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.