Sample SMI-S Java Client

This chapter describes the Sample SMI-S java client developed using JSR 48 specifications. JSR 48 is a set of java WBEM Service APIs and reference implementation for WBEM. WBEM is an initiative from the DMTF that unifies systems management and instrumentation.

The SMI-S java client uses the client and listener packages provided by the JSR 48 specifications, which consists of classes and interfaces for developing WBEM Clients and WBEM Event Listeners. The WBEMClient interface in the client package is used to invoke WBEM operations against a WBEM Server. A WBEMClient implementation can be retrieved from the WBEMClientFactory by specifying the protocol to be used. IndicationListener is implemented by the code that wants to create a listener for indications. The WBEMListener interface is used to add or remove WBEM Indication Listeners.

The SMI-S java client uses the Cisco DCNM SMI-S external agent for managing and monitoring Cisco DCNM for SAN.

You can perform the following tasks from the services provided by the SMI-S client tool:

- List Zonesets, Zones, and Zonemembers in VSAN
  - Obtain VSANs using enumerateInstanceNames API in WBEMClient.
  - Obtain Zonesets, Zones, and Zonemembers using associatorInstances API in WBEMClient on VSAN.
- List Zones in Zoneset
  - Get VSANs using enumerateInstanceNames API in WBEMClient.
  - Get zonesets in selected VSAN using associatorInstances.
  - Get zones in selected zoneset.
- List Zonemembers in Zone
  - Get VSANs using enumerateInstanceNames API in WBEMClient.
  - Get Zones in selected VSAN using associatorInstances.
  - Get Zonemembers in selected zone.
- Create new ZoneSet and Zones in fabric
  - Get VSANs using enumerateInstanceNames API in WBEMClient.
  - Get Zoneservice of the selected VSAN using associatorInstances.
  - Invoke the method createzoneset in Zoneservice class with zoneset and zone name.
- Add existing Zone to existing Zoneset in fabric
  - Get VSANs using enumerateInstanceNames API in WBEMClient.

  - Get Zoneservice of the selected VSAN using associatorInstances.
  - Invoke the method addzone in Zoneservice class with existing zoneset and zone information.
- Add Zonemember to existing Zone in fabric
  - Get VSANs using enumerateInstanceNames API in WBEMClient.
  - Get Zoneservice of the selected VSAN using associatorInstances.
  - Invoke the method CreateZoneMembershipSettingData in Zoneservice class with existing zone and new zonemember information.
- Activate and Deactivate a Zoneset in fabric
  - Get VSANs using enumerateInstanceNames API in WBEMClient.
  - Get Zoneservice of the selected VSAN using associatorInstances.
  - Invoke the method ActivateZoneSet in Zoneservice class with existing zoneset for activation or deactivating the zoneset.

Installing Sample SMI-S Client

Note Cisco SMI-S Client is packaged with DCNM SMI-S Server.

When you choose Cisco DCNM with SMI-S Sever during installation, SMI-S Client is also available. For more information, see the "Installing Cisco DCNM SMI-S Server" section on page 3-1.

The following are the SMI-S Client installation locations:"Installing Cisco DCNM SMI-S Server" section on page 3-1

- On Microsoft Windows, by default, Cisco DCNM is installed at C:\Program Files\Cisco Systems.
- On a UNIX (Solaris or Linux) machine, Cisco DCNM is installed at /usr/local/cisco/dcm or $HOME/dcm.
- The SMI-S Client is available at <*DCNM install dir*>dcm/smis/client.
- The batch files to compile and execute SMI-S Client and SMI-S Indication Client are available at <*DCNM install dir*>*/dcm/smis/client/bin*.
- The wbem jars are available at <*DCNM install dir*>/SmisClient/lib.
- The source files of SMI-S Client and SMI-S Indication Client are available at <*DCNM install dir*>/dcm/smis/client/src and the corresponding class files are available at <*DCNM install dir*>/dcm/smis/client/build.
- The configuration file used to retrieve Cisco DCNM credentials to access DCNM SMI-S agent are available at <*DCNM install dir*>/SmisClient/conf.
- You can choose the type of indication to be subscribed in the *filter.properties config* file.

Services Provided by SMI-S Java Client

The following NMS services are supported:

- List Switches
- List Hosts

- List Targets
- List VSANs in fabric
- List Switches in VSAN
- Enumerate the Instance of Any CIM class
- Enumerate the names of Any CIM class
- Execute Zone Operations

Table 5-1 provides the details of the SMI-S java client services.

Table 5-1 Services Provided by SMI-S Java Client

Services Name	Description
List Switches	This service lists the VSANs present in the fabric. You need to select the VSAN to see the list of switches. The tool uses the enumerateInstances API in Interface WBEMClient.
List Hosts	This service lists the hosts present in the fabric and also lists the endports present in the host. The tool uses the enumerateInstances API in Interface WBEMClient and an associatorInstances to get the host associated to the end ports.
List Target	This service lists the targets present in the fabric and also lists the end ports present in the target. The tool uses the enumerateInstances API in Interface WBEMClient and an associatorInstances to get the target associated to the end ports.
List VSANs in Fabric	This service lists all of the VSANs present in the fabric and their properties. For example, the status, WWN, VSAN ID, and temporary name of the VSAN. The tool uses the enumerateInstances API in the Interface WBEMClient.
List Switches in the VSAN	This service lists all the VSANs that are up and active in the fabric. Once you select it, the logical computer systems (logical switches) present in that VSAN are displayed. It also gives information of the physical switch each logical switch is hosted on.
Enumerate the Names of any CIM Class	This option allows you to select a CIM class of your choice to view CIMObjectPath names for a specified CIM class.
Enumerate the Instance of any CIM Class	This option allows you to select a CIM class of your choice to view instances of a specified CIM class.
Execute Zone Operations	This option lists the bunch of zone services supported by the Cisco SMI-S client tool.
Subscription of the Event Listener	SMI-S Indication client can be used to subscribe to events.

Examples of Developing SMI-S Client Using WBEM Solutions

The following example shows how to obtain WBEMclient with the specified protocol from WBEMClientFactory:

```
/*
            * WBEMClient handle
            */
            WBEMClient clientObj = null;

            /*
            * The host to connect to. In the form of a WBEM URL. Make sure the WBEM
            * Server is running before trying this example.
            */

            String host = "http://localhost:5988/cimv2";


            try {
                  /*
                  * Create an object path using the host variable.
                  */
                  CIMObjectPath cns = new CIMObjectPath(host);

                  /*
                  * Create the principal - used for authentication/authorization
                  */
                  UserPrincipal up = new UserPrincipal(username);

                  /*
                  * Create the credential - used for authentication/authorization
                  */
                  PasswordCredential pc = new PasswordCredential(password);

                  /*
                  * Add the principal and credential to the subject.
                  */
                  Subject s = new Subject();
                  s.getPrincipals().add(up);
                  s.getPrivateCredentials().add(pc);

                  /*
                  * Create a CIM client connection using the either CIM-XML or
                  * WS-Management protocol
                  */
                  clientObj = WBEMClientFactory.getClient("CIM-XML");
                  clientObj.initialize(cns, s, null);
            } catch (WBEMException e)
```

The following example shows using the enumerateInstanceNames API in WBEMClient:

```
// get wbem client obj with specified protocol
    WBEMClient wbclient = WBEMClientFactory.getClient("CIM-XML");

// pass the CIM class & name space

CIMObjectPath cop = new CIMObjectPath(className, namespace);
CloseableIterator<CIMObjectPath> ei = wbclient.enumerateInstanceNames(cop);
```

The following example shows using the enumerateInstances API in WBEMClient:

```
// get wbem client obj with specified protocol
```

```
    WBEMClient wbclient = WBEMClientFactory.getClient("CIM-XML");

// pass the CIM class
CloseableIterator<CIMInstance> ei = wbclient. enumerateInstances(className,null);


}
```

The following example shows how to get association names:

```
// get wbem client obj with specified protocol
WBEMClient wbclient = WBEMClientFactory.getClient("CIM-XML");

CloseableIterator<CIMObjectPath> an =null;
    try{
            an = wbclient.associatorNames(cimop,null, resultClass, null, null);
        } catch (WBEMException e) {
            System.out.println(e);
        }
```

The following example shows how to get assoication Instances:

```
// get wbem client obj with specified protocol
WBEMClient wbclient = WBEMClientFactory.getClient("CIM-XML");
CloseableIterator<CIMInstance> a =null;
try{
a = wbclient.associatorInstances(cimop,null, resultClass, null, null, false, null);
    } catch (WBEMException e) {
        System.out.println(e);
    }
```

The following example shows how to use the invoke method and use the extrinsic methods of CIM Class:

```
.// get wbem client obj with specified protocol
WBEMClient wbclient = WBEMClientFactory.getClient("CIM-XML");

UnsignedInteger32 uInt32=(UnsignedInteger32) wbclient.invokeMethod(CIM_classname,
methodname,inArgsAlias,outArgs);

Sample code for registering listener for indications:
class Listener implements IndicationListener {
        ……
        public void indicationOccured(String pIndicationURL, CIMInstance pIndication) {
            CIMInstance indicationInstance = pIndication;
            System.out.println("Received indication instance: " + indicationInstance);
        }
…}
```

The following example shows how to register listener indications:

```
//get the WBEMListener from the WBEMListenerFactory
WBEMListener listnr = WBEMListenerFactory.getListener("CIM-XML");
//add the listener to the port. use 0 to specify any available port
Int port = api.addListener(cl,0, "http");
// get wbem client obj with specified protocol
WBEMClient wbclient = WBEMClientFactory.getClient("CIM-XML");
//create the filter
```

```
String filter  ="SELECT SELECT * from CIM_InstIndication WHERE sourceInstance ISA
CISCO_PhysicalComputerSystem";
CIMObjectPath op = new CIMObjectPath("CIM_IndicationFilter", namespace);
        CIMClass filterClass = client.getClass(op, false, true, false, null);
        CIMInstance filterInstance = filterClass.newInstance();
        CIMProperty<?>[] fcpArray = {
        new CIMProperty<String>("Query", CIMDataType.STRING_T, filter, false, false,
null),
        new CIMProperty<String>("QueryLanguage", CIMDataType.STRING_T, "WQL", false,
false, null) };
        filterInstance = filterInstance.deriveInstance(fcpArray);
        CIMObjectPath filterOP = client.createInstance(filterInstance);

//create the listener
CIMObjectPath op = new CIMObjectPath("CIM_ListenerDestinationCIMXML", namespace);
        CIMProperty<?>[] cpa = { new CIMProperty<String>("Destination",
                CIMDataType.STRING_T, "http://" + systemName + ":"+ port +"/", false,
false, null) };

        // create new instance of listener
        CIMInstance listenerInstance = new CIMInstance(op, cpa);
        CIMObjectPath listenerOP = client.createInstance(listenerInstance);
//create a subscription, an association between the filter and listener.
CIMProperty<?>[] sicpArray = {
                            new CIMProperty<CIMObjectPath>("Filter", new CIMDataType(
                                    filterOP.getObjectName()), filterOP, true, false, null),
                            new CIMProperty<CIMObjectPath>("Handler", new CIMDataType(
                                    listenerOP.getObjectName()), listenerOP, true, false,
null) };
                    CIMInstance subscriptionInstance = new CIMInstance(
                            new CIMObjectPath("CIM_IndicationSubscription", namespace,
sicpArray), sicpArray);
                    try {
                        CIMObjectPath subscriptionOP =
client.createInstance(subscriptionInstance);

                    } catch (WBEMException e) {
                        throw e;
                    }
```