rate operation is invoked by sending the following input message:

```
struct ns1__rate
{
        char *_client;
        struct ns2__ArrayOfJavaLangstring_USCOREliteral *_destinations;
        int _category;
        int _resultCount;
};
```

The array pointed to by the '_destinations' parameter is represented by an array in the following format:

```
struct ns2__ArrayOfJavaLangstring_USCOREliteral
{
        int __sizeJavaLangstring;
        char **JavaLangstring;
};
```

Request Example

The following section shows an example of a rate request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:prox="http://com/cisco/topos/proximity" xmlns:java="java:com.cisco.topos.proximity">
   <soapenv:Header/>
   <soapenv:Body>
      <prox:rate>
         <client>209.165.201.1</client>
         <destinations>
            <!--Zero or more repetitions:-->
            <java:JavaLangstring>209.165.201.10</java:JavaLangstring>
            <java:JavaLangstring>209.165.201.8</java:JavaLangstring>
            <java:JavaLangstring>209.165.201.11</java:JavaLangstring>
            <java:JavaLangstring>209.165.201.16</java:JavaLangstring>
            <java:JavaLangstring>209.165.201.5</java:JavaLangstring>
            <java:JavaLangstring>209.165.201.7</java:JavaLangstring>
         </destinations>
         <category>2</category>
         <resultCount>100</resultCount>
      </prox:rate>
   </soapenv:Body>
</soapenv:Envelope>
```

Response

The following are the output parameters of the rate response:

- Address—IP address of the PTA.
- Masklen—Prefix length of the subnet to which the PTA IP Address belongs.
- GroupId (optional)—Configured group for each IP address. For example, AS number, Masklen, or a community. Currently, the Proximity Engine does not return an Group Id.
- Rating—Proximity rating of the PTA based on the proximity algorithm.

The data structure of the rate response is as follows:

```
struct ns1__rateResponse
{
        struct ns2__ArrayOfServiceRatedAddress_USCOREliteral *_RatedAddressArray;
};
```

The ranked PTAs are returned in the ServiceRatedAddress array pointed to by the RatedAddressArray in the following format:

```
struct ns2__ArrayOfServiceRatedAddress_USCOREliteral
{
        int __sizeServiceRatedAddress;
        struct ns2__ServiceRatedAddress *ServiceRatedAddress;
};
```

Note The first member of the ServiceRatedAddress array is the PSA.

Each member of the ServiceRatedAddress is returned in the following format:

```
struct ns2__ServiceRatedAddress
{
        char *Address;
        int Masklen;
        LONG64 *GroupId;
        int Rating;
};
```

The array of PTAs is returned in ranked order of ascending *rating* with the PSA being the first member of the array.

Response Example

The following section shows an example of a rate response:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:ns2="java:com.cisco.topos.proximity"
xmlns:ns1="http://com/cisco/topos/proximity">
   <SOAP-ENV:Header/>
   <SOAP-ENV:Body>
      <ns1:rateResponse>
         <RatedAddressArray>
            <ns2:ServiceRatedAddress>
               <ns2:Address>209.165.201.1</ns2:Address>
               <ns2:Masklen>32</ns2:Masklen>
               <ns2:Rating>0</ns2:Rating>
            </ns2:ServiceRatedAddress>
            <ns2:ServiceRatedAddress>
               <ns2:Address>209.165.201.5</ns2:Address>
               <ns2:Masklen>32</ns2:Masklen>
```

```
                <ns2:Rating>10</ns2:Rating>
             </ns2:ServiceRatedAddress>
             <ns2:ServiceRatedAddress>
                <ns2:Address>209.165.201.7</ns2:Address>
                <ns2:Masklen>32</ns2:Masklen>
                <ns2:Rating>10</ns2:Rating>
             </ns2:ServiceRatedAddress>
             <ns2:ServiceRatedAddress>
                <ns2:Address>209.165.201.8</ns2:Address>
                <ns2:Masklen>20</ns2:Masklen>
                <ns2:Rating>20</ns2:Rating>
             </ns2:ServiceRatedAddress>
             <ns2:ServiceRatedAddress>
                <ns2:Address>209.165.201.10</ns2:Address>
                <ns2:Masklen>129</ns2:Masklen>
                <ns2:Rating>70</ns2:Rating>
             </ns2:ServiceRatedAddress>
             <ns2:ServiceRatedAddress>
                <ns2:Address>209.165.201.11</ns2:Address>
                <ns2:Masklen>129</ns2:Masklen>
                <ns2:Rating>70</ns2:Rating>
             </ns2:ServiceRatedAddress>
             <ns2:ServiceRatedAddress>
                <ns2:Address>209.165.201.16</ns2:Address>
                <ns2:Masklen>129</ns2:Masklen>
                <ns2:Rating>70</ns2:Rating>
             </ns2:ServiceRatedAddress>
          </RatedAddressArray>
       </ns1:rateResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Fault Message

The gSOAP server supports a number of faults, each representing a different error scenario. In most cases, a fault is simply a string describing a fault condition. Table 8-1 presents the list of faults specific to the Proximity Engine and provides an explanation for each fault.

Table 8-1 Proximity Engine Faults

Proximity Engine Fault	Explanation
Proximity Service Failed: Urib[1] failed to send request to protocol	There is a communication problem between the URIB and the IGP or BGP daemons.
Proximity Service Failed: Urib failed to receive from protocol	There is a communication problem between the URIB and the IGP or BGP daemons.
Proximity Service Failed: Route lookup failed in URIB	The PSA cannot be resolved by URIB. That is, the PSA cannot be found in the IP routing table.
Proximity Service Failed: Unrecognized exception	An unknown error has occurred.

1. URIB = unicast routing information base

Fault Response

The data structures of a SOAP fault response are as follows:

```
/* SOAP Fault Code: */
```

```
struct SOAP_ENV__Code
{
        char *SOAP_ENV__Value;
        struct SOAP_ENV__Code *SOAP_ENV__Subcode;
};

/* SOAP-ENV:Detail */
struct SOAP_ENV__Detail
{
        int __type;
        void *fault;            char *__any;
};
/* SOAP-ENV:Reason */
struct SOAP_ENV__Reason
{
        char *SOAP_ENV__Text;
};
/* SOAP Fault: */
struct SOAP_ENV__Fault
{
        char *faultcode;
        char *faultstring;
        char *faultactor;
        struct SOAP_ENV__Detail *detail;
        struct SOAP_ENV__Code *SOAP_ENV__Code;
        struct SOAP_ENV__Reason *SOAP_ENV__Reason;
        char *SOAP_ENV__Node;
        char *SOAP_ENV__Role;
        struct SOAP_ENV__Detail *SOAP_ENV__Detail;
};
```

A redirect fault occurs when the Proximity Engine does not consider itself the most appropriate Proximity Engine to service the request, and redirects the proximity client to a set of Proximity Engines it considers more appropriate.

Redirect Response

The Proximity Engine uses the following redirect response data structure wrapped inside the SOAP_ENV_FAULT data structure to send a redirect fault:

```
struct ns2__RedirectResponse
{
/// @brief PSA grouping info.
/// Element psa of type java:com.cisco.topos.proximity":ServiceGroupRange.
    struct ns2__ServiceGroupRange*       psa                             1;  ///< Required
element.
/// @brief SG Endpoints for the redirect.
/// Element EndPoints of type "java:com.cisco.topos.proximity":EndPoints.
    struct ns2__EndPoints*               EndPoints                       1;  ///< Required
element.
};
```

The Proximity Engine returns as part of this fault the group information of the PSA and the list of EndPoints. The data structure of the PSA group information is as follows:

struct ns2__ServiceGroupRange

```
{
        char *Address;
        int Masklen;
        LONG64 *GroupId;
};
```

The Proximity Engine also returns as part of this fault the list of EndPoints containing the list of IP addresses of other Proximity Engines in the network that are more appropriate for this proximity request. The data structure of the list of EndPoints is as follows:

```
struct _ns2__EndPoints
{
        int __sizeEndPoint;
        char **EndPoint;
};
```

Proximity Engine WSDL File

A WSDL file is an XML-formatted file that describes the details as a web service and specifies the operations it knows how to perform. The WSDL file for the SOAP interface of the Proximity Engine is as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<s0:definitions name="NetworkProximityServiceDefinitions"
targetNamespace="http://com/cisco/topos/proximity" xmlns=""
xmlns:s0="http://schemas.xmlsoap.org/wsdl/" xmlns:s1="http://www.w3.org/2001/XMLSchema"
xmlns:s2="java:com.cisco.topos.proximity" xmlns:s3="http://com/cisco/topos/proximity"
xmlns:s4="http://schemas.xmlsoap.org/wsdl/soap/">
  <s0:types>
    <xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="java:com.cisco.topos.proximity"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:complexType name="ServiceRatedAddress">
          <xs:annotation>
            <xs:documentation>Rated IP address.</xs:documentation>
          </xs:annotation>
          <xs:sequence>
              <xs:element minOccurs="1" name="Address" nillable="false" type="xs:string">
                <xs:annotation>
                  <xs:documentation>Get IP address.</xs:documentation>
                </xs:annotation>
              </xs:element>
              <xs:element minOccurs="1" name="Masklen" nillable="false" type="xs:int">
                <xs:annotation>
                  <xs:documentation>Get subnet masklen.</xs:documentation>
                </xs:annotation>
              </xs:element>
                <xs:element minOccurs="0" name="GroupId" nillable="false" type="xs:long">
                <xs:annotation>
                  <xs:documentation>Get group Id.</xs:documentation>
                </xs:annotation>
              </xs:element>
            <xs:element minOccurs="1" name="Rating" nillable="false" type="xs:int">
              <xs:annotation>
                <xs:documentation>Get rating.</xs:documentation>
              </xs:annotation>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      <xs:complexType name="ServiceGroupRange">
        <xs:annotation>
          <xs:documentation>Group Id with addresss range.</xs:documentation>
        </xs:annotation>
        <xs:sequence>
          <xs:element minOccurs="1" name="Address" nillable="false" type="xs:string">
            <xs:annotation>
              <xs:documentation>Get IP address.</xs:documentation>
            </xs:annotation>
          </xs:element>
          <xs:element minOccurs="1" name="Masklen" nillable="false" type="xs:int">
            <xs:annotation>
              <xs:documentation>Get subnet masklen.</xs:documentation>
            </xs:annotation>
          </xs:element>
            <xs:element minOccurs="0" name="GroupId" nillable="false" type="xs:long">
            <xs:annotation>
              <xs:documentation>Get group Id.</xs:documentation>
            </xs:annotation>
```

```
          </xs:element>
        </xs:sequence>
      </xs:complexType>
      <xs:element name="psa" type="java:ServiceGroupRange"
xmlns:java="java:com.cisco.topos.proximity">
            <xs:annotation>
                <xs:documentation>PSA grouping info</xs:documentation>
              </xs:annotation>
      </xs:element>
        <xs:complexType name="EndPoints">
            <xs:annotation>
                <xs:documentation>SG EndPoints for the redirect</xs:documentation>
            </xs:annotation>
            <xs:sequence>
                <xs:element name="EndPoint" maxOccurs="unbounded" minOccurs="1"
nillable="false" type="xs:string"/>
            </xs:sequence>
        </xs:complexType>
        <xs:complexType name="RedirectResponse">
          <xs:annotation>
            <xs:documentation>Redirect response.</xs:documentation>
          </xs:annotation>
          <xs:sequence>
              <xs:element maxOccurs="1" minOccurs="1" name="psa" nillable="false"
              type="java:ServiceGroupRange"
              xmlns:java="java:com.cisco.topos.proximity">
                <xs:annotation>
                  <xs:documentation>PSA grouping info.</xs:documentation>
                </xs:annotation>
              </xs:element>
              <xs:element maxOccurs ="1" minOccurs="1" name="EndPoints" nillable="false"
              type="java:EndPoints" xmlns:java="java:com.cisco.topos.proximity">
                <xs:annotation>
                  <xs:documentation>SG Endpoints for the redirect.</xs:documentation>
                </xs:annotation>
              </xs:element>
          </xs:sequence>
        </xs:complexType>
      <xs:complexType name="ArrayOfServiceRatedAddress_literal">
        <xs:sequence>
          <xs:element maxOccurs="unbounded" minOccurs="0" name="ServiceRatedAddress"
nillable="true" type="java:ServiceRatedAddress"
xmlns:java="java:com.cisco.topos.proximity"/>
        </xs:sequence>
      </xs:complexType>
      <xs:element name="ArrayOfServiceRatedAddress_literal"
type="java:ArrayOfServiceRatedAddress_literal"
xmlns:java="java:com.cisco.topos.proximity"/>
      <xs:complexType name="ArrayOfServiceGroupRange_literal">
        <xs:sequence>
          <xs:element maxOccurs="unbounded" minOccurs="0" name="ServiceGroupRange"
nillable="true" type="java:ServiceGroupRange"
xmlns:java="java:com.cisco.topos.proximity"/>
        </xs:sequence>
      </xs:complexType>
      <xs:element name="ArrayOfServiceGroupRange_literal"
type="java:ArrayOfServiceGroupRange_literal" xmlns:java="java:com.cisco.topos.proximity"/>
      <xs:complexType name="ArrayOfJavaLangstring_literal">
        <xs:sequence>
          <xs:element maxOccurs="unbounded" minOccurs="0" name="JavaLangstring"
nillable="true" type="xs:string"/>
        </xs:sequence>
      </xs:complexType>
```

```
      <xs:element name="ArrayOfJavaLangstring_literal"
type="java:ArrayOfJavaLangstring_literal" xmlns:java="java:com.cisco.topos.proximity"/>
    </xs:schema>
  </s0:types>
  <s0:message name="rate">
    <s0:part name="client" type="s1:string"/>
    <s0:part name="destinations" type="s2:ArrayOfJavaLangstring_literal"/>
    <s0:part name="category" type="s1:int"/>
    <s0:part name="resultCount" type="s1:int"/>
  </s0:message>
  <s0:message name="rateResponse">
    <s0:part name="RatedAddressArray" type="s2:ArrayOfServiceRatedAddress_literal"/>
  </s0:message>
  <s0:message name="group">
    <s0:part name="addresses" type="s2:ArrayOfJavaLangstring_literal"/>
  </s0:message>
  <s0:message name="groupResponse">
    <s0:part name="GroupRangeArray" type="s2:ArrayOfServiceGroupRange_literal"/>
  </s0:message>
  <s0:portType name="NetworkProximityPortType">
    <s0:operation name="rate" parameterOrder="client destinations category resultCount">
      <s0:input message="s3:rate"/>
      <s0:output message="s3:rateResponse"/>
    </s0:operation>
    <s0:operation name="group" parameterOrder="addresses">
      <s0:input message="s3:group"/>
      <s0:output message="s3:groupResponse"/>
    </s0:operation>
  </s0:portType>
  <s0:binding name="NetworkProximityServiceSoapBinding"
type="s3:NetworkProximityPortType">
    <s4:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
    <s0:operation name="rate">
      <s4:operation soapAction="" style="rpc"/>
      <s0:input>
        <s4:body namespace="http://com/cisco/topos/proximity" parts="client destinations
category resultCount" use="literal"/>
      </s0:input>
      <s0:output>
        <s4:body namespace="http://com/cisco/topos/proximity" parts="RatedAddressArray"
use="literal"/>
      </s0:output>
    </s0:operation>
    <s0:operation name="group">
      <s4:operation soapAction="" style="rpc"/>
      <s0:input>
        <s4:body namespace="http://com/cisco/topos/proximity" parts="addresses"
use="literal"/>
      </s0:input>
      <s0:output>
        <s4:body namespace="http://com/cisco/topos/proximity" parts="GroupRangeArray"
use="literal"/>
      </s0:output>
    </s0:operation>
  </s0:binding>
  <s0:service name="NetworkProximityService">
    <s0:port binding="s3:NetworkProximityServiceSoapBinding"
name="NetworkProximityServicePort">
      <s4:address location="http://localhost:7003/"/>
    </s0:port>
  </s0:service>
</s0:definitions>
```

APPENDIX A

Program Files in the CDS Software

CDS software uses programs to enable support for live multicast and scheduled rebroadcast events. A program in the CDS software is defined as a scheduled event in which the content is presented to the end user. The three attributes of a program are:

- Schedule—Defines when the content is presented to the end user.
- Content—Defines what is presented to the end user. in the CDS software, this can be pre-positioned or live content.
- Presentation—Defines how the content is presented to the end user. The presentation attributes include the set of Service Engines that know about the program, and a service type that identifies the streaming server used to deliver the content. The streaming server can exist in the Service Engine (Windows Media Technology [WMT] or Movie Streamer).

A program file contains the elements that define the schedule, content, and presentation parameters. It is a text file written in XML format, similar to the Manifest file. For more information about Manifest files, refer to *Cisco Internet Streamer CDS 2.6 Software Configuration Guide*. See the "Related Publications" section on page ix for links to documentation online.

Program types determine the hardware or software component involved in delivering content to the user. Different program types are:

- Movie Streamer
- WMT

The CDSM manages multicast addresses to be used for programs. Each Service Engine assigned to the program uses the multicast address for broadcast. The Service Engine determines which multicast address is to be used based on the program data. A set of multicast addresses can be specified either in the Program API or by using the CDSM. Each time a program requires a multicast address, the CDSM associates one of the addresses with the program. Addresses are allocated for the life of a program. Programs can be configured with an autodelete feature, which allows program addresses to be freed up automatically about 24 hours after a program schedule is complete.

When users request a specific address or a set of addresses to be used for a program, CDS software issues only those addresses that are not used by any of the existing programs. Users receive an error message if there is no multicast address associated with the imported program file and no addresses are available to be configured from the pool or if the multicast pool has not been configured.

When you define a Movie Streamer live program using the createProgram API, you can specify a single backup broadcast server for the program. To do this, you must specify the IP addresses of the primary and backup broadcast servers in the program file using the <media> tag. The <media> tag in the program file should be in the following format:

<media index="*number*" src="*primary_broadcast_server*:*port*;*backup_broadcast_server*:*port*"/>

Program File DTD

The following is the Document Type Definition (DTD) for CDS program files. You can use the DTD to create program files for importing programs from third-party systems.

```
<?xml version="1.0"?>
<!DOCTYPE program[
  <!ELEMENT program (media+, ucastInfo?, mcastInfo?, schedule*, attribute?)>
  <!ATTLIST program
       version               CDATA "1.0"
       name                  CDATA #REQUIRED
       serviceType           (wmt | movieStreamer) "wmt"
       description           CDATA #IMPLIED
       playTime              CDATA #IMPLIED
       lastModificationTime  CDATA #IMPLIED
       gracefulExit          (false | true) "false"
       shuffle               (false | true) "false"
       autoDelete            (false | true | default) "default"
       blockPerSchedule      (false | true) "false"
       live                  (false | true) "false"
  >
  <!ELEMENT media EMPTY>
  <!ATTLIST media
       index                 CDATA #IMPLIED
       src                   CDATA #REQUIRED
       id                    CDATA #IMPLIED
       playTime              CDATA #IMPLIED
>
  <!ELEMENT ucastInfo EMPTY>
  <!ATTLIST ucastInfo
       referenceUrl          CDATA #REQUIRED
  >
  <!ELEMENT mcastInfo (addrPort*)>
  <!ATTLIST mcastInfo
       referenceUrl          CDATA #REQUIRED
       TTL                   CDATA #IMPLIED
  >
  <!ELEMENT addrPort EMPTY>
  <!ATTLIST addrPort
       addrVal               CDATA #REQUIRED
       portVal               CDATA #REQUIRED
       id                    CDATA #IMPLIED
  >
<!ELEMENT schedule (repeats?)>
  <!ATTLIST schedule
       timeSpec        (local | gmt) "local"
       startTime       CDATA #IMPLIED
       activeDuration  CDATA #IMPLIED
  >
  <!ELEMENT repeats (dayOffset*)>
  <!ATTLIST repeats
       type            (timeInterval | days) "timeInterval"
       interval        CDATA #IMPLIED
       endTime         CDATA #IMPLIED
  >
  <!ELEMENT dayOffset EMPTY>
  <!ATTLIST dayOffset
       value        (0 | 1 | 2 | 3 | 4 | 5 | 6) "0"
  >
  <!ELEMENT attribute EMPTY>
  <!ATTLIST attribute
       value         CDATA #REQUIRED
  >
```

```
]>
```

Table A-1 describes the elements in the DTD and their attributes.

Table A-1 Program File DTD Elements and Attributes

Element	Attributes	Description
program	version	Version of the program file. CDS software generates playlist files with a version level of 1.
	name	Name of the program.
	serviceType	Type of program, which dictates the mode of delivery. This element identifies the software or hardware component involved in delivering the content to the user.
	description	Brief description of the program.
	playTime	Total playtime in seconds. This is the sum of the playtime values of the media files, if set. If there are files in the program that have invalid playtimes, then this field is set to –1.
	lastModificationTime	Time when the playlist was created or modified last, as recorded in the CDSM. The format is hh:mm:ss. The assumption is that all devices in the CDS network are time-synchronized (for example, using the NTP[1].
	gracefulExit	Specifies how to handle scheduled exits. Options are: • True—Exit after the current media file is played completely. • False—Exit immediately.
	shuffle	Specifies whether the media files should play in any order. Options are: • True—Play media files at random. • False—Play media files in order. When this attribute is not specified, it is set to false by default.
	autoDelete	Specifies whether the program should be automatically deleted 24 hours after it is last played. Options are: • True—Delete the program 24 hours after it is last played. • False—Retain the program for more than 24 hours after it is last played. • Default—When the value for the live attribute is set to true, the default value is true for *autoDelete*, and false if the live attribute is set to false.

Table A-1 Program File DTD Elements and Attributes (continued)

Element	Attributes	Description
	blockPerSchedule	Specifies whether active streams should be terminated when the scheduled program ends. This attribute is used only when a live unicast event is scheduled to be delivered by the WMT streaming server. Options are: • True—WMT terminates active streams when the scheduled program ends. • False—WMT does not terminate active streams when the scheduled program ends. When this attribute is not specified, it is set to false by default.
	live	Specifies whether the program contains live content. Options are: • True—The program contains live content. • False—The program does not contain live content.
media	index	Order of the media file in the list of files, ranging from 1 to mediaCount (the number of media files in the program). The index attribute specifies the order of the media files when the shuffle attribute in the <media> tag is set to false.
	src	Reference to the source of the media file. • For live content, this field contains information about how the streaming server will correlate with the live feed. • For prefetched content, this field contains the portion of the URL that follows the origin server; that is, the FQDN[2]. For example, if the source file URL is http://mycontentorigin/mydirectory/myfile, the value assigned to this field is mydirectory/myfile. **Note** When prefetched content is exported, this field contains the URL for the file that can be routed in the CDS network, without the protocol specification. • Live source failover is supported. For WMT live, multiple encoders or streaming servers can be specified. src="http://encoder_1:8080;rtsp://source_hostip/filename" For Movie Streamer live, only a single backup can be specified. src="sourceaddress1:destinationport1;sourceaddress2: destinationport2"
	id	Media file identifier. For WMT rebroadcast events, this field contains the ID of the delivery service containing this media file. For Movie Streamer rebroadcast events, this field contains the track number. In the case of live events, this field is used to correlate a stream source with a multicast address. **Note** For live unicast programs, do not include the ID attribute.

Table A-1 Program File DTD Elements and Attributes (continued)

Element	Attributes	Description
	playTime	Playtime for the file in seconds, when it is known. This attribute is used only for MPG media files. Options are: • –2—If the file is not an MPG file • –1—If the file is an MPG file but the CDS software cannot determine the playtime • 0 or greater—If the playtime is correctly determined from the file
ucastInfo	referenceUrl	URL used by the end user to request this program over the network using unicast. **Note** All letters in the reference URLs must be in lowercase.
mcastInfo	referenceUrl	URL used by the end user to request this program over the network using multicast. **Note** All letters in the reference URLs must be in lowercase.
	TTL	Multicast TTL[3] value to be used for the packets sent using multicast.
addrPort	addrval	Address to be used when this program is multicast.
	portVal	Port (within the multicast address) to be used when this program is multicast.
	id	Address and port identifier. For rebroadcast events, this field contains the ID of the delivery service using this address and port. In the case of live events, this field is used to correlate a stream source with a multicast address.
schedule	timeSpec	Specifies how time values should be interpreted. Options are: • Local • GMT[4]
	startTime	Time (in seconds) since the epoch (January 1, 1970) when the program should start playing. **Tip** For UNIX operating systems, the epoch is 00:00:00 GMT, January 1, 1970. This represents the time and date corresponding to 0 in the UNIX operating system's date and time stamp. System time is measured in seconds past the epoch.
	activeDuration	Duration of the program (in seconds). For a scheduled rebroadcast, this value specifies how long the files should loop (that is, loop for *x* seconds). If there is no looping, this value is 0. For live programs, this value is the duration of the event.
repeats	type	Type of repeat. For example, you can set the program to repeat every *x* seconds, or repeat on specified days of the week at the same time specified in the start time. Options are: • TimeInterval • Days

Table A-1 Program File DTD Elements and Attributes (continued)

Element	Attributes	Description
	interval	Time interval (in seconds) for the repeat broadcast of the program. For example, if this value to 28800 seconds, the program repeats every 8 hours.
	endTime	Time (in seconds) since the epoch (January 1, 1970) when program repeats should end. For a program that repeats forever, enter the value zero (0).
dayOffset	value	Day to repeat the program, for example, every Monday. The time (during the day) of the repeat is inherited from the startTime attribute.
attribute	value	Element used if a third-party device is used to import some data that is transparent to a CDS network, and that is directly used by the software or hardware component involved in delivering the content to the user. The CMS[5] relays the data without interpreting it. A recommended method for encoding this field is to use a name/value pair in the string, for example, name1=value1; name2=value2.

1. NTP = Network Time Protocol.
2. FQDN = fully qualified domain name.
3. TTL = time-to-live.
4. GMT = Greenwich Mean Time.
5. CMS = Centralized Management System.

Program File Examples

This section contains program file examples, each describing the contents for specific event types. The examples are provided for the following event types:

- WMT Multicast Live Event, page A-6
- WMT Multicast Rebroadcast Event, page A-7
- Movie Streamer Multicast Event, page A-7
- Movie Streamer Live-Split Event, page A-8

WMT Multicast Live Event

The following example shows the program file for a WMT multicast live event in which the multicast address is specified using the addrPort element:

```
<?xml version="1.0"?>
<!DOCTYPE program SYSTEM "program.dtd">
<program version="1.0" name="liveProgram" serviceType="wmt" description="test"
autoDelete="true" blockPerSchedule="true" live="true">
<media index="1" src="http://WMT_encoder:8080" id="media0"/>
<mcastInfo referenceUrl="http://contentacquirer/liveprogram.nsc" TTL="22">
<addrPort addrVal="239.232.25.95" portVal="61248" id="media0"/>
</mcastInfo>
<schedule timeSpec="gmt" startTime="0" activeDuration="0"/>
</program>
```

WMT Multicast Rebroadcast Event

This example shows the program file for a WMT multicast rebroadcast event:

```
<?xml version="1.0"?>
<!DOCTYPE program SYSTEM "program.dtd">
<program version="1.0" name="chanrebroad" serviceType="wmt" description="test"
autoDelete="false" live="false">
<media index="1" src="sen/beck.asf" id="Channel_35748"/>
<media index="2" src="sen/CSCin53585.wmv" id="Channel_35748"/>
<media index="3" src="sen/starsnstripes.asf" id="Channel_35749"/>
<mcastInfo referenceUrl="http://contentacquirer/chanrebroad.nsc">
<addrPort addrVal="239.232.25.195" portVal="61248" id="Channel_35748"/>
</mcastInfo>
<schedule timeSpec="local" startTime="1010170800" activeDuration="1300">
<repeats type="timeInterval" interval="2600"/>
</schedule>
</program>
```

The *referenceUrl* attribute is the link that the user clicks to join the program. You can provide the external IP address of the Content Acquirer (for example, http://ServiceEngine/prog1.nsc) in the *referenceUrl* attribute.

Note A media file can be uniquely identified using a URL of the form <protocol>://<FQDN>/<relative_URL>. The *id* attribute in the media element specifies the ID of the delivery service containing the media file. Each delivery service is associated with the FQDN of a Service Engine or that of an origin server. The *src* attribute in the media element provides the relative part of the URL, which along with the *id* attribute identifies the file.

You can provide the FQDN of the Service Engine that hosts the media file if a Service Router is used to direct the user request to the appropriate Service Engine. In this case, the FQDN must be associated with a website or delivery service that maps to the same Service Engines that can serve the program.

You can provide the name of the Service Engine if the user request goes to a preselected Service Engine. If a third-party device assigns the Service Engines directly to the program, you can use any one of the Service Engines assigned to the program in the *referenceUrl* attribute. If the third-party device assigns a delivery service to the program, you can use the name of any Service Engine in that delivery service (for example, the Content Acquirer) in the *referenceUrl* attribute.

Movie Streamer Multicast Event

This example shows the program file for a Movie Streamer multicast event. This event can also be accessed using unicast by specifying the *referenceUrl* attribute in the ucastInfo element.

```
<?xml version="1.0" ?>
<!DOCTYPE program SYSTEM "program.dtd">
<program version="1.0" name="prog5lfs_1673" serviceType="movieStreamer"
description="prog5lfs" playTime="3600" autoDelete="false" live="true">
<media index="1" src="source_ip_address:destination_port" id="media0"/>
<media index="2" src="source_ip_address:destination_port" id="media1"/>
<ucastInfo referenceUrl="rtsp://pm_fqdn_or_ip_addr/pm_1673.sdp"/>
<mcastInfo referenceUrl="http://pm_fqdn_or_ip_addr/programs/1673" TTL="15">
<addrPort addrVal="224.2.250.195" portVal="61036" id="media0"/>
```

```
<addrPort addrVal="224.2.250.196" portVal="61038" id="media1"/>
</mcastInfo>
<schedule timeSpec="gmt" startTime="3264429600"/>
<attribute value="unicastPushSDP:http://2.43.12.6/programs/1673"/>
</program>
```

Note The media source (src) is the live feed. The *src* attribute contains the IP address of the Broadcast Server and the destination port of the Content Acquirer. The Content Acquirer listens for the program stream on the specified destination port. There is more than one media source, because audio, video, and other feeds may be broadcast on a separate stream, using a separate multicast address. The *id* attribute in the media element and the *id* attribute in the addrPort element are used to correlate the address to the stream.

Movie Streamer Live-Split Event

This example shows the program file for a Movie Streamer live-split event:

```
<?xml version="1.0" ?>
<!DOCTYPE program SYSTEM "program.dtd">
<program version="1.0" name="prog5lfs_1674" serviceType="movieStreamer"
description="prog52fs" playTime="3600" autoDelete="false" live="true">
<media index="1" src="source_ip_address:destination_port" />
<media index="2" src="source_ip_address:destination_port" />
<ucastInfo referenceUrl="rtsp://pm_fqdn_or_ip_addr/pm_1674.sdp"/>
<schedule timeSpec="gmt" startTime="3264429600" activeDuration="7200"/>
<attribute value="unicastPushSDP:http://2.43.12.6/programs/1673"/>
</program>
```

Note Attributes for the schedule element must be specified for the Movie Streamer streaming server. The *id* attribute is not required because there are no separate multicast addresses for the program streams.