Introduction to the Cisco WAAS Central Manager Monitoring API

This chapter describes the Cisco WAAS Central Manager monitoring application programming interface (API), which provides a programmable interface for system developers to integrate with customized or third-party monitoring and management applications.

This chapter contains the following sections:

Monitoring API Overview

The Central Manager monitoring API communicates with the WAAS Central Manager to retrieve status information and monitoring statistics. This API does not allow device configuration.

The Central Manager monitoring API is a Web Service implementation. Web Service is defined by the W3C standard as a software system designed to support interoperable machine-to-machine (client and server) interaction over the network. The client and server communication follows the Simple Object Access Protocol or Service Oriented Architecture Protocol (SOAP) standard.

SOAP, which exchanges XML-based messages over the network using HTTP or HTTPS, is the foundation layer of the Web Service stack. It provides a basic messaging framework that allows more abstract layers to build on. SOAP encoding wraps XML headers and tags in a SOAP envelope.

To call a service, you connect to a particular Central Manager through a web browser by using a service URL that contains the IP address or hostname of the Central Manager and the name of the particular monitoring service (such as DeviceConf or TrafficStats). For example, https://<host/ip>:8443/ws/TrafficStats is the service URL for the Traffic Acceleration Service.

Next, you must post a SOAP request written in an XML format to retrieve the information. The request calls for a particular action (such as retrieveTrafficStats) and contains the WS-Security header (username and password) and the input parameter content when required. The Central Manager responds with a SOAP envelope that contains the answer in an XML format. The response contains the output values for this action.

SOAP message exchanges follow the WS-Security specification. The WS-Security specification provides a Username Token mechanism to authenticate SOAP message exchanges.

The following example shows an XML-formatted SOAP request to perform the getAPIVersion action. There are no input parameters for this particular action. The example then shows the SOAP response that contains the output values for this action, such as the hostname, IP address, location, MAC address, and so forth.

Request

```
POST https://10.64.62.177:8443/ws/DeviceConf HTTP/1.1
Accept-Encoding: gzip,deflate
Content-Type: text/xml;charset=UTF-8
SOAPAction: "urn:getWANInfo"
User-Agent: Jakarta Commons-HttpClient/3.1
Host: 10.64.62.177:8443
Content-Length: 397
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<soapenv:Header>
<wsse:Security
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.
xsd">
<wsse:UsernameToken>
    <wsse:Username>admin</wsse:Username>
    <wsse:Password>default</wsse:Password>
    </wsse:UsernameToken>
    </wsse:Security>
    </soapenv:Header>
<soapenv:Body/>
</soapenv:Envelope>
```

Response

```
HTTP/1.1 200 OK
Date: Mon, 15 Nov 2010 05:57:55 GMT
Server: Apache/1.3.41 (Unix) mod_ssl/2.8.31 OpenSSL/0.9.8g mod_jk/1.2.15 mod_auth_pam/1.0a
Transfer-Encoding: chunked
Content-Type: text/xml;charset=UTF-8
<?xml version='1.0' encoding='UTF-8'?><soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"><soapenv:Body><ns:getWANInfoResp
onse xmlns:ns="http://config.ws.waas.cisco.com" xmlns:ax22="http://io.java/xsd"
xmlns:ax23="http://config.ws.waas.cisco.com/xsd" xmlns:ax21="http://rmi.java/xsd">
<ns:return type="com.cisco.waas.ws.config.Device">
<ax23:hostName>gowri-device-1</ax23:hostName>
<ax23:id>180</ax23:id><ax23:ipAddress>10.64.62.177</ax23:ipAddress>
<ax23:location></ax23:location><ax23:macAddress>00:21:5e:57:a7:2c</ax23:macAddress>
<ax23:model>OE674</ax23:model>
<ax23:name>gowri-device-1</ax23:name>
<ax23:role>Primary</ax23:role>
<ax23:softwareVersion>4.3.1.0.1</ax23:softwareVersion>
<ax23:status>Online</ax23:status>
<ax23:type>CM</ax23:type>
</ns:return><ns:return type="com.cisco.waas.ws.config.Device">
<ax23:hostName>datacenter-wae</ax23:hostName>
<ax23:id>345</ax23:id><ax23:ipAddress>192.168.3.2</ax23:ipAddress>
<ax23:location>datacenter-wae-location</ax23:location>
<ax23:macAddress>00:21:5e:57:a7:6a</ax23:macAddress>
<ax23:model>OE674</ax23:model>
<ax23:name>datacenter-wae</ax23:name>
<ax23:role>Application Accelerator</ax23:role>
<ax23:softwareVersion>4.3.0-npe</ax23:softwareVersion>
<ax23:status>Online</ax23:status>
<ax23:type>WAE</ax23:type>
</ns:return><ns:return type="com.cisco.waas.ws.config.Device">
<ax23:hostName>we-3900-1</ax23:hostName>
<ax23:id>323</ax23:id>
```

```
<ax23:ipAddress>10.64.62.167</ax23:ipAddress>
<ax23:location>we-3900-1-location</ax23:location>
<ax23:macAddress>88:43:e1:99:46:80</ax23:macAddress>
<ax23:model>Cisco (CISCO3945-CHASSIS) </ax23:model>
<ax23:name>we-3900-1</ax23:name>
<ax23:role>WAAS Express</ax23:role>
<ax23:softwareVersion>15.1(2)T2/1.1.0</ax23:softwareVersion>
<ax23:status>Online</ax23:status>
<ax23:type>WAE</ax23:type>
</ns:return></ns:getWANInfoResponse></soapenv:Body></soapenv:Envelope>
```

The Central Manager monitoring API consists of the following eight Web Services:

- Device Configuration
- Traffic Acceleration
- CIFS
- Video Stream
- HTTP and HTTPS
- MAPI
- NFS
- Events and Status

Administrators may control API access for a device or device group by configuring user authorization settings using the CLI or the Central Manager GUI. The authorization for a Web Service is implemented system wide as mandatory and at the service level as optional.

Web Services Description Language

In the Central Manager monitoring API, Web Services Description Language (WSDL) is used with SOAP and XML schemas to provide Web Services. WSDL is an XML-based service that describes the functionality offered by the Web Service and defines the actions, parameter names, input parameter data types, and return data types for the Web Service. When you connect to a Web Service through a web browser, you can read the WSDL file to determine which functions are available on the server. Any special data types that are used are embedded in the WSDL file in an XML schema. You can then call one of the functions listed in the WSDL file by sending a SOAP request message.

To obtain the WSDL file defined for a particular service in the Central Manager monitoring API implementation, submit a URL to the service with a **?wsdl** suffix. For example, to retrieve the WSDL for the TrafficStats service running on https://localhost:8443/ws/TrafficStats, call the WSDL file by using the URL **https://localhost:8443/ws/TrafficStats?wsdl**.

Using the Central Manager Monitoring API

This section describes how to use the Central Manager monitoring API. It contains the following topics:

- Required Software, Web Standards, and Supported Hardware
- Generating the Client Code to Invoke a Web Service

Required Software, Web Standards, and Supported Hardware

The Central Manager monitoring API is supported in WAAS version 4.1.1 and later. The API functions on the following types of development environment:

- Apache Axis2 (Version 2.1.3)
- WSDL Support: 1.1 or 2.0
- SOAP 1.1 or 1.2
- Axis Data Binding (ADB)
- WS-Security

Generating the Client Code to Invoke a Web Service

You can use the WSDL2java utility to generate the client code which calls and implements a Web Service. The WSDL2java utility takes a WSDL document and generates fully annotated Java code from which to implement the service.

To use the WSDL2java utility, follow these steps:

Step 1 Query the Central Manager for the WDSL definitions for a particular service by using the following WSDL URL format:

```
https://<host/ip>:8443/ws/<NameOfService>?wsdl
```

where the host/ip value is the hostname or IP address of the Central Manager that has the service running, and the NameOfService value is the Web Service designation.

Step 2 Save the XML response to a file, such as NameOfService.xml.

Step 3 Call the WSDL2java script for your development environment: wsdl2java.sh or wsdl2java.bat. These scripts can be found under the bin directory of the Axis2 distribution.

Step 4 Run the following command line to generate the client code:

```
wsdl2java -uri NameOfService.xml -p com.cisco.waas.wsc -d adb -s
```

The WSDL2Java command is run against the WSDL file to create deployment descriptor templates. The utility processes the WSDL file and generates JAVA code based on the WDSL definitions for a particular service.

You may then create scripts using any general-purpose, high-level programming language, such as Python, to generate SOAP requests and parse SOAP responses. A sample of the soap request and response in Perl script is provided at this location: https://supportforums.cisco.com/docs/DOC-13202

Monitoring API Version Compatibility

As the monitoring API is enhanced with new WAAS software versions, API changes will occur that may or may not be compatible with existing client code.

The following kinds of monitoring API changes should be considered backward compatible and existing API users should be able to seamlessly integrate with future versions without any changes required to clients:

- Adding a new operation—For example, adding the retrieveResponseStats operation for the HttpStats service. Existing clients continue invoking existing operations while new operations are available for new clients. Existing clients are compatible.
- Adding new optional data structures to the request message—For example, adding the direction parameter to the getStats operation, where the order of the previous parameters is maintained:
  Old API: getStats(deviceName, deviceType, timeframe)
  New API: getStats(deviceName, deviceType, timeframe, direction)

  Existing clients are compatible because they are unaware of the new request data structures.
- Changing cardinality of existing request data structures from mandatory to optional—Existing clients continue using request data structures as if they were mandatory and are compatible.
- Adding new elements to the response message—Fo example, in version 4.2.1, a new element, deviceName, is added to the HttpConnOptRate response. Existing clients continue to retrieve the previous elements, but not the new elements.

Note This type of change could cause a problem for some client code generating tools where strict binding is implemented (such as WSDL2Java). If unexpected subelement exceptions are returned, we recommend that clients either patch the code generating tool to ignore the unexpected elements or use tools that have loose binding with response messages. For more information, see the *Release Note for Cisco Wide Area Application Services*.

The following kinds of monitoring API changes are not backward compatible and existing API clients will cause errors if such API changes are made:

- Removal of an operation—The old API is no longer a proper subset of the new one. Existing clients using the removed operation will be impacted.
- Renaming an operation—This action is equivalent to removing an operation and introducing a new operation.
- Changing cardinality of existing response data structures—Changing the cardinality of fields in the response message, such as changing mandatory fields to optional fields.
- Changing the definition of the data types—Most changes to the data types in the request or response messages are not backward compatible. For example, changing an integer to a double data type will cause an error for existing clients.
- Changing the order of parameters in an operation—Any change to the parameter order will cause an error for existing clients.

The monitoring API will not change in any of these ways that are not backward compatible and instead will define new APIs as needed in new versions, to minimize version compatibility problems.