About the Simple Object Access Protocol Login Event Generator

Revised: March 22, 2012, OL-24194-06

Introduction

This chapter describes the Simple Object Access Protocol (SOAP) Login Event Generator (LEG) software module.

Information About SOAP LEG

The Service Control Management Suite (SCMS) Subscriber Manager (SM) SOAP LEG is a software module that can query an external server via the SOAP to obtain additional information for the subscribers that were logged-in to the SM via various APIs and LEGs. The main purpose of the SOAP LEG is to define the policy of the subscriber based on the input data, the package association configuration, and the query results.

The LEG can query any external server via the SOAP communication protocol if the external server implements an interface defined by the SCMS SM SOAP LEG.

The SCMS SM SOAP LEG supports SOAP 1.1.

The SCMS SM SOAP LEG is an extension of the SM software and runs as part of the SM.

- SOAP Integration Overview, page 28-1
- Common Topologies, page 28-3

SOAP Integration Overview

The SM activates the SOAP LEG to obtain the policy value (or part of the policy value) for the subscribers that are already logged in to the SM.

With the data that the SOAP LEG receives from the SM, it creates a SOAP request, which it issues to the external server to retrieve the policy value. After the external server replies, the SOAP LEG determines the policy value according to the input data, the package association configuration, and the query results. It then initiates a subscriber login to the SM. For more information about the package association, see the “Information about Configuring the Package Association” section on page 29-3.
Query Interface

The SOAP installation package includes a WSDL file. This WSDL file defines the SOAP LEG query to the external server:

```plaintext
QuerySubscriberOut querySubscriber(QuerySubscriberIn subIn)
```

The `QuerySubscriberIn` parameter contains the following data:

- **subscriberId**—Contains the ID of the subscriber.
- **mappings**—Contains the Network IDs of the subscriber.
- **keys/values**—May contain additional data that the external server may need to perform the query.

The Web Server responds to the query and SOAP LEG analyzes the results. The output of the Web Server (QuerySubscriberOut) consists of the following elements:

- **subscriberId**—Contains the ID of the subscriber.
- **mappings**—Contains the Network IDs of the subscriber.
- **keys/values**—May contain additional data that the SOAP LEG may need to determine the package value.
- **propertyKeys/propertyValues**—May contain subscriber properties; for example, packageId or monitor.

Note that **keys** and **values** are used internally by the LEG for the package association procedure and are not passed to the SM when the subscriber is logged in.

Upon receiving a reply from the Web Server, the SOAP LEG adds the query output values to the query input values. Following this, if the SOAP LEG is configured to do so, the LEG uses this data as the input for the package association procedure. See the “Information about Configuring the Package Association” section on page 29-3.

Secure Requests

The SOAP LEG is able to issue a secure request to the external server using the UsernameToken profile as defined in the WS-Security specification. Specifically, it attaches a username and password to every SOAP request it sends. For further information on configuring the username and password, see the “Using the Simple Object Access Protocol Login Event Generator Command-Line Utility” section on page 30-1.

---

**Note**

The SOAP LEG supports only text passwords.
Implementing Query Interface at the Server

Integrating the external server with the SOAP LEG is a two stage process:

1. Compile the provided WSDL file using one of the various tools available. For example, Apache Axis can be used (http://ws.apache.org/axis/). The WSDL file is included in the Cisco Web Services Description Language chapter.

2. Provide the implementation of the \texttt{querySubscriber} function according to the server business logic.

Common Topologies

You can use the SOAP LEG in any SM topology, provided it is possible to supply the LEG with the information it needs to perform the query to the policy server and determine the subscriber policy.

The following figures show the most common topologies.

Figure 28-1 shows the topology with the SM API:

\textbf{Figure 28-1} \hspace{1cm} \textit{SOAP Topology with SM API}

The SM API performs a login operation to the SM (1). The SM identifies that the SOAP LEG needs be activated, and therefore it does not perform a subscriber login at this stage. The SM core passes the information received from the SM API to the SOAP LEG (2). The SOAP LEG queries the SOAP server and identifies the relevant packageId based on the configuration, input parameters, and the query results (3). The SOAP LEG then performs a login operation to the SM (4).
Figure 28-2 shows the topology with the DHCP Sniffer LEG:

The DHCP traffic passes through the SCE (1), which sends a DHCP RDR to the DHCP Sniffer LEG (2). The DHCP Sniffer LEG extracts the relevant information and performs a login operation to the SM (3). The SM identifies that the SOAP LEG needs to be activated, and therefore it does not perform a subscriber login operation at this stage. The SM core passes the information received from the DHCP Sniffer LEG to the SOAP LEG (4). The SOAP LEG queries the SOAP server and identifies the relevant packageId based on all the information received and the query results (5). The SOAP LEG then performs a login operation to the SM (6).
Figure 28-3 shows the topology with the DHCP Lease Query LEG:

**Figure 28-3  SOAP LEG Topology with DHCP Lease Query LEG**

Unknown traffic passes through the SCE (1), which issues a pull request to the SM (2). The SM issues an anonymous-pull-request to the DHCP Lease Query LEG (3). The DHCP Lease Query LEG then queries the DHCP server (4), after which it performs a login operation to the SM (5). The SM identifies that the SOAP LEG needs to be activated, and therefore it does not perform a subscriber login at this stage. The SM Core passes all of the information received from the DHCP Lease Query LEG to the SOAP LEG (6). The SOAP LEG queries the SOAP server and identifies the relevant packageId based on the information received and the query results (7). The SOAP LEG then performs a login operation to the SM (8).
Figure 28-4 shows the topology where SOAP acts as a standalone LEG:

**Figure 28-4  SOAP LEG Standalone Topology**

In this topology, all LEGs, except the SOAP LEG, are disabled. Unknown traffic passes through the SCE (1), which issues a pull request to the Cisco SCMS Subscriber Manager by providing the subscriber IP address. The SM queries (2) the external SOAP server directly, retrieves the subscriber policy details, and then logs in to the SM.

In the SOAP leg configuration (soap_leg.cfg) file:

- If the value of the supported_pull_type parameter is *anonymous*, the SOAP LEG can act as a standalone LEG.
- If the value of this parameter is *introduced*, the SOAP LEG cannot act as a standalone LEG. The default value of the parameter is *introduced*.

**Updating the Package Details from an External SOAP Server**

Information about the package changes in external SOAP server are not sent to the Cisco SCMS Subscriber Manager. This may lead to a mismatch between the package details in the external SOAP server and the Cisco SCMS Subscriber Manager.

In the Cisco SCMS Subscriber Manager, the updateSubscriberDetails (QuerySubscriberIn subscriberIn) web service method is available to synchronize the package details between the external SOAP server and the Cisco SCMS Subscriber Manager.

If you modify any policy in the external SOAP server, run the updateSubscriberDetails webservice method to send the package details to the Cisco SCMS Subscriber Manager.
**updateSubscriberDetails (QuerySubscriberIn subscriberIn) Webservice Method**

**Syntax**

```java
public void updateSubscriberDetails(com.pcube.management.soapserver.generated.QuerySubscriberIn in0) throws java.rmi.RemoteException;
```

**Description**

This method updates the details of the subscriber in the Subscriber Manager database.

**Parameters**

querySubscriberIn contains the subscriber details.

The QuerySubscriberIn parameter contains the following data:

- subscriberId—Contains the ID of the subscriber
- mappings—Contains the Network IDs of the subscriber
- keys/values—Contains subscriber properties; for example, packageId or monitor.

These values are updated in SM database.

**Error Codes**

None.

**Return Value**

None

---

**updateSubscriberDetails (QuerySubscriberIn subscriberIn) Webservice Method—Program Example**

```java
package com.pcube.management.soapserver.generated;
import java.net.MalformedURLException;
import java.rmi.RemoteException;
import javax.xml.rpc.ServiceException;
public class SoapClientSampleProgram {
    public static void main(String[] args) {
        UpdateSubscriberInterfaceService service = new
        UpdateSubscriberInterfaceServiceLocator();

        String subsIP = "2.1.1.1";
        try {
            String smIp = "10.34.41.100";
            java.net.URL endpoint = new
            java.net.URL("http://" + smIp + ":8090/axis/services/updateSubscriber?wsdl");
            UpdateSubscriberInterface
            updateSubscriberInterfaceObj = service.getupdateSubscriber(endpoint);

            QuerySubscriberIn subsIn = new
            QuerySubscriberIn();
            subsIn.setSubscriberId("s1234");
            subsIn.setMappings(new String[]{subsIP});
            subsIn.setKeys(new
            String[]{"packageId"});
```

---

**Cisco SCMS Subscriber Manager LEGs User Guide**

OL-24194-06

28-7
subsIn.setValues(new String[]{"124"});

updateSubscriberInterfaceObj.updateSubscriberDetails(subsIn);

) catch (ServiceException e) {
    e.printStackTrace();
) catch (RemoteException e) {
    e.printStackTrace();
) catch (MalformedURLException e) {
    e.printStackTrace();
}
Troubleshooting the SOAP LEG

Scenario
The client opens multiple ports for updating the packageID and faces a connection timeout exception or BindException.

Symptom
The client gets an AxisFault (Connection timeout exception or BindException) when the number of ports crosses the limit supported by the client machine.

Workaround
Configure the number of ephemeral ports based on the number of connections the client should open.

Use the `netstat -an | grep 8090 | wc -l` command to find the number of port connections being used by the client.

On Linux, use the `cat /proc/sys/net/ipv4/ip_local_port_range` command to view the ephemeral port limit configured.

The value can be changed as follows:

```
echo "1024 65535" > /proc/sys/net/ipv4/ip_local_port_range
```

On Solaris, use the `/usr/sbin/ndd /dev/tcp tcp_smallest_anon_port tcp_largest_anon_port` command to view the ephemeral port limit configured.

To increase the number of ports on an Apache server, see the corresponding Apache documentation pertaining to the topics related to AxisFault.