CHAPTER 30

Using Object Grouping Services

The CWCS Object Grouping Service (OGS) provides a generic means for creating, managing and sharing groups of objects, regardless of type.

The following topics describe the OGS and how to use it in your applications:

For more information about OGS, refer to the following resources:

- *OGS System Functional Specification*, ENG-101932
- *OGS GUI Client Functional Specification*, EDCS-120040
- *OGS GUI Client Software Unit Design Specification*, EDCS-152942
- *Triveni OGS ASA Specification,* EDCS-161116
- *Secure Views and Common Groups In Object Grouping Service Functional Specification*, EDCS-298617
- *Shared Groups in OGS*, EDCS-341632
- *Functional Specifications related to Client side changes in OGS 1.3*, EDCS-347045
- *OGS 1.3 Client related Software Design Specification,* EDCS-358505
- *Kilner Virtual ASA Functional Specification*, ENG-203936
- *Object Selector User Guide,* EDCS-158538
- *SDK Developer's Guide for UII, Release 6.1*, EDCS-275335

Understanding OGS

OGS allows applications to create, manage, and share persistent groups of objects.

OGS is a generic grouping service. It does not supply you with predefined groups. Instead, it provides tools that allow you to define groups useful to your application. Once you have defined the groups you want, you can supply them in predefined form with your application.

OGS places no limits on the types of objects you can group. Most developers use OGS to group network devices. However, you can also use it to manage groups of scheduled jobs, policies, users, tasks, VLANs, subnets, IP phones, user interface views, fault conditions, or any other kind of object that can be grouped based on shared attributes.

The following topics explain the basics of OGS:

About the OGS Components

A complete OGS system includes the following components

- OGS Server: Manages groups of objects to be shared by applications. For details about implementing this component, see the "Implementing OGS Servers" section on page 30-7.
- Application Service Adapters (ASAs): Application-specific processes that serve as sources of the Objects that are grouped by the OGS Server. For details on creating ASAs for your applications, see the"Creating OGS ASAs" section on page 30-17.
- OGS Clients: Applications that use the OGS Server to create and manipulate groups in order to perform an application function. You can display OGS data and coordinate OGS interactions using the Object Selector in the GUI for your client application. For details, see the "Creating an OGS GUI" section on page 30-40.

Basic OGS Concepts

OGS makes use of several concepts whose definitions differ from the commonly used definitions:

- **OGS Class**:

  An OGS class describes the representation of a set of application abstractions. This differs from traditional object-oriented systems, where a class defines a set of attributes and the operations that can be performed on instances of that class.

An OGS class is concerned only with the representation and retrieval of attributes. Class hierarchies are supported and are interpreted in the conventional sense; that is, a subclass inherits all the attributes of its superclasses. An instance of an OGS class can be used in any context where an instance of one of its superclasses is expected.

- **OGS Object**:

  An OGS object is a collection of attribute values. The OGS class of an object determines the set of attributes associated with that object. Associated with every object is a unique and immutable object ID (OID). When an ASA evaluates a rule, the data returned to the OGS Server is a collection of proxy objects that contains:

  - The OIDs of the grouped objects.
  - Any attributes of those objects that were requested.

  These proxy objects are referred to as OGS objects in the rest of this document.

- **OGS Group**:

  An OGS group is a named aggregate entity comprising a set of objects belonging to a single class or a set of classes with a common superclass. Groups can be shared between users or applications, subject to access-control restrictions. The membership of a group is determined by a rule.

- **OGS Rule**:

  An OGS rule consists of one or more rule expressions combined by operators, which can be AND, OR or EXCLUDE. A rule always evaluates to objects of a particular class defined in an application schema.

About OGS Groups

A group in OGS is a named aggregate entity comprised of a set of objects. Each group has:

- A set of properties, such as group ID, name, description, permission, etc.
- An associated rule. The rule determines the members of a group, which may change whenever the rule is evaluated.

OGS Servers represent the group membership as a list of object IDs. Besides the OID list, attributes associated with the objects in a group may also be queried. The attributes of a class and the hierarchical relationship of classes are defined in the Application Service Adaptor schema, which must be registered with the OGS.

Each OGS group has a set of properties, including a unique name. Table 30-1 describes these properties.

Table 30-1 OGS Group Properties

Property	Description
GroupID	A unique identifier for the group.
GroupName	Fully qualified name of the group. This name must include the parent's path, with each element of the path being separated by the / character.
Description	User-specified text description of the group.
Created By	ID of the user who created the group.
Created Time	Date and time the group was created.
Last Modified By	ID of the user who last made changes to the group.
Last Modified Time	Date and time of last modification.

Table 30-1 OGS Group Properties

Property	Description
Last Evaluated By	ID of the user who last evaluated the group.
Last Evaluated Time	Date and time of last group evaluation.
Type	Static or Dynamic.
Access Control	Permission list to read/write this group.
Rule	Rule to determine memberships of the group.
Tags	Variable list of name-value pairs defined by the user.

About OGS Group Types

OGS groups can be either dynamic or static (see also the "About OGS Container Groups" section on page 30-5).

A dynamic group is one whose membership list is effectively computed every time a user views its members. For each request to view the group, the OGS Server may automatically request the relevant Application Service Adapter to recompute the group's rule. The Server is not required to store a dynamic group's membership list (although it may cache the results of the last re-evaluation), and viewing a dynamic group will always give the latest group membership.

Example

A user creates the dynamic group "MyDevices", which has the rule "IPAddress in Subnet 172.20.32.0/24". At creation time, the rule evaluates to devices D1, D2, and D3. If a user attempts to view the group membership at any later time, the OGS Server will return the current membership.

If new devices D4 and D5 joined subnetwork 172.20.32.0/24 in between the time the group was created and the time its membership was queried, OGS will return devices D1, D2, D3, D4 and D5 as the current members of the group.

A static group is one whose membership is refreshed only when a user explicitly requests it. Between re-evaluations, the OGS Server stores the static group's membership list and group definition. Whenever a user views a static group, OGS returns the membership list the ASA created the last time the group rule was evaluated.

Note that the OGS may cache the membership and maintain the cache by other means. Group definitions are always stored, regardless of whether the group is static or dynamic.

Example

A user creates the static group "MyDevices", which has the rule "IPAddress in Subnet 172.20.32.0/24". The rule evaluates to devices D1, D2, and D3, and the OIDs for these devices is stored along with the group definition.

If a user attempts to view the group, he will see only devices D1, D2 and D3, as these were the devices that satisfied the group rule when the group was created.

If the user wants to refresh the membership of the group, then he must explicitly request it. The OGS Server will then ask the ASA to reevaluate the rule.

If new devices D4 and D5 joined subnetwork 172.20.32.0/24 in between the time the group was created and the time the user requested the refresh, OGS will return devices D1, D2, D3, D4 and D5 as the current members of the group. This device list will also be stored as the membership of "MyDevices" until the next refresh.

Most OGS groups are dynamic. Static groups are useful when you want users to be able to “snapshot” a group’s membership and maintain it until a later time, when the user can quickly update it without having to change the group definition.

Example

A network administrator in a large enterprise is responsible for the management of all operational Catalyst 6K series switches. Switches in this enterprise must go through several configuration stages before they are considered operational. Configuration is done by a department to which the Network Administrator does not belong. Before becoming operational, the switches may also be available on the network.

To cope with this, the Network Administrator could create a static group called "My Cat6K Devices" with the rule "DeviceType == Catalyst 6K". This would allow him to accomplish the following:

- The group "My Cat6k Devices" would be populated with the operational Catalyst 6K devices in the network at the time.
- The enterprise acquires new Catalyst 6K devices, deploys them in the network, and begins configuring them. The administrator continues to use the group "My Cat6k Devices" with the assurance that the new Catalyst 6K devices will be excluded from the group.
- When the new switches are operational, the administrator requests a refresh of the group without making changes to the group rule. “MyCat6K Devices” now includes the new switches.

About OGS Container Groups

Container groups are a separate type of group, on par with normal groups. This is because they are groups who have no membership of their own. A container group is an “empty” container, whose membership is simply the union of the membership of all its subgroups.

This is different from normal groups that must have a rule to determine its membership. Instead, a container group’s effective rule consists of:

- If it has no subgroups: No rule.
- If it has only one subgroup: The same effective rule as its subgroup.
- If it has more than one subgroup: A rule composed of the effective rules of all its subgroups, combined with the operator OR.

Container groups are can be static or dynamic depending on the behavior of their subgroups:

- Dynamic Container: Its membership is the aggregate membership of its subgroups at any given time.
- Static Container: Its membership (as the aggregate of its subgroup memberships) is recomputed only when new subgroups are added, existing subgroups are deleted, when the subgroups’ rules are modified, or upon request.

About OGS Group Hierarchy

OGS manages groups in a hierarchical fashion and supports subgrouping. Each child group is a subgroup of a parent group and its group membership will be a subset of its parent group. OGS also supports container groups. Container groups are groups with no rule, whose membership is simply the union of the membership of its children.

Regardless of whether a group's parent is static or dynamic, the result of evaluating a group is the intersection of the objects that satisfy its rule and the objects that satisfy its parent's effective rule.

The effective rule for a static group is a rule that enumerates its members. When a static group is reevaluated, all its descendant static groups will also be reevaluated.

The effective rule for a dynamic group is an expression that is formed by applying the operator AND to the operands that are the group's rule and its parent's effective rule.

How Rules Are Constructed

A group rule consists of one or more expressions, which can be combined using the operators AND, OR or NOT. Each expression has the following components, concatenated with a ":"separator

- Domain: The cluster of applications sharing the group (e.g., "Triveni" or "ALL").
- Application: The specific application within the domain that shares the group (e.g., "RME" or "Campus")
- Class: The class of object (e.g., "Device").
- Attribute: The specific object attribute whose value the rule will test (e.g., "DeviceType" or "IPAddress"),
- Operator: The evaluation operation defined in the Application Schema (e.g., "IsEqualTo" or "Contains").
- Value: The value to be tested for (e.g., "Router").

Example

The following rule will select Devices which are either Routers or have an IOS Version greater than 11.3 and that are assigned IP Addresses beginning with the octets "172.20":

```
MYAPP:RME:Device.DeviceTypeEquals "Router" or MYAPP:RME:Device.IOSVersion > "11.3" AND
MYAPP:RME:Device.IPAddressContains "172.20"
```

Choosing to Implement OGS

If you want to implement OGS in your application without extensive customization, you must also:

- Install the Common Services Transport Mechanism (CSTM). The OGS Server uses CSTM to communicate with its clients. For more information on CSTM, see Chapter 31, "Using the Common Services Transport Mechanism."
- Implement a local database. The OGS Server persists group information to a local data store to ensure that group information is not lost between activations. Your choice of DBMS to accomplish this should adhere to any guideline or prescription established for your application. For information on implementing a database in Common Services, see Chapter 11, "Using the Database APIs."
- Implement Event Services Software (ESS), including the Java Messaging Service (JMS). For more information on these topics, see Chapter 19, "Using Event Services Software."
- Implement the OGS Server. For more information on this task, see the "Implementing OGS Servers" section on page 30-7.
- Create an OGS Application Service Adapter and Application Schema File for your application. For more information on this task, see the "Creating OGS ASAs" section on page 30-17.

In addition to these requirements, your application must support JDK 1.3.1.

Implementing OGS Servers

The OGS Server performs the following tasks:

- Creating and maintaining group information.
- Interacting with Application Service Adapters (ASAs) to:
  - Evaluate group membership.
  - Retrieve the requested attributes of the member objects of a group.
  - A later release will allow ASAs to register their schema. Currently, you must register ASA schema statically, via configuration files.

The OGS Server provides its clients with a unified and consistent view of group definitions and results of rule evaluation. It does this by recording any change to the group information (through creation, modification, deletion or evaluation of a group).

The OGS Server relies on:

- A means of persisting the group definition and membership data. This can be a relational database, object-oriented database, or file system.
- The Common Services Transport Mechanism (CSTM) to interact with clients.
- ASAs to perform rule evaluation.

The following topics discuss OGS Server and its implementation in detail:

Getting Started with OGS Server

Setting up a running OGS Server instance normally requires you to create an Application Service Adaptor (ASA) for your application and then set the OGS Server to use it.

You can also create a very simple OGS Server implementation using the test ASA supplied on the SDK. The following steps explain how to install OGS Server and configure it to use TestASA through a script file.

For details on writing an ASA that performs work useful to your application, see "Creating OGS ASAs" section on page 30-17.

Step 1 Retrieve the packaged version of the OGS WAR file from the OGS Portal.

Step 2 Extract the WAR file's contents to the default application directories under Tomcat. For example:

- Extract all OGS jar files to /tomcat/webapps/*appname*/WEB_INF/lib (where *appname* is the name of your application).
- Extract all other files (in including property and configuration files) to /tomcat/webapps/*appname*/WEB_INF/classes.

Step 3 Retrieve TestASA.tar (or TestASA.zip) from the SDK portal.

Step 4 Extract TestASA.tar (or TestASA.zip) to the default application directories under Tomcat.

Step 5 Run the OGS Server test setup script ServerSetup.sh (ServerSetup.cmd). The script will update OGSServer.properties and ASARegistry files to point to TestASA, exactly as you would do when implementing your own custom ASA (see "Running a Customized ASA" section on page 30-36).

Step 6 Start CWCS and OGS Server. Use the Administrative GUI to add some groups.

How OGS Server Works

OGS Server makes use of five classes to instantiate its most important behaviors:

- OGSClassDefinition:

  This class models the OGS notion of classes (see the "Basic OGS Concepts" section on page 30-2). The OGS Server creates instances of OGSClassDefinition using the definitions of OGS classes in an application schema. An OGSClass instance holds information about an application class. such as its name, its domain and the application to which it belongs, the attributes associated with it, attributes that can be used in composing rules, the operations that are permitted on those attributes, and the allowable target values.

- OGSObject:

  This class models the OGS notion of objects (see the "Basic OGS Concepts" section on page 30-2). ASAs create instances of OGSObject and return lists of them to the OGS Server whenever the ASAs evaluate rules. Each OGSObject instance contains the OID of the application class object it represents, and the values of the other attributes that were requested by a client.

- OGSRule, OGSRuleExpression:

  These two classes model the expressions used in rules and the rules themselves.

- OGSGroupDefinition:

  This class models OGS groups in a straightforward manner. It has instance variables that store group information, such as group ID, name, ownership, creation and modification times, and so on.

An active OGS Server instance is an executing operating system process that runs as a daemon. Developers should use the CWCS Daemon Manager (see Chapter 17, "Using the Daemon Manager") to manage OGS Server states.

Upon activation, an OGS Server process performs initialization tasks, such as reading static ASA registration and group definitions from its persistent store.

If initialization is completed successfully, OGS Server publishes its URN to CSTM and waits for requests. When it receives requests, it passes them to the appropriate registered ASA, and returns the result.

Using the OGS Server APIs

The OGS Server API allows you to:

- Create, delete, copy, modify and rename groups
- Copy group hierarchies
- Evaluate the members of a group

- Retrieve application schema and class information
- Retrieve group definitions
- Retrieve lists of all group names
- Verify service

Usage, input arguments and other details for each OGSServer API call are fully documented in the OGS Javadoc available on the OGS Portal at https://mco.cisco.com/ubiapps/portal/go.jsp?portal_id=6625.

The following example demonstrates how to implement the API as part of a client application.

Example 30-1 A Sample OGS Client

```
package com.cisco.nm.xms.ogs.client;

import java.util.ArrayList;
import com.cisco.nm.xms.ogs.server.OGSInterface;
import com.cisco.nm.xms.ogs.util.OGSOid;
import com.cisco.nm.xms.ogs.util.OGSSecurityContext;
import com.cisco.nm.xms.ogs.util.OGSObjectList;
import com.cisco.nm.xms.ogs.util.OGSObject;
import com.cisco.nm.xms.ogs.util.ClassPath;
import com.cisco.nm.xms.ogs.client.OGSServerProxy;

public class GetDeviceGroups {

    // This program retrieves all groups managed by an OGS Server
    // that have at least one device object as a member. The program
    // is invoked with a single argument that names the class that
    // represents device objects.

    public static void main(String[] args)
    {
        if (args.length != 1) {
            System.out.println("usage: GetDeviceGroups device-class-name");
            System.exit(-1);
        }
        _devClassName = args[0];
        _devClassLength = _devClassName.length();
        try {
            _devClassElements = new ClassPath(args[0]).elements();
        } catch (Exception ex0) {
            System.err.println("Error getting elements of class: " + args[0]
              + ": " + ex0.getMessage());
        }
        TaskID task=new TaskID ("OGS", "OGSOPERATION", null); OGSSecurityContext ctxt=new
OGSSecurityContext(_adminName,task);
        try {
            // Get the hierarchical membership of the root of all
            // groups. Having retrieved the membership in this
            // fashion, we can now examine the membership of
            // all the descendant groups and determine the ones
            // whose membership contains at least one device object.

            OGSServerProxy pxy = new OGSServerProxy();
            Integer mtype = new Integer(OGSInterface.HIERARCHICAL_MEMBERSHIP);
            OGSObjectList list
              = pxy.evaluateGroup(ctxt,
                                  null,
                                  null,
```

```
                                _root,
                                null,
                                mtype,
                                Boolean.FALSE);

            String[] devices = getDeviceGroups(list);
            for (int i = 0; i < devices.length; ++i) {
                System.out.println(devices[i]);
            }
        } catch (Exception ex) {
            System.err.println("Error getting device groups: " +
              ex.getMessage());
        }
    }

    private static String[] getDeviceGroups(OGSObjectList list)
    {
        ArrayList result = new ArrayList();
        OGSObjectList[] children = list.children();
        for (int i = 0; i < children.length; ++i) {
            getDeviceGroups(list, result);
        }
        return (String[])result.toArray(new String[0]);
    }

    // This method gathers the names of all the groups
    // (in the hierarchy rooted at the group represented by 'list')
    // have at least one device object as a member.

    private static boolean getDeviceGroups(OGSObjectList list, ArrayList result)
    {
        boolean hasDevice = false;
        OGSObjectList[] children = list.children();
        for (int j = 0; j < children.length; ++j) {
            if (getDeviceGroups(children[j], result)) {
                hasDevice = true;
            }
        }
        // If any descendant has a device member, this group
        // has it too.

        if (hasDevice) {
            result.add(list.name());
            return true;
        }

        // This group has no descendants that have a device
        // member, so check the membership of the group for
        // devices.

        OGSObject[] objects = list.objects();
        for (int i = 0; i < objects.length; ++i) {
            String oid = objects[i].objectId();
            try {
                String objClass = OGSOid.getClassFromOid(oid);
                if (isDeviceClass(objClass)) {
                    result.add(list.name());
                    return true;
                }
            } catch (Exception ex) {
                System.err.println("Error processing object: " + oid);
            }
        }
        return false;
```

```
    }

    // Returns true if the class named by 'cname' is the same
    // as the device class or is a subclass.

    private static boolean isDeviceClass(String cname)
    {
        if (cname.length() < _devClassLength) {
            return false;
        } else if (cname.equals(_devClassName)) {
            return true;
        }

        // The following can also be achieved by verifying that
        // _devClassName is a prefix of cname and that the character
        // following the matching prefix is ':'.

        try {
          ArrayList classElements = new ClassPath(cname).elements();
          if (classElements.size() < _devClassElements.size()) {
            return false;
          }
          for (int i = 0; i < _devClassElements.size(); ++i) {
            String s1 = (String)classElements.get(i);
            String s2 = (String)_devClassElements.get(i);
            if (!s1.equals(s2)) {
                return false;
            }
          }
          return true;
        } catch (Exception ex) {
            System.err.println("Error getting elements of: " + cname);
            return false;
        }
    }

    private static int _devClassLength;
    private static String _devClassName;
    private static ArrayList _devClassElements;
    private static String _root = "/";
    private static String _adminName = "admin";
}
```

Note This code is in the OGS VOB and can be found at: /vob/enm_ogs/share/classes/client/com/cisco/nm/xms/ogs/client/OGSClient.java

Customizing OGS Server Interfaces

All OGS Server interfaces that you can customize are listed as settable parameters in the OGSServer.properties file. These customizable interfaces include:

- **CacheImplClass**

  This is an implementation of the interface OGSGroupCacheIf (see Figure 30-1). OGS uses an instance of this class as a source of grouping data. Though an ASA ultimately evaluates the rule, this abstraction is defied so that group membership can be cached. The OGS Server uses an instance of

the class specified for this property for every ASA that is registered. A limitation in this release of OGS is that instances of only a single class can be used for caching, even when multiple ASAs are used (that is, the caching strategy cannot be customized for the individual ASAs). The default value is com.cisco.nm.xms.ogs.server.GroupCacheImpl. Use the default implementation if possible, or subclass GroupCacheImpl (as Kilner does).

- **GroupPersistorImplClass**

  An implementation of the interface OGSGroupPersistorIf (see Figure 30-1). The OGS Server uses this class to persist and retrieve group definitions. The default value for this property is com.cisco.nm.xms.ogs.server.GroupPersistorImpl. This implementation uses the CWCS database to persist group definitions. Use the default implementation unless your product has special requirements.

- **GroupCachePersistorImplClass**

  This is an implementation of the interface GroupCachePersistorIf (seeFigure 30-2 and the caching information in this topic). Instances of GroupCacheImpl use an instance of this class to persist cached membership data, so this property is relevant if the OGS is configured to use class GroupCacheImpl (or its subclass) only.

  The default value for this property is com.cisco.nm.xms.ogs.server.GroupCachePersistorImpl. Use the default implementation unless your product has special requirements.

- **SecurityImplClass**

  This is an implementation of OGSSecurityHandlerIf (see Figure 30-1). OGS uses an instance of this class to enforce access control on groups, but not on the group membership.

  The default value for this property is com.cisco.nm.xms.ogs.server.DummySecurityHandler, which allows unrestricted access by all users. You can find examples of implementations that follow different security requirements (those for Campus Manager and Kilner) in the OGS VOB.

- **ASARegistrationFile**

  This specifies the XML file used to register ASAs with the OGS Server and identifies the ASA implementation to be used. See the "Registering the ASA with OGS" section on page 30-36 for an example ASA Registration File and its DTD.

- **SystemGroupsFile**

  The file that contains the definitions of any predefined groups you have customized for your product and that you will ship with OGS. For a sample version of this, see Kilner's System Groups file, vasa-system-groups.xml.

- **SystemGroupCreator**

  This specifies the name of the creator of the system predefined groups given in the SystemGroupsFile. This can be any string, but your security module may place some restrictions on what this should be. The default value is "ogs".

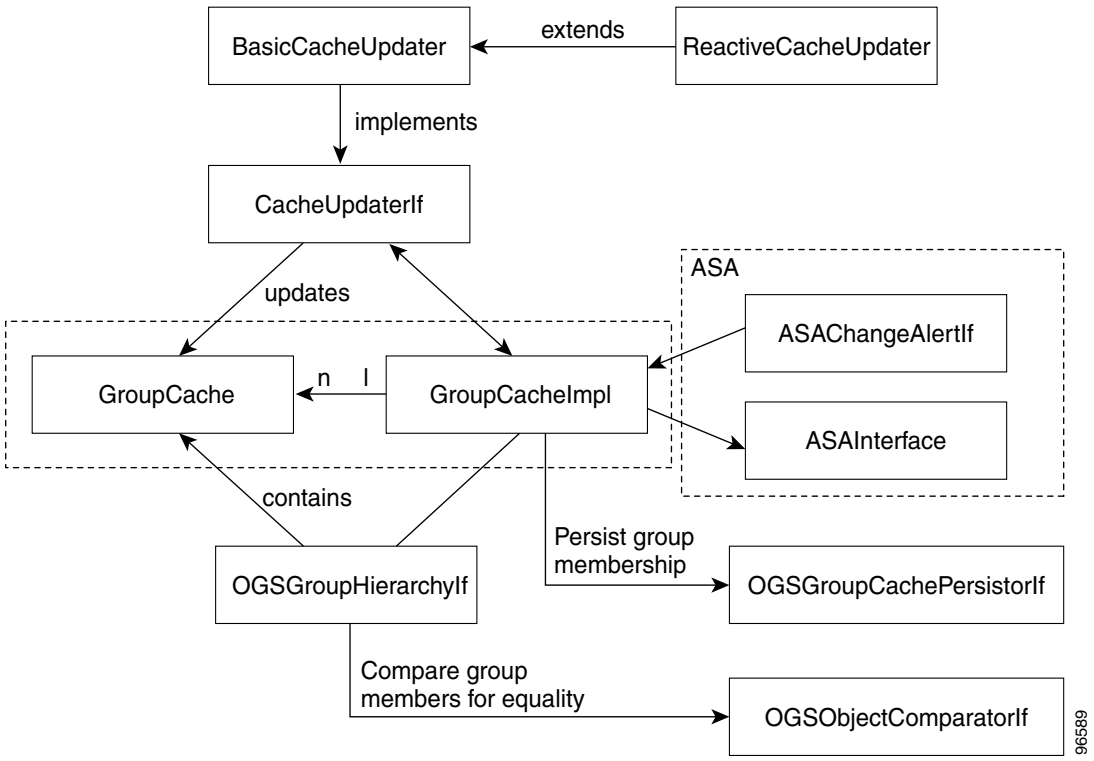
- **OGSEnvironmentAdapterImpl**

  This class can be used to do any postprocessing operation after the OGSServer starts in the environment.

  For example, the published URNs need to be unpublished URNs need to be unpublished when the daemon manager stops the OGS server. This is done by the implementation of OGSEnvironmentAdapterIf interface. Another related property is OGSEnvRegistrationName which specifies the process or daemon name of the OGS server.

Figure 30-1 shows the relationships among these OGS Server properties.

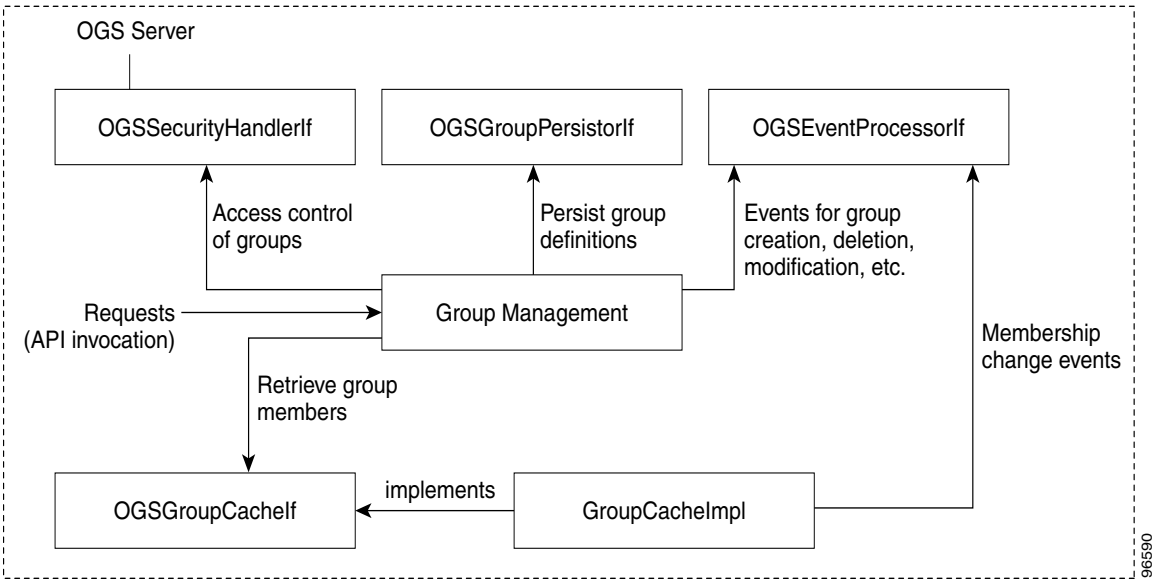
Figure 30-1 OGS Server Interfaces



In addition to these interfaces, the default OGS Cache Manager (class GroupCacheImpl) has three cache update mechanisms that it activates periodically. All three update mechanisms use implementations of the interface CacheUpdaterIf shown in Figure 30-2:

1. **BasicCacheUpdater—**Updates the cached membership by evaluating all the groups.
2. **ReactiveCacheUpdater**—Collects change information (such as objects added or deleted, attribute value changes, etc.) and updates the cached membership f the groups affected by those changes.
3. **InvalidCacheUpdater—**Updates cached membership by evaluating groups for which the previous evaluation had failed.

Figure 30-2 OGS Cache Manager Interfaces



Of these three updater mechanisms, the first two (BasicCacheUpdater and ReactiveCacheUpdater) can be customized. To customize the two updaters, you can set the following OGSServer.properties parameters:

- **GroupUpdateFrequency**

  Sets the interval in seconds between activations of the BasicCacheUpdater. A value of -1 disables this updater. The default value is 60.

- **GroupUpdaterClass**

  TBD

- **ReactiveUpdaterClass**

  Specifies the name of the class that implements CacheUpdaterIf (see Figure 30-2). This is normally the default class or one of its subclasses: ReactiveCacheUpdater in com.cisco.nm.xms.ogs.server. For an example of a custom reactive updater, see KilnerReactiveUpdater in com.cisco.nm.xms.ogs.Kilner10.

- **ReactiveUpdateFrequency**

  Sets the interval in seconds between activations of the reactive updater. A value of -1 disables this updater. The default value is -1 (that is, the reactive updater is disabled in the default configuration.

OGSServer.properties also allows you to customize the following properties:

- **ProductId**

  Sets the string representing the ID of the product in which this OGS implementation is used. This is used to create strings for names of events published by this instance of OGS.

- **InstanceId**

  Sets the string representing this instance of OGS. This is reserved for future use.

- **IgnoreCacheInitErrors**

Specifies whether you want cache initialization errors to be ignored when creating dynamic groups, so that dynamic group creation will not fail. This property does not affect static groups, since these groups, by definition, require that their membership be completely determined at creation time. We recommend that you set this property value to True; the default value is False.

- **EventProcessorImplClass**

  Specifies the name of the class that implements OGSEventProcessorIf (see Figure 30-2). This interface allows a product to customize how OGS events are dispatched. Normally, this is the default class DefaultEventProcessor, which dispatches OGS events with no modifications.

  For an example of a custom Event Processor, see KilnerEventConsolidator in com.cisco.nm.xms.ogs.Kilner10 and the "Creating a Custom OGS Event Processor" section on page 30-15, which examines KilnerEventConsolidator in detail.

Creating a Custom OGS Event Processor

The DefaultEventProcessor in OGS dispatches OGS events without modifications. Each such event is dispatched as a message, with a subject name that looks like this:

`cisco.mgmt.cw.`*`product_id`*`.`*`ip_address`*`.ogs.alert`

Where:

- *`product_id`* is the name of the product publishing the event.
- *`ip_address`* is the IP of the server where OGS is executing.

The subject of the message describes an atomic change to the data OGS manages, including group creation, deletion, modification, renaming, or membership change. This can often result in hundreds of messages.

For example, consider a group high in a group hierarchy that is deleted. In this case, atomic "deletion" messages will be generated for every subgroup. If there are 50 such subgroups, the default Event Processor will forward 51 "group deletion" messages. Similarly, a single modification in a group containing 1000 objects would result in 1000 group-membership change events.

You can create your own Event Processor, and use the OGSServer.properties parameter "EventProcessorImplClass" to implement it (see the "Customizing OGS Server Interfaces" section on page 30-11). The custom Event Processor can handle events in any way you wish, up to and including suppressing all events.

Kilner offers an example of a custom Event Processor: KilnerEventConsolidator in com.cisco.nm.xms.ogs.Kilner10. This processor dispatches customized "consolidated events", which logically group or aggregate atomic OGS event information based on the operation that triggered those events.

The subject name for a consolidated event in Kilner looks like this:

`cisco.mgmt.cw.kilner.`*`ip_address`*`.ogs.alert.`*`consolidatedevent`*

Where *`consolidatedevent`* is a single message that consolidates the individual events that are the results of a single "groupcreation", "groupdeletion", "groupmodification", "grouprename", "membershipchange", or "serverstart" operation. The body of the message will contain the atomic event data consolidated under that "heading".

For every consolidated event, an instance of class ConsolidatedOGSEvent in com.cisco.nm.xms.ogs.kilner10 is sent in an object message.

Processing requires:

1. Using the method ogsOperation() to retrieve the OGS operation that resulted in the event. The relevant OGS operations (defined in class ConsolidatedOGSEvent) are:
   - Group Creation: GROUP_CREATION, GROUP_COPY, GROUP_HIERARCHY_COPY, CREATE_PARTITION
   - Group Deletion: GROUP_DELETION
   - Group Rename: GROUP_RENAME
2. If the event is of interest, retrieving and processing the collection of associated AtomicEventData.

For more on creating a consolidated Event Processor, see:

- The Kilner VOB.
- The ConsolidatedOGSEvent and AtomicEventData Javadoc at the OGS Portal, https://mco.cisco.com/ubiapps/portal/go.jsp?portal_id=6625.

Handling OGS Exceptions

The OGS API defines the exception classes shown in Table 30-2. The list includes a description of the kinds of operations that typically result in these exceptions. All of these exception classes are subclasses of OGSException. In addition to these exceptions, you should be sure that your application is prepared to handle instances of OGSException that represent internal errors.

Table 30-2 OGSException Classes

Exception	Description	Operations
OGSException.AccessDenied	The user does not have the access required to perform the requested operation.	All operations except retrieveApplicationSchema, validateRule and verifyService.
OGSException.GroupExists	The named group already exists.	createGroup, copyGroup, copyGroupHierarchy, renameGroup
OGSException.GroupDoesNotExist	The named group does not exist.	All operations except retrieveApplicationSchema, validateRule, createGroup, retrieveGroupList, verifyService.
OGSException.InvalidParent	The parent group, as specified in the name of a group, is invalid (does not exist).	createGroup, copyGroup, copyGroupHierarchy
OGSException.IllegalRename	The new name provided while renaming a group is not legal. OGS does not allow renaming a group such that its parent group is changed. For example: This exception will be thrown if you attempt to rename group /A/B/C to /A/D/C1.	renameGroup
OGSException.IllegalContainer	A process attempted to create a container group (a group with no rule) under the root.	createGroup, copyGroup, copyGroupHierarchy

Table 30-2 OGSException Classes (continued)

Exception	Description	Operations
OGSException.IllegalName	The group or class name is malformed.	All operations except retrieveApplicationSchema, retrieveGroupList, verifyService.
OGSException.IllegalGroupDefinition	Thrown whenever the definition of a group is malformed. Currently, the only condition that will provoke this exception is when the evaluation type is not specified as either static or dynamic.	All constructors of OGSGroupDefinition
OGSException.InvalidRule	The rule is invalid.	createGroup, copyGroup, copyGroupHierarchy, modifyGroup, evaluateRule, validateRule.
OGSException.IllegalNullParameter	One of the parameters being passed has a illegal null value.	All operations except verifyService, retrieveApplicationSchema.
OGSException.IllegalParameter	One of the parameters has an illegal (but not null) value.	evaluateGroup (when the membership format is not one of the specified formats).
OGSException.ASANotFound	The operation could not find an ASA that could evaluate the rule.	createGroup, modifyGroup, validateRule, evaluateRule.
OGSException.GroupInitializationFailed	The initialization of the group's membership failed.	evaluateGroup
OGSException.ASAError	The process received an error from the ASA while evaluating a rule.	createGroup, copyGroup, copyGroupHierarchy, modifyGroup, evaluateRule, evaluateGroup.

Creating OGS ASAs

In order to use OGS, you must create and associate with your application an Application Service Adapter (ASA). The OGS Server depends on ASAs to:

- Respond to queries that seek to verify the presence of an ASA.
- Register an application schema that the OGS Server supports. OGS Servers process schema registration files at startup.
- Evaluate a rule and return to the OGS Server the OGS objects that satisfy it.
- Given a list of object IDs and a set of attributes, return the corresponding OGS objects.

The remainder of this topic explains what is needed to create an OGS ASA, including:

- Understanding ASA Infrastructure Modules
- Customizing ASA Infrastructure Modules
- Running a Customized ASA

About ASA Implementations

There is no default ASA supplied with OGS. However, the OGS team provides a set of libraries and tools to assist in ASA development, and default or example implementations for most modules.

Application developers and the OGS Team have also collaborated in creating OGS ASAs for several applications. These serve as ready-made examples of how to create customized ASAs, and their code is available on a read-only basis to Cisco developers with access to ClearCase:

- For code related to the ASA infrastructure classes (and example implementations, such as the Generic SQL ASA), see the OGS vob, /vob/enm_ogs/share/classes/server. All packages are defined within this vob.
- For the latest code related to the generic OGS, use the views created in project ogs.
- To access code related to Kilner and its ASAs (such as the VMSAQueryGenerator in the class com.cisco.nm.xms.ogs.kilner.), use the views created in project ogs_kilner.

Managing ASA Processes

OGS Server will load an ASA into its JVM if the ASA Registration File (see the "Running a Customized ASA" section on page 30-36) specifies a Location type of "local". OGS Server will manage startup and shutdown, and registration with CSTM, of any local ASA.

Developers should use the CWCS Daemon Manager (also known as Process Manager; see Chapter 17, "Using the Daemon Manager") to manage startup and shutdown of the OGS Server.

A "local" ASA is the only ASA type allowed in this release of OGS. Future OGS versions will allow ASAs to run remotely, either alone or within the JVM of a non-OGSServer application.

When planning your application's use of OGS in future releases, please note that remote ASAs will need to register themselves with OGSServer and with CSTM, and must be controlled by the CWCS Daemon Manager, to ensure they are available to requests from OGS Servers.

Using ASAs to Aggregate Data

Under normal circumstances, OGSServer is responsible for aggregating data resulting from the evaluation of a query on multiple instances of an ASA. However, it is possible to customize ASAs to perform this function if requirements make this necessary.

For example, an ASA may require tight cooperation between its instances to evaluate rules. In this case, the ASA can register a single instance that will then be responsible for communicating client requests to the other instances and aggregating the results before returning them to the OGS Server.

About the OGSServer-ASA Interface

Provided as `ASAInterface` in `com.cisco.nm.xms.ogs.asa`, this component allows the ASA and OGSServer to exchange commands, queries and result data via the Common Services Transport Mechanism.

All current ASA structures implement this interface. However, the interface is pluggable, which means that you can substitute your own ASA structure for the one described here.

You may want to do this if you have a very simple object grouping requirement.

For example, you may require OGS to parse a flat file containing nothing but a list of user names. In this case, you can write your own simple ASA class and implement it. However, your ASA class must:

- Be capable of responding on its own to requests from the OGSServer as normal ASAs do (that is, it must be able to parse rules, access the flat file, etc.).

- Implement ASAInterface.
- Be registered with OGSServer. You do this by entering the name of this class as the `classname` value in the ASA Registration file (for more on this, see the "Registering the ASA with OGS" section on page 30-36).

Understanding ASA Infrastructure Modules

The basic function of your application's ASA is to validate and evaluate rules passed to it by the OGS Server. To perform these functions, your ASA needs the following modules:

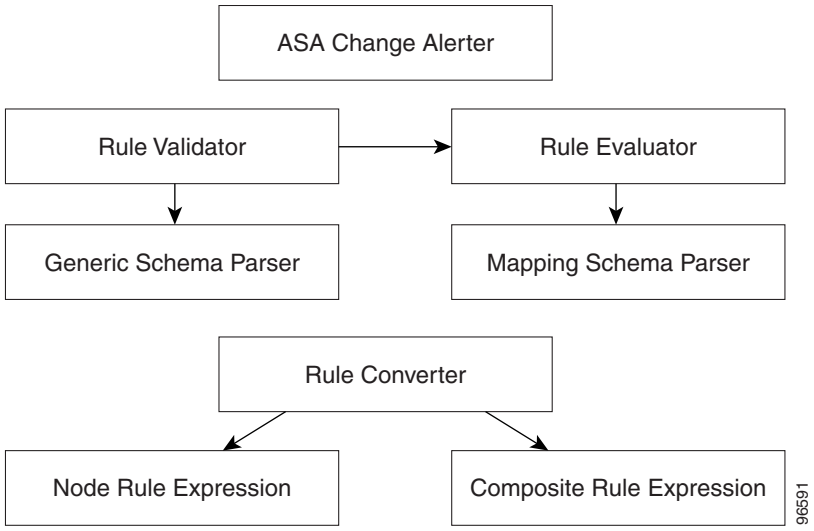
1. Rule Validator: Ensures that a rule passed to it by the OGS Server is valid. For details, see the "About the Rule Validator" section on page 30-20.
2. Generic Schema: Provides for the Rule Validator a parsed object tree containing all the combinations of groupable classes, attributes, operators and values allowable in rules. For details, see the "About the Generic Schema" section on page 30-20.
3. Rule Evaluator: Converts a validated rule into a query appropriate to the application's data-storage structure, and returns to the OGS Server the objects that match the rule. For details, see the "About the Rule Evaluator" section on page 30-21.
4. Mapping Schema: Maps groupable classes against the data-storage elements in an application domain and supplies the mapping to the Rule Evaluator for query formation. For details, see the "Customizing the Mapping Schema" section on page 30-27.
5. Rule Converter: Translates the notation for a rule used by the OGS Server into the notation used internally by the ASA infrastructure. For details, see the "About the Rule Converter" section on page 30-21.
6. Node Rule Expression: Models the basic type of Rule Expression that the ASA will validate and evaluate. For details, see the "About Node Rule Expressions" section on page 30-21.
7. Composite Node Rule Expression: Models complex Rule Expressions involving two or more Node Rule Expressions. For details, see the "About Composite Rule Expressions" section on page 30-22.
8. ASA Change Alerter: Alerts the OGS Server when an object changes. For details, see the "About the ASA Change Alerter" section on page 30-22.

Figure 30-3 shows the relationships among these modules.

For guidelines and examples on how to customize these infrastructure modules for your application, see the "Customizing ASA Infrastructure Modules" section on page 30-22.

For other examples of custom ASAs, see the ClearCase VOBs listed in the "Creating OGS ASAs" section on page 30-17.

Figure 30-3 ASA Infrastructure Modules



About the Rule Validator

The Rule Validator ensures that:

1. Each groupable class in the rule is valid.
2. Each groupable attribute in the rule is valid.
3. Each groupable operator in the rule is valid.
4. The rule as a whole evaluates to a valid class. This includes all of the Rule Expressions, combined by AND, OR, or EXCLUDE operators.
5. The object ID values specified in the rule refer to actual objects. The OIDs can contain the value "ANY" or a "$"- separated value produced by the ASA.

In addition to these tasks, the Rule Validator can also check the validity of any:

- Value range specified in the rule.
- Value enumeration specified in the rule.

The Rule Validator depends on the Generic Schema to perform these tasks.

About the Generic Schema

The Generic Schema consists of a Parser and XML Schema file describing the groupable classes, attributes, operators and values (including any value ranges and enumerations) valid for all objects. The Parser supplies this structure to the Rule Validator.

About the Rule Evaluator

The Rule Evaluator:

1. Evaluates the rule for a particular OGS class.
2. Combines individual Rule Expressions within the rule so that the object data can be retrieved. “Combining” the Rule Expressions can mean turning all of them into one query (as in the SQLQueryGenerator), or evaluating each Rule Expression separately and then combining the results based on the operator (as with the VMSAQueryGenerator).
3. Returns to the OGS Server the objects that match the rule.
4. Defines a unique object ID to represent the OGS object in the domain. For example, in ANI, the database identifier (DBID) can be used as the OID to define a unique piece of data. In the SMARTS repository, it can be the instance name for a class (which is assumed to be unique).
5. For each object, retrieves other attributes as required. The attribute data can map to other columns in the tables for a relational database, or it can map to class attributes in the SMARTS repository.

When combining Rule Expressions, the Rule Evaluator must group them into queries specific to the needs of the data-storage resource. These queries can be:

- SQL queries, as in the case where the storage domain is a relational database management system.
- A custom query mechanism, as in the case where the storage domain is a SMARTS repository.

The Rule Evaluator depends on the Mapping Schema to format the query appropriately.

About the Mapping Schema

The Mapping Schema consists of a Parser and XML Schema file that specifies the mapping between groupable OGS classes and the elements in the application’s data-storage resource where the data for these objects is actually stored. The Rule Evaluator uses this information to convert rules into data-store-specific queries.

About the Rule Converter

The ASA infrastructure notation for a Rule Expression is different from the notation used by the OGS Server. This is necessary because the ASA can store extra information in the rule to help the Rule Evaluator generate queries.

Example

A Rule Expression for use with a SQL ASA can contain the entire query formed by the Query generator. The Rule Expression to be used with the Kilner VMSA can contain other types of information specific to a SMARTS Repository.

The Rule Converter converts any rule passed by the OGS Server into a rule format used by the Rule Validator and Rule Evaluator.

About Node Rule Expressions

A Node Rule Expression contains four elements:

1. The groupable class.
2. The groupable attribute.

3. The groupable operator.
4. The value.

For example: In the Node Rule Expression `Device IpAddress contains "172.20"`, the class is `Device`, the attribute is `IpAddress`, the operator is `contains`, and the value is `"172.20"`.

Similar examples of Node Rule Expressions would include: `Device SystemLocation contains "US"`, or `Device SystemLocation contains"San Jose"`.

About Composite Rule Expressions

A Composite Rule Expression contains two or more Rule Expressions combined by the operators AND, OR or EXCLUDE.

The Rule Expressions combined in this way can be either Node Rule Expressions or other Composite Rule Expressions.

For example, we can create a Composite Rule Expression using just two Node Rule Expressions: `Device IpAddress contains"172.20"`AND Device SystemLocation contains "US"

We can also create a Composite Rule Expression that combines the foregoing Composite Rule Expression with another Node Rule Expression: `(Device IpAddress contains"172.20" AND Device SystemLocation contains "US") OR Device SystemLocation contains"San Jose"`.

About the ASA Change Alerter

The ASA Change Alerter informs the OGS Server whenever a change to an object takes place in the data-storage resource containing the object data. The ASA Change Alerter must inform OGS Server whenever:

- An object is added.
- An object is deleted.
- An object is modified.
- An individual object's attribute values changes.
- A groupable class’s attribute values change..
- The domain data provider is available (that is, when the relational or object-oriented database comes up).

Customizing ASA Infrastructure Modules

The ASA infrastructure modules must be customized depending upon your application’s requirements for:

- The types of objects your application will group.
- The way you want to group them.
- The way your application stores the object data.

The following topics provide guidelines and examples for customizing each of the ASA Infrastructure Modules:

- Customizing the Rule Validator
- Customizing the Generic Schema

Customizing the Rule Validator

The Rule Validator function is provided by the default validator class GenericValidatorImpl in com.cisco.nm.xms.ogs.asa. You will rarely need to extend this class to customize it.

Customizing the Generic Schema

The Generic Schema parser has been designed for reuse with very little change. The Generic Schema parser function is provided by a default schema parser class, OGSSchemaParser in com.cisco.nm.xms.ogs.server.parser. You should reuse this class as needed.

It is not possible to create a default Generic Schema file that will work with any application. However, you can create a Generic Schema file for practically any combination of classes, attributes, operators and values using the following XML DTD:

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

<!ELEMENT domain (application)>
<!ATTLIST domain
          name  CDATA #REQUIRED>

<!ELEMENT application (class+,complexType+,group+)>
<!ATTLIST application
          name  CDATA #REQUIRED>

<!ELEMENT class EMPTY>
<!ATTLIST class
          name CDATA #REQUIRED
          type CDATA #REQUIRED
     groupable (yes|no) #REQUIRED >


<!ELEMENT complexType (extension? , attribute*) >
<!ATTLIST complexType
          name  CDATA #REQUIRED>

<!ELEMENT extension EMPTY>
<!ATTLIST extension
          base CDATA #REQUIRED>

<!ELEMENT attribute (operatorGroup?,restriction?) >
<!ATTLIST attribute
          name CDATA #REQUIRED
          type CDATA #REQUIRED
     groupable (yes|no) #REQUIRED
     displayable (yes|no) #IMPLIED
     returnable (yes|no) #IMPLIED>


<!ELEMENT operatorGroup EMPTY>
<!ATTLIST operatorGroup
          ref CDATA #REQUIRED>
```

```
<!ELEMENT restriction (enumeration*,minInclusive?,maxInclusive?) >

<!ELEMENT enumeration EMPTY>
<!ATTLIST enumeration
          value CDATA #REQUIRED
          alias CDATA #IMPLIED>

<!ELEMENT minInclusive EMPTY>
<!ATTLIST minInclusive
          value CDATA #REQUIRED>

<!ELEMENT maxInclusive EMPTY>
<!ATTLIST maxInclusive
          value CDATA #REQUIRED>

<!ELEMENT group (extension? , operator*) >
<!ATTLIST group
          name  CDATA #REQUIRED>

<!ELEMENT operator EMPTY>
<!ATTLIST operator
          name  CDATA #REQUIRED>



]>

<domain name="CMF">
    <application name="DCR">
        <class name = "Device" type="DeviceType" groupable="yes"/>

        <complexType name="DeviceType">
               <attribute name="DisplayName" type="string" groupable="yes"
displayable="yes" returnable="yes">
                  <operatorGroup ref="StringOperatorsGroup"/>
               </attribute>

               <attribute name="ManagementIpAddress" type="string" groupable="yes"
displayable="no" returnable="yes">
                  <operatorGroup ref="StringOperatorsGroup"/>
               </attribute>

               <attribute name="HostName" type="string" groupable="yes" displayable="no"
returnable="yes">
                  <operatorGroup ref="StringOperatorsGroup"/>
               </attribute>

              <attribute name="DomainName" type="string" groupable="yes" displayable="no"
returnable="yes">
                  <operatorGroup ref="StringOperatorsGroup"/>
               </attribute>

               <attribute name="DeviceIdentity" type="numeric" groupable="yes"
displayable="no" returnable="yes">
                  <operatorGroup ref="EqualityOperatorsGroup"/>
               </attribute>

               <attribute name="SystemObjectID" type="string" groupable="yes"
displayable="no" returnable="yes">
                  <operatorGroup ref="StringOperatorsGroup"/>
               </attribute>
```

```
                <attribute name="Category" type="string" groupable="yes" displayable="no"
returnable="yes">
              <operatorGroup ref="StringOperatorsGroup"/>
               </attribute>

               <attribute name="Series" type="string" groupable="yes" displayable="no"
returnable="yes">
              <operatorGroup ref="StringOperatorsGroup"/>
               </attribute>

               <attribute name="Model" type="string" groupable="yes" displayable="no"
returnable="yes">
              <operatorGroup ref="StringOperatorsGroup"/>
               </attribute>

               <attribute name="MDFId" type="string" groupable="yes" displayable="no"
returnable="yes">
                  <operatorGroup ref="EqualityOperatorsGroup"/>
               </attribute>

               <attribute name="UDF0" type="string" groupable="yes" displayable="no"
returnable="yes">
              <operatorGroup ref="StringOperatorsGroup"/>
           </attribute>

           <attribute name="UDF1" type="string" groupable="yes" displayable="no"
returnable="yes">
               <operatorGroup ref="StringOperatorsGroup"/>
           </attribute>

           <attribute name="UDF2" type="string" groupable="yes" displayable="no"
returnable="yes">
               <operatorGroup ref="StringOperatorsGroup"/>
           </attribute>

           <attribute name="UDF3" type="string" groupable="yes" displayable="no"
returnable="yes">
               <operatorGroup ref="StringOperatorsGroup"/>
           </attribute>

               <attribute name="UDF4" type="string" groupable="yes" displayable="no"
returnable="yes">
              <operatorGroup ref="StringOperatorsGroup"/>
           </attribute>

           <attribute name="UDF5" type="string" groupable="yes" displayable="no"
returnable="yes">
               <operatorGroup ref="StringOperatorsGroup"/>
           </attribute>

           <attribute name="UDF6" type="string" groupable="yes" displayable="no"
returnable="yes">
               <operatorGroup ref="StringOperatorsGroup"/>
           </attribute>

           <attribute name="UDF7" type="string" groupable="yes" displayable="no"
returnable="yes">
               <operatorGroup ref="StringOperatorsGroup"/>
           </attribute>

               <attribute name="UDF8" type="string" groupable="yes" displayable="no"
returnable="yes">
              <operatorGroup ref="StringOperatorsGroup"/>
           </attribute>
```

```
            <attribute name="UDF9" type="string" groupable="yes" displayable="no"
returnable="yes">
                 <operatorGroup ref="StringOperatorsGroup"/>
            </attribute>

            <attribute name="UDF10" type="string" groupable="yes" displayable="no"
returnable="yes">
                 <operatorGroup ref="StringOperatorsGroup"/>
            </attribute>

            <attribute name="UDF11" type="string" groupable="yes" displayable="no"
returnable="yes">
                 <operatorGroup ref="StringOperatorsGroup"/>
            </attribute>

                <attribute name="UDF12" type="string" groupable="yes" displayable="no"
returnable="yes">
                <operatorGroup ref="StringOperatorsGroup"/>
            </attribute>

            <attribute name="UDF13" type="string" groupable="yes" displayable="no"
returnable="yes">
                 <operatorGroup ref="StringOperatorsGroup"/>
            </attribute>

            <attribute name="UDF14" type="string" groupable="yes" displayable="no"
returnable="yes">
                 <operatorGroup ref="StringOperatorsGroup"/>
            </attribute>

            <attribute name="UDF15" type="string" groupable="yes" displayable="no"
returnable="yes">
                 <operatorGroup ref="StringOperatorsGroup"/>
            </attribute>

        </complexType>


        <group name="StringOperatorsGroup">
            <operator name="equals" />
           <operator name="contains" />
           <operator name="startswith" />
           <operator name="endswith" />
        </group>

        <group name="EqualityOperatorsGroup">
            <operator name="equals" />
        </group>

      </application>
</domain>
```

Customizing the Rule Evaluator

The Rule Evaluator must implement the interface QueryGeneratorIf in class com.cisco.nm.xms.ogs.asa.

As the Rule Evaluator must query the application's data resource, no generic implementation of the Rule Evaluator exists. You must create a custom Rule Evaluator for your application and the data-storage resource with which your application works.

However, implementations of the Rule Evaluator can be found at:

- com.cisco.nm.xms.ogs.asa.genericsqlasa.SQLQueryGenerator. This SQLQueryGenerator Rule Evaluator is designed to work with a SQL-compatible relational database management system, and with the custom Mapping Schema parser and file described in the "Customizing the Mapping Schema" section on page 30-27.
- com.cisco.nm.xms.ogs.kilner10.vmsa.VMSAQueryGenerator. This VMSAQueryGenerator Rule Evaluator is designed to work with the SMARTS Repository. It is a good example of how to work with object-oriented data sources.

SQLQueryGenerator uses the Generic SQL ASA module SQLEvaluatorIf to evaluate SQL queries in the database. The default version of the SQLEvaluatorIf interface is CMFEvaluator in com.cisco.nm.xms.ogs.asa.genericsqlasa. CMFEvaluator uses the CWCS database utility DBService2 to evaluate SQL queries.

You must configure CMFEvaluator for your application by changing the "Database_Name" value in the DBServer.properties file to give the database name (e.g., "RME") followed by "_Implementation_Details.properties".

You need not provide the user name, password, data source URL, or other database properties. They will be picked up from DBServer.properties automatically.

Customizing the Mapping Schema

The Mapping Schema's task of modeling the data source for the Rule Evaluator means that no default Mapping Schema implementation can exist, as the data-storage resource for each application can vary widely.

For example, the elements in the data resource can be:

- Tables, if the data source is a relational database.
- Classes, if the data source is an object-oriented data store.
- Files, if the data source relies on flat files.

Accordingly, you must write a Mapping Schema appropriate for the way your application sources and organizes its data. Since the Mapping Schema parser requires an XML file to parse, you must also create a Mapping Schema XML file customized for your application's data structure.

The Generic SQL ASA is a customization of the ASA infrastructure for querying a SQL-compliant RDBMS. It uses a custom Mapping Schema parser (RDBMSSchemaValidator in class com.cisco.nm.xms, ogs.asa.genericsqlasa) which is in turn used by the SQLQueryGenerator Rule Evaluator.

RDBMSSchemaValidator was written to parse a Mapping Schema XML file conforming to the following DTD:

```
<?xml version="1.0"?>
<!DOCTYPE Domain [
<!ELEMENT Domain (Application)>
<!ATTLIST Domain
          Name  CDATA #REQUIRED>

<!ELEMENT Application (Mappings)>
<!ATTLIST Application
          Name  CDATA #REQUIRED>


<!ELEMENT Mappings (ClassMap+,group*)>
<!ATTLIST Application
```

```
          Name  CDATA #REQUIRED>

<!ELEMENT ClassMap (OGSClass,ToTable,HierarchyInfo,PropertyMap+) >

<!ELEMENT OGSClass EMPTY>
<!ATTLIST OGSClass
          Name CDATA #REQUIRED>

<!ELEMENT ToTable (PrimaryKey) >
<!ATTLIST ToTable
          Name CDATA #REQUIRED>

<!ELEMENT PrimaryKey (Column+)>

<!ELEMENT Column (ColumnName,JDBC_Type,Operator)>
<!ELEMENT ColumnName (#PCDATA) >
<!ELEMENT JDBC_Type (#PCDATA) >
<!ELEMENT Operator (#PCDATA) >
<!ELEMENT HierarchyInfo (HierarchyType,ConditionalColumnInfo?) >
<!ELEMENT HierarchyType (#PCDATA) >
<!ELEMENT ConditionalColumnInfo (ColumnName,JDBC_Type,Operator,ColumnCondition) >
<!ELEMENT ColumnCondition (#PCDATA) >

<!ELEMENT PropertyMap (OGSAttribute,(AttributeToColumnMapping |
AttributeToTableDireCSTMapping | AttributeToTableIndireCSTMapping |
AttributeToTableConditionalDireCSTMapping),operatorMapGroup?)>

<!ELEMENT OGSAttribute EMPTY>
<!ATTLIST OGSAttribute
          Name CDATA #REQUIRED>

<!ELEMENT AttributeToColumnMapping (ColumnName,JDBC_Type)>


<!ELEMENT AttributeToTableDireCSTMapping (MappedTable,PrimaryToMappedAssociation)>
<!ELEMENT MappedTable EMPTY >
<!ATTLIST MappedTable
          Name CDATA #REQUIRED>

<!ELEMENT PrimaryToMappedAssociation (PrimaryTableColumnNames+,MappedTableColumnNames+)>

<!ELEMENT PrimaryTableColumnNames (PrimaryTableColumn+)>

<!ELEMENT PrimaryTableColumn (ColumnName) >

<!ELEMENT MappedTableColumnNames (MappedTableColumn+) >

<!ELEMENT MappedTableColumn (ColumnName) >


<!ELEMENT AttributeToTableConditionalDireCSTMapping
(MappedTable,PrimaryToMappedAssociation,MappedConditionalColumnName,MappedConditionalColum
nValue,MappedValueColumnName)>

<!ELEMENT MappedConditionalColumnName (#PCDATA) >
<!ELEMENT MappedConditionalColumnValue (#PCDATA) >
<!ELEMENT MappedValueColumnName (#PCDATA) >

<!ELEMENT AttributeToTableIndireCSTMapping
(MappedTable,IndirectTable,IndirectToMappedAssociation,IndirectToPrimaryAssociation) >
<!ELEMENT IndirectTable EMPTY >
<!ATTLIST IndirectTable
          Name CDATA #REQUIRED>
```

```
<!ELEMENT IndirectToMappedAssociation (IndirectTableColumnNames+,MappedTableColumnNames+)
>
<!ELEMENT IndirectTableColumnNames (IndirectTableColumn+) >

<!ELEMENT IndirectTableColumn (ColumnName) >

<!ELEMENT IndirectToPrimaryAssociation
(IndirectTableColumnNames+,PrimaryTableColumnNames+) >


<!ELEMENT operatorMapGroup EMPTY>
<!ATTLIST operatorMapGroup
          ref CDATA #REQUIRED>

<!ELEMENT group (OperatorMap+) >
<!ATTLIST group
          Name  CDATA #REQUIRED>

<!ELEMENT OperatorMap (OGSOperator,ToOperator) >

<!ELEMENT OGSOperator EMPTY>
<!ATTLIST OGSOperator
          Name CDATA #REQUIRED>

<!ELEMENT ToOperator (SQLOperator) >

<!ELEMENT SQLOperator EMPTY>
<!ATTLIST SQLOperator
          Name CDATA #REQUIRED>

]>
```

The RDBMSSchemaValidator DTD was written to allow you to map data from any SQL-compliant relational DBMS. It lets you specify exactly how individual OGS classes map to RDBMS tables and how to map OGS attributes, operators and values to table columns.

With it, you can create Mapping Schema XML files that will handle all aspects of the task of mapping OGS classes to RDBMS tables.

The following examples demonstrate how to do this for many of the most typical cases.

Example: One Table Maps to One Concrete Class

In this case, the RDBMS contains a table called “DeviceTable”. It has many columns, including “SysName” and “SysLocation”. Using the RDBMSSchemaValidator DTD, we can specify that the OGS class “Device” maps directly to “DeviceTable”.

Similarly, the OGS class “Device” can have the attributes “SysName” and “SysLocation”. These OGS attributes map directly to the corresponding columns in “DeviceTable”.

In the Mapping Schema XML file, you can make this type of mapping explicit by supplying the value ONE_TABLE_PER_CONCRETE_CLASS for the element *HierarchyType* inside the element *HierachyInfo*.

Note The Generic SQL ASA Rule Evaluator (SQLQueryGenerator) currently supports conversion of rules into SQL queries for Hierarchy classes of the type given in this example. You need not modify SQLQueryGenerator for this purpose.

Example: One Table Maps to a Hierarchy of Classes

In this example, the RDBMS table called “Device Table” covers all device types. Each row in the table can contain data for a different type of device, such as “Router”, “Switch”, or “PhoneAccessSwitch”. Every row has the column "DeviceType", which has a different value depending on the type of device described in that row.

Using the RDBMSSchemaValidator DTD, we can design many OGS classes for this mapping. One of the simplest would be to define a base OGS class called “Device”, with many subclasses for “Router”, “Switch”, etc.

All of these subclasses can also map to “DeviceTable”, but with a condition on the “DeviceType” column. In the XML file, the <HierarchyType> for these classes will be ONE_TABLE_PER_HIERARCHY.

Note The Generic SQL ASA Rule Evaluator (SQLQueryGenerator) can *not* convert rules into SQL queries for this Hierarchy class. If you want to create table mappings for this format, you should implement your own SQLQueryGenerator in the lines of the default implementation.

Example: Multiple Tables Map to One Abstract Class

In this case, the RDBMS has no single “Device” table. Instead, it has a table for each kind of device: “Router”, “Switch”, “PhoneAccessSwitch”, and so on.

To handle this, we create an abstract base OGS class called “Device” which does not map to any DB table, and concrete subclasses of “Device”, like “Router” and “Switch”, each of which maps to the corresponding RDBMS table.

The objects in the “Device” class are just a summation of the objects in the concrete subclasses. The set of OGS attributes common to each of the concrete subclasses belongs to the parent “Device” OGS class, so the similarly named columns in the tables for individual devices are integrated into the class hierarchy. In the Mapping Schema XML file, the *HierarchyType* for these classes is ONE_TABLE_PER_CLASS.

Note The Generic SQL ASA Rule Evaluator (SQLQueryGenerator) can *not* convert rules into SQL queries for this Hierarchy class. If you want to create table mappings for this format, you should implement your own SQLQueryGenerator in the lines of the default implementation.

Example: Mapping Class Attributes to Table Columns

We have the OGS class “Device” mapped to the RDBMS table “Device”, and the OGS “Device” attribute “SystemLocation” mapped to the column “SystemLocation” in the “Device” table. We could write this in the Mapping Schema XML file as:

```
<Domain Name="SomeDomainName">
<Application Name="OGS">
<Mappings>
<ClassMap>
  <OGSClass Name="Device"/>
  <ToTable Name="Device">
      <PrimaryKey>
            <Column>
            <ColumnName>DeviceID</ColumnName>
            <JDBC_Type>INTEGER</JDBC_Type>
            <Operator>=</Operator>
            </Column>
      </PrimaryKey>
   </ToTable>
<HierarchyInfo>
```

```
   <HierarchyType>ONE_TABLE_PER_CONCRETE_CLASS</HierarchyType>
   </HierarchyInfo>

   <PropertyMap>
      <OGSAttribute Name="SystemLocation"/>
      <AttributeToColumnMapping>
               <ColumnName>SystemLocation</ColumnName>
               <JDBC_Type>VARCHAR</JDBC_Type>
       </AttributeToColumnMapping>
       <operatorMapGroup ref="StringOperatorMap"/>
   </PropertyMap>
</ClassMap>

<group Name="StringOperatorMap">

     <OperatorMap>
          <OGSOperator Name="equals"/>
          <ToOperator>
             <SQLOperator Name="="/>
          </ToOperator>
     </OperatorMap>

     <OperatorMap>
          <OGSOperator Name="contains"/>
          <ToOperator>
             <SQLOperator Name="LIKE"/>
          </ToOperator>
     </OperatorMap>
</group>

</Mappings>
</Application>
</Domain>
```

In this example, the column “SystemLocation” is of type “VARCHAR”. The mappings for the OGS operators for this OGS attribute type, “equals” and “contains”, are defined in the operators group, “StringOperatorMap”.

Note The Generic SQL ASA Rule Evaluator (SQLQueryGenerator) currently supports the kind of attribute mapping shown in this example

Examples: Mapping Attributes Across Tables Using Foreign Keys

If you want to have an OGS class called “Device” with the attributes “IPAddress” and “IPSubnetMask”. But the RDBMS tables have:

- Table “Device”, with columns “SystemLocation”, “SysObjectID”, “SystemContact” and “DeviceID”.
- Table “IPTable”, with columns “DevID”, “IPAddress” and “IPSubnetMask”. “DevID” is a foreign key pointing to the primary key “DeviceID” on table “Device”.

You could design the classes and mapping as follows:

- OGS class “Device” maps to table “Device”.
- Class “Device” has the attributes “SystemLocation”, “SystemContact” and “SysObjectID”, which map to columns with the same names in table “Device”.
- Class “Device” contains a “composite” attribute of type "Class IP".

- Class “IP” maps to table “IPTable”.
- Class “IP” has the attributes “IPAddress” and “IPSubnetMask” mapped to the columns with the same names in table “IPTable”.

The Mapping Schema XML file for this design would contain:

```
<Domain Name="SomeDomainName">
<Application Name="OGS">
<Mappings>
<ClassMap>
  <OGSClass Name="Device"/>
  <ToTable Name="Device">
      <PrimaryKey>
            <Column>
            <ColumnName>DeviceID</ColumnName>
            <JDBC_Type>INTEGER</JDBC_Type>
            <Operator>=</Operator>
            </Column>
      </PrimaryKey>
   </ToTable>

   <HierarchyInfo>
   <HierarchyType>ONE_TABLE_PER_CONCRETE_CLASS</HierarchyType>
   </HierarchyInfo>

   <PropertyMap>
       <OGSAttribute Name="IP"/>
       <AttributeToTableDireCSTMapping>
           <MappedTable Name="IPTable" />
           <PrimaryToMappedAssociation>
                <PrimaryTableColumnNames>
                    <PrimaryTableColumn>
                    <ColumnName>DeviceID</ColumnName>
                    </PrimaryTableColumn>
                </PrimaryTableColumnNames>
                <MappedTableColumnNames>
                    <MappedTableColumn>
                    <ColumnName>DevID</ColumnName>
                    </MappedTableColumn>
                </MappedTableColumnNames>
           </PrimaryToMappedAssociation>
        </AttributeToTableDireCSTMapping>
   </PropertyMap>
</ClassMap>

<ClassMap>
  <OGSClass Name="IP"/>
  <ToTable Name="IPTable">
      <PrimaryKey>
            <Column>
            <ColumnName>DevID</ColumnName>
            <JDBC_Type>INTEGER</JDBC_Type>
            <Operator>=</Operator>
            </Column>
            <Column>
            <ColumnName>IPAddress</ColumnName>
            <JDBC_Type>VARCHAR</JDBC_Type>
            <Operator>=</Operator>
            </Column>
      </PrimaryKey>
   </ToTable>

   <HierarchyInfo>
```

```
   <HierarchyType>ONE_TABLE_PER_CONCRETE_CLASS</HierarchyType>
   </HierarchyInfo>

   <PropertyMap>
      <OGSAttribute Name="IPAddress"/>
      <AttributeToColumnMapping>
               <ColumnName>IPAddress</ColumnName>
               <JDBC_Type>VARCHAR</JDBC_Type>
       </AttributeToColumnMapping>
       <operatorMapGroup ref="StringOperatorMap"/>
   </PropertyMap>

   <PropertyMap>
      <OGSAttribute Name="IPSubnetMask"/>
      <AttributeToColumnMapping>
               <ColumnName>IPSubnetMask</ColumnName>
               <JDBC_Type>VARCHAR</JDBC_Type>
       </AttributeToColumnMapping>
       <operatorMapGroup ref="StringOperatorMap"/>
   </PropertyMap>

</ClassMap>

<group Name="StringOperatorMap">

     <OperatorMap>
          <OGSOperator Name="equals"/>
          <ToOperator>
             <SQLOperator Name="="/>
          </ToOperator>
     </OperatorMap>

     <OperatorMap>
          <OGSOperator Name="contains"/>
          <ToOperator>
             <SQLOperator Name="LIKE"/>
          </ToOperator>
     </OperatorMap>
</group>

</Mappings>
</Application>
</Domain>
```

Note The Generic SQL ASA Rule Evaluator (SQLQueryGenerator) currently supports the kind of attribute mapping shown in this example

Example: Mapping Attributes Across Multiple Tables

You want to have an OGS class “ICS” to represent a chassis containing multiple devices. “ICS” must have an attribute “IPAddress” for each one of its contained SPE devices, so that we can sort on it.

The RDBMS tables are as follows:

- Table “ICS” contains columns with all relevant ICS chassis data.
- Column “ICSID” is the primary key of table “ICS”.
- Table “Device” has columns for all relevant device information, including:
  - An “ICSID” column, which is a foreign key pointing to the primary key of table “ICS”.
  - A “DeviceID” column, which uniquely identifies each SPE device in the chassis.

- Table “IP” contains:
  - “IPAddress” and “Subnet” columns for the various interfaces for each device within a chassis.
  - A “DevID” column, which is a foreign key pointing to the primary key “DeviceID” of table “Device”.

You can design a solution as follows:

- Class “ICS” maps to table “ICS”.
- Class “ICS” has one attribute, “IPAddress”, which is of type "IP Class".
- Attribute “IPAddress” maps to table “IP”. The attribute “IP” contains information on the PK,FK relationship between the tables “ICS”, “Device” and “IP”.

The Mapping Schema XML file for this design is as follows:

```
<Domain Name="SomeDomainName">
<Application Name="OGS">
<Mappings>
<ClassMap>
  <OGSClass Name="Device"/>
  <ToTable Name="Device">
      <PrimaryKey>
            <Column>
            <ColumnName>DeviceID</ColumnName>
            <JDBC_Type>INTEGER</JDBC_Type>
            <Operator>=</Operator>
            </Column>
      </PrimaryKey>
   </ToTable>

   <HierarchyInfo>
   <HierarchyType>ONE_TABLE_PER_CONCRETE_CLASS</HierarchyType>
   </HierarchyInfo>

  <PropertyMap>
       <OGSAttribute Name="IP"/>
       <AttributeToTableIndireCSTMapping>
           <MappedTable Name="IP" />
           <IndirectTable Name="Device" />

           <IndirectToMappedAssociation>
                <IndirectTableColumnNames>
                    <IndirectTableColumn>
                    <ColumnName>DeviceID</ColumnName>
                    </IndirectTableColumn>
                </IndirectTableColumnNames>
                <MappedTableColumnNames>
                     <MappedTableColumn>
                     <ColumnName>DevID</ColumnName>
                     </MappedTableColumn>
                </MappedTableColumnNames>
           </IndirectToMappedAssociation>

           <IndirectToPrimaryAssociation>

                <IndirectTableColumnNames>
                    <IndirectTableColumn>
                    <ColumnName>ICSID</ColumnName>
                    </IndirectTableColumn>
                </IndirectTableColumnNames>
                <PrimaryTableColumnNames>
                    <PrimaryTableColumn>
                    <ColumnName>ICSID</ColumnName>
```

```
                    </PrimaryTableColumn>
                  </PrimaryTableColumnNames>

          </IndirectToPrimaryAssociation>
        </AttributeToTableIndireCSTMapping>
    </PropertyMap>

</ClassMap>

<ClassMap>
  <OGSClass Name="IP"/>
  <ToTable Name="IPTable">
      <PrimaryKey>
            <Column>
            <ColumnName>DevID</ColumnName>
            <JDBC_Type>INTEGER</JDBC_Type>
            <Operator>=</Operator>
            </Column>
            <Column>
            <ColumnName>IPAddress</ColumnName>
            <JDBC_Type>VARCHAR</JDBC_Type>
            <Operator>=</Operator>
            </Column>
      </PrimaryKey>
   </ToTable>

   <HierarchyInfo>
   <HierarchyType>ONE_TABLE_PER_CONCRETE_CLASS</HierarchyType>
   </HierarchyInfo>

   <PropertyMap>
      <OGSAttribute Name="IPAddress"/>
      <AttributeToColumnMapping>
               <ColumnName>IPAddress</ColumnName>
               <JDBC_Type>VARCHAR</JDBC_Type>
       </AttributeToColumnMapping>
       <operatorMapGroup ref="StringOperatorMap"/>
   </PropertyMap>

   <PropertyMap>
      <OGSAttribute Name="IPSubnetMask"/>
      <AttributeToColumnMapping>
               <ColumnName>IPSubnetMask</ColumnName>
               <JDBC_Type>VARCHAR</JDBC_Type>
       </AttributeToColumnMapping>
       <operatorMapGroup ref="StringOperatorMap"/>
   </PropertyMap>

</ClassMap>

<group Name="StringOperatorMap">

     <OperatorMap>
          <OGSOperator Name="equals"/>
          <ToOperator>
             <SQLOperator Name="="/>
          </ToOperator>
     </OperatorMap>

     <OperatorMap>
          <OGSOperator Name="contains"/>
          <ToOperator>
             <SQLOperator Name="LIKE"/>
          </ToOperator>
```

```
    </OperatorMap>
</group>

</Mappings>
</Application>
</Domain>
```

Note The Generic SQL ASA Rule Evaluator (SQLQueryGenerator) currently supports the kind of attribute mapping shown in this example

Customizing the Rule Convertor and Rule Expressions

The default implementation of the Rule Converter is RuleConverterImpl in com.cisco.nm.xms.ogs.asa. This Rule Converter is re-used by most applications.

The Node Rule Expression is modeled by class NodeRuleExpression in com.cisco.nm.xms.ogs.asa. The class CompositeRuleExpression in com.cisco.nm.xms.ogs.asa models the Composite Rule Expression. Extensions of NodeRuleExpression and CompositeRuleExpression created to hold ASA-specific information must be maintained in product-specific directories.

For examples of custom Rule Expressions, see the following Generic SQL ASA implementations:

- SQLNodeRuleExpression in com.cisco.nm.xms.ogs.asa.genericsqlasa.
- SQLCompositeRuleExpression in com.cisco.nm.xms.ogs.asa.genericsqlasa.

Customizing the ASA Change Alerter

ASA Change Alerters are rarely customized. The interface that defines the ASA Change Alerter functions is ASAChangeAlertIf in the class com.cisco.nm.xms.ogs.asa. The default implementation is ASAChangeAlerterBase, also in com.cisco.nm.xms.ogs.asa.

Running a Customized ASA

After you have created your application ASA, you must implement it, to ensure that the OGS Server will instantiate it at runtime.

The following topics explain the tasks necessary to run your customized ASA:

- Registering the ASA with OGS
- Creating the ASA Configuration File
- Example: Using the Generic SQL ASA

Registering the ASA with OGS

To register an ASA with the OGS Server, you must create a registration file for the ASA and specify it in the OGSServer.properties file.

The OGS Server uses the ASA registration file to:

- Instantiate an ASA at startup.
- Assign to this ASA instance the name specified in the registration file.

The ASA registration file's name must appear in the OGSServer.properties file as the value of the `ASARegistrationFile` parameter. You are free to assign this file any unique name, although it is customary to name it as follows:

```
ASARegistrationFile=XXXX-mapping.xml
```

where *XXX* is the application name.

The only real restrictions on the ASA registration file are that its filename must end in the extension "xml" and that it must conform to the following DTD:

```
<?xml version="1.0"?>
<!DOCTYPE ASA_Mappings [
<!ELEMENT ASA_Mappings (Mapping+) >

<!ELEMENT Mapping (OGSSchemaFile+,Location+,ClassName+,Specialization+)>

<!ELEMENT OGSSchemaFile EMPTY>
<!ATTLIST OGSSchemaFile
          name CDATA #REQUIRED >

<!ELEMENT Location EMPTY>
<!ATTLIST Location
          type (Remote|Local) #REQUIRED >

<!ELEMENT ClassName EMPTY>
<!ATTLIST ClassName
          name CDATA #REQUIRED >

<!ELEMENT Specialization EMPTY>
<!ATTLIST Specialization
          name CDATA #REQUIRED >

]>
```

Example

You have created an ASA for one of the applications in Kilner, called "VMSA". You have created an ASA Registration File, called vmsa-mapping.xml, for the VMSA ASA. It has the following content:

```
<?xml version="1.0"?>
<!DOCTYPE ASA_Mappings [
<!ELEMENT ASA_Mappings (Mapping+) >

<!ELEMENT Mapping (OGSSchemaFile+,Location+,ClassName+,Specialization+)>

<!ELEMENT OGSSchemaFile EMPTY>
<!ATTLIST OGSSchemaFile
          name CDATA #REQUIRED >

<!ELEMENT Location EMPTY>
<!ATTLIST Location
          type (Remote|Local) #REQUIRED >

<!ELEMENT ClassName EMPTY>
<!ATTLIST ClassName
          name CDATA #REQUIRED >

<!ELEMENT Specialization EMPTY>
<!ATTLIST Specialization
          name CDATA #REQUIRED >

]>
```

```
<ASA_Mappings>
    <Mapping>
         <OGSSchemaFile name="vms-ogs-schema.xml"/>
         <Location type="Local"/>
         <ClassName name="com.cisco.nm.xms.ogs.asa.ASABaseImpl"/>
         <Specialization name="VMSA"/>
    </Mapping>
</ASA_Mappings>
```

When creating an ASA Registration file, note that:

- You must either include the DTD in the file or reference an external location for the DTD.
- Although you can enter either “local” or “remote” as the “Location type” value, this version of OGS will process only “local”.
- You must specify a Specialization name if you are using ASABaseImpl. OGS will use the Specialization name as a tag to locate the property files and modules used by this instance.

Creating the ASA Configuration File

The ASA configuration file tells a newly instantiated ASA which infrastructure modules to use when validating and evaluating rules. It acts, essentially, as a list of all the customizations performed on the generic ASA implementation modules.

The ASA configuration filename must always:

- Start with the name of your application as you registered it with the OGS Server in the OGSServer.properties file.
- End with the string “_Implementation_Details.properties”.

Example

You have created an ASA for one of the applications in Kilner, called “VMSA”. The ASA configuration file for this application will be named "VMSA_Implementation_Details.properties".

The ASA configuration file content must list:

- All of the ASA infrastructure modules the ASA uses, whether generic or customized. In the VMSA ASA example, we have created custom versions or re-used generic versions of the components shown in Table 30-3.
- The Generic Schema filename. In the VMSA ASA, this file is named “vms-ogs-schema.xml”.
- The Mapping Schema filename. In the VMSA ASA, this file is “vmsa-mapping-schema.xml”.

Table 30-3 VMSA ASA Infrastructure Modules

Component	Class	In	Type
Rule Validator	GenericValidatorImpl	com.cisco.nm.xms.ogs.asa.	Generic
Generic Schema Parser	OGSSchemaParser	com.cisco.nm.xms.ogs.server.parser	Generic
Rule Evaluator	VMSAQueryGenerator	com.cisco.nm.xms.ogs.kilner10.vmsa	Custom
Mapping Schema Parser	VMSAValidator	com.cisco.nm.xms.ogs.kilner10.vmsa	Custom
Rule Converter	RuleConverterImpl	com.cisco.nm.xms.ogs.asa	Generic
Node Rule Expression	VMSANodeRuleExpression	com.cisco.nm.xms.ogs.kilner10.vmsa	Custom

Table 30-3 VMSA ASA Infrastructure Modules (continued)

Component	Class	In	Type
Composite Rule Expression	VMSACompositeRuleExpression	com.cisco.nm.xms.ogs.kilner10.vmsa	Custom
ASA Change Alerter	VmsaChangeAlerter	com.cisco.nm.xms.ogs.kilner10.vmsa	Custom

The ASA Configuration File lists this information using the parameters shown in Table 30-4. The Configuration File is a simple ASCII text file containing name=value pairs. The order in which the parameters appear in the file is not important.

However, you must:

- Use all of the parameter names shown in Table 30-4, exactly as they appear in the table.
- Supply complete values for each parameter.

Table 30-4 ASA Configuration File Parameters

For this component	Use this parameter	And assign to it the
Rule Validator	Rule_Validator=	Classname of your Rule Validator
Generic Schema Parser	Schema_Validator_Class=	Classname of your Generic Schema Parser
Generic Schema File	Generic_Schema_File=	Path and name of the XML schema file used by the Generic Schema Parser.
Rule Evaluator	Rule_Evaluator=	Classname of your Rule Evaluator
Mapping Schema Parser	Class_Mapping_Parser=	Classname of your Mapping Schema Parser.
Mapping Schema File	Mapping_File=	The path and name of the XML schema file used by the Mapping Schema Parser.
Rule Converter	Rule_Converter=	Classname of your Rule Convertor
Node Rule Expression	Node_Expression_Class=	Classname of your Node Rule Expression model.
Composite Rule Expression	Composite_Expression_Class=	Classname of your Composite Rule Expression model.
ASA Change Alerter	Asa_Change_Alerter=	Classname of your ASA Change Alerter.

The VMSA ASA Configuration File might look like this:

```
Rule_Validator = com.cisco.nm.xms.ogs.asa.GenericValidatorImpl
Schema_Validator_Class = com.cisco.nm.xms.ogs.server.parser.OGSSchemaParser
Generic_Schema_File = /vms-ogs-schema.xml
Rule_Evaluator = com.cisco.nm.xms.ogs.kilner10.vmsa.VMSAQueryGenerator
Class_Mapping_Parser = com.cisco.nm.xms.ogs.kilner10.vmsa.VMSAValidator
Mapping_File = /vms-mapping-schema.xml
Rule_Converter = com.cisco.nm.xms.ogs.asa.RuleConverterImpl
Node_Expression_Class = com.cisco.nm.xms.ogs.kilner10.vmsa.VMSANodeRuleExpression
Composite_Expression_Class =
com.cisco.nm.xms.ogs.kilner10.vmsa.VMSACompositeRuleExpression
Asa_Change_Alerter = com.cisco.nm.xms.ogs.kilner10.vmsa.VmsaChangeAlerter
```

Example: Using the Generic SQL ASA

The Generic SQL ASA is a customization of the ASA infrastructure for querying a relational database using OGS. It uses the generic or customized components shown in Table 30-5.

Table 30-5 Generic SQL ASA Components

Component	Class	Status
Rule Validator	GenericValidatorImp in com.cisco.nm.xms.ogs.asa	Generic
Generic Schema Parser	OGSSchemaParser in com.cisco.nm.xms.ogs.server.parser.	Generic
Rule Evaluator	SQLQueryGenerator in com.cisco.nm.xms.ogs.asa.genericsqlasa.	Custom
Rule Converter	RuleConverterImpl in com.cisco.nm.xms.ogs.asa.	Generic
Mapping Parser	RDBMSSchemaValidator in com.cisco.nm.xms.ogs.asa.genericsqlasa.	Custom
Node Expression	SQLNodeRuleExpression in com.cisco.nm.xms.ogs.asa.genericsqlasa.	Custom
Composite Expression	SQLCompositeRuleExpression in com.cisco.nm.xms.ogs.asa.genericsqlasa.	Custom

Creating an OGS GUI

The UII (User Interface Initiative) Object Selector provides a convenient and versatile tree-display component that gives users GUI access to OGS group data. When implemented, it provides users with:

- A group hierarchy display.
- A means of selecting objects within groups, including both single and multiple selection.

Object Selector is part of the UII release package available at the User Experience web site, which is available to Cisco employees at http://picasso (be sure to use version 6.1 or later of the UII release package).

Object Selector comes in standard “Content Area” and space-conserving “Sliding” versions, and has a complete set of API functions. If you plan on using Object Selector in your application, please remember:

- Integration between Object Selector and OGS is application-specific. You should refer to the *SDK Developer’s Guide for UII (*the SDK for UII Release 6.1 is available as EDCS-348564) or the online help version of the same document at http://picasso, for details on programming Object Selector to work with your application.
- As part of your integration effort, you will need to develop an Object Selector Tree Generator which retrieves data from OGSServerProxy and converts it for display in Object Selector. This Tree Generator is part of the application deliverable, and must be maintained by you.
- OGS supplies a default Object Selector Tree Generator for use by the OGS Administration GUI. This default Tree Generator can serve as a model for a custom Tree Generator for your application:
  - The default Tree generator is ObjectSelectorTreeGenerator in com.cisco.nm.xms.ogs.client.ostaglib.util.

  - For source code, set up an ogs1_1 view in ClearCase, then view: /vob/enm_ogs/share/classes/client/com/cisco/nm/xms/ogs/client/ostaglib/util/ObjectSelectorTreeGenerator.java.

- If you plan on using Object Selector with an OGS implementation that includes Secure Views, see the "Using Secure Views with Object Selector" section on page 30-46

To use the Object Selector in your application:

Step 1 Install the latest UII package following the instructions in the UII SDK.

Step 2 Copy the following directories and files from the UII package to the corresponding directories of your project, under the Tomcat webapps/YOUR_APPLICATION directory.

- /objectselector
- /WEB-INF/lib/ogs-objselector.jar
- /WEB-INF/tlds/ogs-objectselector-taglib.tld

Step 3 Insert the following lines into the *action-mappings* section of your project's /WEB-INF/struts-config.xml file.

```
<action path="/objselcaching"
type="com.cisco.nm.xms.ogs.client.uii.ObjectSelectorCachingAction"/>
<action path="/objselsliding"
type="com.cisco.nm.xms.ogs.client.uii.ObjectSelectorSlidingPanelAction"/>
```

Step 4 Using the Dreamweaver Extension manager, install the included Dreamweaver Extension file, OGS-OS-DWX.mxp.

Step 5 Drag and drop into your JSP file the icon corresponding to the type of Object Selector you want to create. In order to preserve the selection, put it into the current forms[0] (within the first <uii:form> tag of the page).

If you are not using Dreamweaver, insert directly into the corresponding JSP file the code appropriate for the type of Object Selector you want to create:

- For a Sliding Object Selector, insert this code:

```
<%@ taglib uri="/WEB-INF/tlds/ogs-objectselector-taglib.tld" prefix="ogs" %>
<ogs:slidingSelector
osName="YOUR_OBJECT_SELECTOR_NAME"
selectionMode="multiple"
preserveSelection="false"
isGroupSelector="false"
mixSelection="false"
showFilter="true"
showQuickFilter="true"
showSaveBtn="true"
treeGenerator="com.cisco.nm.xms.ogs.test.TestTreeGenerator"/>
```

- For a Content Area Object Selector, insert this code:

```
<%@ taglib uri="/WEB-INF/tlds/ogs-objectselector-taglib.tld" prefix="ogs" %>
<ogs:contentAreaSelector
osName="YOUR_OBJECT_SELECTOR_NAME"
selectionMode="multiple"
preserveSelection="false"
isGroupSelector="false"
mixSelection="false"
width="200"
height="300"
treeGenerator="com.cisco.nm.xms.ogs.test.TestTreeGenerator"/>
```

- Once you have inserted the code, set the attributes in the JSP tags to match your requirements.

Step 6 Implement the JavaScript functions as needed. For details on the available functions, see the UII SDK.

Step 7 In the corresponding Action class, which handles the form containing the object selector, instantiate as the first object in the "perform" method an instance of the class OGSObjectSelectorAdapter (in com.cisco.nm.xms.ogs.client.uii).

Step 8 Implement a custom Tree Generator class, as follows:

a. Create a class extended from BasicObjectSelectorTreeGenerator in com.cisco.nm.xms.ogs.client.ostaglib.util.

b. Set this class name in the JSP tag attribute *treeGenerator*.

If you do not do this, the Object Selector will call the default tree generator class, ObjectSelectorTreeGenerator in com.cisco.nm.xms.ogs.client.ostaglib.util. This class also extends from BasicObjectSelectorTreeGenerator and is provided by the application.

For reference information on the Tree Generator, see the UII SDK.

c. In this tree generator class, implement the following methods:

```
getTreeEntries REQUIRED
getUserFilterEntries OPTIONAL (For sliding object selector only).
getAppletParameterImages OPTIONAL
getAppletParameterImageFile OPTIONAL
getAppletParameterToolTipFile OPTIONAL
getAppletParameterToolTipLineContinue OPTIONAL
getAppletParameterToolTipWidth OPTIONAL
getAppletParameterBkgColor OPTIONAL
getAppletParameterBkgColorSelect OPTIONAL
getAppletParameterBkgColorHover OPTIONAL
getAppletParameterTabOpenColor OPTIONAL
getAppletParameterTabCloseColor OPTIONAL
getAppletParameterFontName OPTIONAL
getAppletParameterFontSize OPTIONAL
getAppletParameterPadding OPTIONAL
getAppletParameterOnSelect OPTIONAL
getAppletParameterOnUnSelect OPTIONAL
getAppletParameterOnCachingExpand OPTIONAL
getAppletParameterSelectAllChild OPTIONAL
getAppletParameterUnSelectAllChild OPTIONAL
getAppletParameterSelectAncestors OPTIONAL
getAppletParameterUnSelectAncestors OPTIONAL
getAppletParameterTabs OPTIONAL
getAppletParameterCheckWhenLabelClick OPTIONAL
getAppletParameterOnHighLight OPTIONAL
getAppletParameterDisable OPTIONAL
getAppletParameterSort OPTIONAL
getAppletParameterXml OPTIONAL
getAppletParameterXmlSource OPTIONAL
getAppletParameterDeDuplicate OPTIONAL
getAppletParameterNumberOfSelectedLeaves OPTIONAL
```

For more information about these methods, see the UII SDK.

Using OGS Secure Views

Starting with CWCS 3.0, OGS supports Secure Views. OGS Secure Views allow your application to control user access to individual members of a group, instead of just to groups as a whole.

Using the user ID, application ID, and task ID (and optionally, the session ID) of the requesting client, and an interface to the CAM-maintained ACS Server security system, Secure Views can return a subset of the total membership of a group. This subset will contain only those devices to which the user has access.

OGS Secure Views are explained in the following topics:

How Secure Views Work

OGS Secure Views filter the memberships of a group based on the IDs of:

- The user requesting access to a group's members.
- The application the user is using to access the group's members.
- The task the user intends to perform on the group's members.

The basic process flow proceeds as follows:

1. The OGSClient gets a complete list of objects for the requested group from the OGS Server.
2. The OGSClient gets from CAM the list of devices that the user is authorized to use for the given task, application and (optionally) session ID.
3. The OGSClient performs an intersection of the two lists:
   a. It ignores non-device objects, passing them through to the OGSObjectList.
   b. It filters out of the OGSObjectList all device objects that do not also exist in the CAM list.
   c. The OGSClient returns to the application GUI an OGSObjectList containing all non-device objects and only filtered device objects.

OGS Secure Views accomplishes this using the following classes and methods new in CWCS 3.0:

- The basic interface class OGSFilterIf, in com.cisco.nm.xms.ogs.client.

  The default implementation is OGSFilterImpl in the same package. OGSFilterImpl looks up the value of the "DeviceClasses" property in OGSClient.properties, uses it to identify all of the device classes that need to be filtered, and checks in the CAM-returned list for objects of those classes.

  Only when the class of object in the CAM list matches the "DeviceClasses" value will the object be filtered. Applications using SQLASA need to use the OGSSQLFilterImpl implementation.
- OGSManagementFormAction class in com.cisco.nm.xms.ogs.client.mgmt
- OGSServerProxy in com.cisco.nm.xms.ogs.client. OGSServerProxy detects if the client requesting group or rule evaluation is an OGSAdminClient. It does this by looking for the Application ID "OGS" and Task ID "OGSOPERATION" in OGSSecurityContext.

  For details, see the "Using Secure Views With the OGS Administrative GUI" section on page 30-47).

- Class OGSSecurityContext, which defines a constructor that accepts the user ID, application ID, task ID and session ID (which can be null), that CAM requires to perform device filtering.

  For details, see the "Using OGS SecurityContext" section on page 30-45

The design of OGS Secure Views imposes several requirements on applications using it:

- While all the filtering of the membership list is actually performed on the client, Secure Views depends entirely on the CWCS CAM security interface to get the list of group members for which the user is authorized.

  It therefore requires that your application operate in ACS (TACACS+) mode, not in CMF security mode. If your application operates in CMF mode, the default implementation of Secure Views will not perform any filtering at all. In this case, evaluating a group or rule will result in all members of that group being returned.

- You must always specify the OGS classes your application uses to represent devices. You do this using the "DeviceClasses" property in the OGS client's OGSClient.properties file.

  For example: An OGSClient.properties entry of `DeviceClasses="CMF:DCR:Device"` specifies that objects of the class "CMF:DCR:Device", and only this class, represent devices. Similarly, the entry `DeviceClasses="CMF:DCR:Device","CMF:DCR:Interface"` specifies that both these classes represent devices.

  If you do not do this, your Secure Views implementation will pass all device objects through to the client, regardless of the user's authorizations (just as it does with non-device objects).

- You should use the unique DCR device IDs supplied by the Device Credentials Repository Server available starting with CWCS 3.0.

  For more information about DCR, see Chapter 14, "Using the Device Credentials Repository"). The simplest way to do this is to ensure that:

  - A DCR Server is running.
  - The application data source has been populated with DCR IDs.
  - The application OGS ASA you use to evaluate rules relating to device classes uses the DCR ID in the "value" parameter of the Object ID of every object that represents a device.

    For example: If you are using the class `CMF:DCR:Device` to represent devices, and one of these devices has the DCR ID `123456`, your ASA must return the Object ID for this device as `CMF:DCR:Device$123456`.

    If you do not do this, your Secure Views implementation will be unable to match any of entries in the CAM-returned list of authorized objects (which is a list of DCR IDs) against the OGSObjectList, and will not return any of them to the client (even though the user may be authorized to see them).

    Note that you can work around the requirement of an ASA that preserves DCR IDs, as long as you can supply these IDs in another way (see the "Using Secure Views Without DCR IDs" section on page 30-48).

- You must ensure that the running instance of OGSServerProxy performing the filtering is running in the same Java VM as the CAM instance with which it communicates.

  This restriction is imposed by the current implementation of CAM, which by default expects all of its clients to be servlets executing in the same servlet engine. CAM can be packaged with the process using OGSServerProxy.

- The default implementation of the filtering interface allows unrestricted access to members that do not represent devices. If your application needs additional filtering on non-device group members, you can either extend the default implementation or create a new implementation of the interface to meet your requirements. Be sure to indicate that you are using a new implementation, as explained in the "Specifying a Non-Default Implementation" section on page 30-48.

Implementing Secure Views

Secure Views is intended for use in applications where:

- ACS mode is enabled.
- You intend to filter access to devices only.
- Devices are identified using device IDs supplied by the Device Credentials Repository (DCR IDs).

As long as your implementation observes the requirements explained in the "How Secure Views Work" section on page 30-43, using Secure Views is relatively simple, as explained in the following topics:

- Installing Secure Views
- Using OGS SecurityContext
- Using Secure Views With DCR IDs
- Using Secure Views with Object Selector
- Using Secure Views With the OGS Administrative GUI

The requirement for ACS mode is an absolute requirement. If your application cannot observe the requirements for device-only filtering and use of DCR IDs for devices, see the "Customizing Your Secure Views Implementation" section on page 30-47

Installing Secure Views

To implement Secure Views, OGS 1.1 must be packaged with your application. You will need to extract the file ogs1.1.war from the OGS SDK, and to include the following packages in your build:

- com.cisco.nm.xms.ogs.client
- com.cisco.nm.xms.ogs.util
- com.cisco.nm.xms.ogs.server

Using OGS SecurityContext

The OGS interface specifies filtering of a group's membership based on the user/application/task context in which a request is made. Additionally, class OGSSecurityContext defines a constructor that accepts the user ID, application ID, and task ID (and, optionally, the session ID) that CAM requires to identify the devices for which the user has access.

If your application requires Secure Views, be sure that you are passing the proper information to this constructor before you make the request for group evaluation. If you do not do this, CAM will not have a chance to return the list of authorized devices before group evaluation is conducted.

The normal process for creating and using instances of OGSSecurityContext is:

Step 1 Create an instance of the class TaskID (in package com.cisco.nm.xms.ogs.util) and assign the proper values to its attributes TaskID, ApplicationID, and SessionID (the SessionID can be null).

Step 2 Create an instance of OGSSecurityContext (in package com.cisco.nm.xms.ogs.util).

Step 3 Pass the instance of the TaskID class to the OGSSecurityContext constructor.

Step 4 Issue your group- or rule-evaluation request.

Using Secure Views With DCR IDs

If you are using standard Object IDs that include the standard DCR ID for all group members that are devices, the reference OGSFilterImpl class and its DoFiltering method perform all of the following tasks automatically:

```
DoFiltering()
{
    Check if CiscoWorks is in ACS Mode;
    If CiscoWorks is not in ACS Mode, return OGSObjectList without filtering;
    If CiscoWorks is in ACS Mode;
        Get Admin Groups from CAM.
        Get Admin Device Groups that User is authorized to use for the given TaskID and
            ApplicationID
        Get the DCR ID of the devices in these Admin Device Groups;
        Avoid Filtering all non-device objects, using the property "DeviceClasses" in
            OGSClient.properties
        Get the Device Objects in the OGSObjectList passed to the function as a parameter;
        Get the Value portion of the Device Object. This is assumed to be the DCR ID;
        Intersect the list and return an OGSObjectList containing all non-device objects
            and only filtered device objects
```

Using Secure Views with Object Selector

As explained in the "Creating an OGS GUI" section on page 30-40, integrating OGS with an Object Selector GUI is fairly simple. If your OGS implementation also includes Secure Views, however, you have a few extra tasks to perform:

- In the Action Class, which handles the form containing the Object Selector, your application must pass in the appropriate Application ID, Task ID, and Object Selector Name.
- Your application Tree Generator must retrieve the Task ID attribute and use it to create the OGSSecurityContext instance for the request.

  As a reference, the default Tree Generator uses method getOGSUserContext to perform all the steps mentioned in the "Using OGS SecurityContext" section on page 30-45. This includes:

  - Creating an instance of the class TaskID with all the proper attribute values
  - Creating an instance of OGSSecurityContext
  - Passing the TaskID instance to the OGSSecurityContext constructor

To adapt your application Object Selector for use with Secure Views:

Step 1 Set the Application ID into the HTTP Session.

Step 2 Get an instance of OGSObjectSelectorAdapter.

Step 3 Use the setObjectTaskIdList method to set the Task ID to the Task ID on which you want to filter.

For example: Let us assume that your:

- Object Selector Name is `FH.OGScontent` (this is the same as the osname specified in the objectselector JSP tag)
- Application ID is `iptm`.
- Task ID is `write_priv`.

Using these names, you would modify your Object Selector Action Class as follows:

```
public ActionForward perform (ActionMapping mapping)
                    ActionForm form
                    HttpServletRequest request
                    HttpServletResponse response)
                    throws IOException ServletException
{
              String YOUR-OBJECT-SELECTOR-NAME=”FH.OGScontent”
              String YOUR-APPLICATION-ID=”iptm”
              String YOUR-TASK-ID=”write-priv”
              HttpSession session=request.getSession();
              session.setAttribute("AppID"+YOUR-OBJECT-SELECTOR-NAME, YOUR-APPLICATION-ID);
              OGSObjectSelectorAdapter osAdapter=new
                          OGSObjectSelectorAdapter(request,"FH.OGScontent");
              osAdapter.setObjectTaskIdList("write_priv");
              ...
```

Using Secure Views With the OGS Administrative GUI

As long as you have observed the limits described in the “How Secure Views Work” section on page 30-43, your Secure Views implementation class will filter all device requests from all clients by default.

This can be a problem if your application uses the OGS Administrative GUI to allow users with sufficient authority to define groups. The OGS Client performing the filtering must be able to distinguish between normal client requests and those from an Administrative GUI, so that the Administrative GUI user has access to all devices when defining groups and their rules.

To distinguish them, make sure that the OGS Administrative GUI you have implemented in your application identifies itself to CAM with an Application ID of “OGS” and a TaskID of “OGSOPERATION” when issuing requests (these two IDs are case-sensitive and must be entered exactly as shown here).

These two values, when passed with OGSSecurityContext, turn off filtering functions in Secure Views implementations automatically.

If your application is used in a CiscoWorks suite, you may want to consider whether you want to implement the OGS Administrative GUI in your application at all.

CWCS 3.0 supplies the OGS Administrative GUI at the CiscoWorks level, with filtering turned off by default (that is, the CiscoWorks implementation of the OGS Administrative GUI already identifies itself with the Application ID “OGS” and the Task ID “OGSOPERATION”).

If you must implement a separate OGS Administrative GUI for your application, you will probably want to keep filtering turned on.

Customizing Your Secure Views Implementation

It is possible to adapt Secure Views to some special requirements, including:

- Specifying a Non-Default Implementation
- Using Secure Views Without DCR IDs

Specifying a Non-Default Implementation

In the default OGS packaging, the OGSClientProperties parameter SecurityFilterImpl= is explicitly set to OGSFilterImpl. This is the default or reference implementation of OGSFilterIf in com.cisco.nm.xms.ogs.client.

If you must handle special filtering requirements, you must either override the OGSFilterImpl implementation's DoFilter () function or create a new class which implements OGSFilterIf (you are likely to do the latter if you are using Secure Views without DCR IDs, as explained in the "Using Secure Views Without DCR IDs" section on page 30-48.)

If you have created a new implementation of OGSFilterIf, you must set SecurityFilterImpl to the name of your new class. For example: `SecurityFilterImpl=MyOGSFilterImpl`.

Using Secure Views Without DCR IDs

If your application does not use the DCR ID as the value of the Object ID for devices, you will need to create a new implementation of OGSFilterIf or extension of OGSFilterImpl that maps between whatever you are using in the Value portion of a Device Object and the actual DCR IDs.

Example

If you are using a Device Name for the Value portions of the Object ID for devices, you will need to create a new class with a DoFiltering() method that implements OGSFilterIf. You can do this mapping in any way that is convenient to you, but it must be performed before OGSServerProxy attempts to compare the CAM and OGS object lists:

```
DoFiltering()

{

    Check if CiscoWorks is in ACS Mode;

    If CiscoWorks is not in ACS Mode, return OGSObjectList without filtering;

    If CiscoWorks is in ACS Mode;

        Get Admin Device Groups from CAM.

        Get Admin Device Groups that User is authorized to use for the given TaskID and
            ApplicationID

        Get the DCR ID of the devices in these Admin Device Groups;

       Avoid filtering all non-device objects in the OGSObjectList returned by OGS Server

       Get the Device Objects in the OGSObjectList passed to the function as a parameter;

      Get the Value portion of the Device Object. This will be something other than DCRID;

    Map the OGS Value to DCRID and return the ObjectList with DCR IDs in Value position

       Intersect the list and return an OGSObjectList containing all non-device objects
            and only filtered device objects
```

Using OGS Common and Shared Groups

OGS Common Groups are groups created based on DCR attributes. They are propagated to other OGS Servers running on the same server and to peer servers in the same DCR cluster.

For example, the CWCS Common Groups will be propagated to the RME OGS Server, if RME is installed, so users of RME OGS Server will also be able the view these Common Groups.

OGS Shared Groups are application groups propagated to CWCS and other applications on the same server and to peer servers in the same DCR cluster.

For example, all Campus Manager groups are propagated to the RME OGS Server on the same server.

Common Groups have enumerated rules. Their membership is cached in a way similar to caching of a local group. Shared Groups are remote links to the respective OGSServer and are evaluated at runtime. Membership is not cached locally for Shared Groups, so group-membership change events are not sent for them.

A Provider Group refers to the root group name under each hierarchy.

For example, the Provider Group for a Common Services group hierarchy is CS@*hostname*, and for the RME hierarchy, it is RME@*hostname*. The hostname is appended to make the Provider Group unique, as Shared Groups are propagated across servers .

By default, the Provider Group name is of the format *application@hostname*. If the hostname changes, or the admin user configures the CWHP server name as the Provider Group suffix, the Provider Group name will be changed after a daemon restart.

For examples and User Interface of Common Groups and Shared Groups, see Chapter 5, Administering Groups, in User Guide for CiscoWorks Common Services.

Using Common and Shared Groups requires you to perform the tasks covered in the following topics:

- Configuring OGSServer.properties
- Configuring SharedGroups.properties
- Implementing the SharedGroupObjectMapperIf Interface

Configuring OGSServer.properties

To enable Common and Shared Groups, applications should configure the following in OGSServer.properties:

```
# Application OGS instances that do not wish to participate
# in group sharing should remove the following property.
SharedGroupImplClass = com.cisco.nm.xms.ogs.sharedgroups.SharedGroupManager
```

Configuring SharedGroups.properties

The following shows a generic SharedGroups.properties file, with explanation of the fields and how to set them.

```
# Uncomment this line and specify a different value if your OGS
# publishes its own URN
#LocalOgsURN = ogs_server_urn^M
#
# Application OGS instances should uncomment the following line
#LocalSgURN = remote_ogs_urn
#
```

```
# Uncomment the following line and modify to match the URL of the CTMServlet
# deployed by your application
#LocalURL = :443/<application webapp>/CTMServlet
#
SrcDeviceClass = :CMF:DCR:Device
#
# Application OGS instances should specify their device class here
#
DestDeviceClass = <Application Device class>
#
# The following should match the value of the property ProductId
# of CMF-OGS
#
RemoteOgsProductId = CS
#
# Uncomment the following property and specify the desired
# depth of the MDF hierarchy if you wish to limit the depth
# of the MDF hierarchy in common groups
#
#MDFHierarchyDepth = -1
ClientRegistryFile = clients.reg
#Application can uncomment the following to override the default rule for provider group
from CS hierarchy
#UseContainerForCommonGroups=true
```

Implementing the SharedGroupObjectMapperIf Interface

To map application-specific object IDs to a common representation, applications should implement SharedGroupObjectMapperIf.

If your application uses the DCR device ID as the application object ID for OGS objects, you need only configure the SrcDeviceClass and DestDeviceClass in the SharedGroups.properties file. The default implementation (com.cisco.nm.xms.ogs.sharedgroups.DefaultObjectMapper) will handle the object mapping.

If you do not use DCR device IDs as application object IDs, you must implement SharedGroupObjectMapperIf and provide the implementation in the SharedGroups.properties, as follows:

```
#Object ID mapper class
ObjectIdMapClass = Customized object mapper class
```

The customized object mapper class should be in the classpath for the OGSServer.

OGS Utility Class for Common and Shared Groups

The class com.cisco.nm.ogs.util.OGSUtil provides two APIs to get the common and local provider group name.

```
public static String getCommonRoot(); //for getting the CS root group
```
public static String getLocalRoot(); //for getting the local root group

Using OGS 1.3 Client Side Enhancements

Applications using OGS 1.3 have enhanced control over the display of GUI components. These client-side enhancements, which are not available with OGS 1.2, enable applications to determine:

- The display of some GUI components based on group types.
- The flow sequence of UII wizards based on group types.

Changes needed to support these client-side enhancements are available in the com.cisco.nm.xms.ogs.client package of the OGS module.

The following topics explain how to use these enhancements:

- About the Enhanced OGS 1.3 Classes and Data Structures
- Controlling the Display of Wizard Steps
- Integrating OGS 1.3 With Your Application

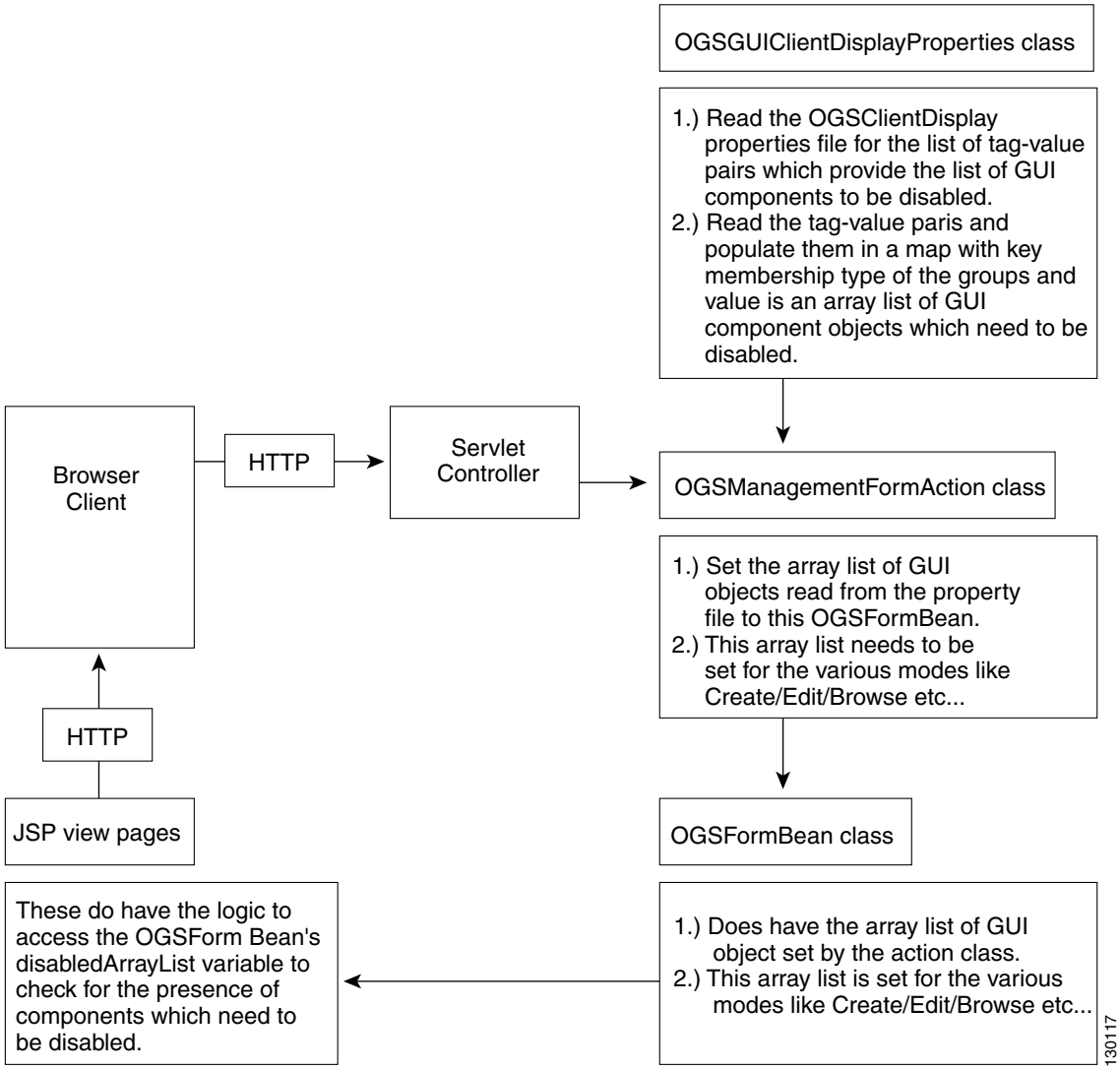
About the Enhanced OGS 1.3 Classes and Data Structures

Table 30-6 shows the OGS 1.3 classes enhanced to allow for GUI controls. Figure 30-4 shows how these classes interact with each other and with the GUI components.

Table 30-6 Classes for Client Side Enhancements

Class	Description
OGSClientUIDisplayProperties	Reads the tag-value pairs from OGSClientUIDisplay.properties, and stores the values in the hash map.
OGSFormBean	Updated to store the Array List of GUI components which need to be disabled. Corresponding getter and setter methods are provided.
OGSManagementFormAction	Updated with logic permitting the setting of the Array List of GUI components which have to be disabled. Based on the membership type of the group entries, the Form Bean is updated with the Array List of GUI objects. Control of the display of the GUI components can be made specific to operational modes like Create, Edit and Browse. You can use the new screen IDs to skip the membership page.

Figure 30-4 OGS 1.3 Class and GUI Component Interactions



The following JSP files were also modified to check for the availability of the GUI components in the disabled Array List:

- ogsBrowseProperties.jsp
- ogsCreateProperties.jsp
- ogsCreateRules.jsp
- ogsEditProperties.jsp
- ogsEditRules.jsp

Based on the presence of the components in the disabled Array List, the display of the view components can be controlled.

Controlling the Display of Wizard Steps

You need to create new screen IDs for controlling the display of wizard steps while in Create or Edit mode. The new screen IDs need to be defined in the site-map XML files of the respective web application. The screen IDs are:

- ogsSkipMemberCreateProperties
- ogsSkipMemberCreateRules
- ogsSkipMemberCreateSummary
- ogsSkipMemberEditProperties
- ogsSkipMemberEditRules
- ogsSkipMemberEditSummary

Configure the screen IDs to skip the Membership page during the Create and Edit wizard tasks of the group entries. However, support is not provided for skipping Rules, Properties and Summary pages, as these are required for group handling.

Note Source code is available in /vob/enm_ogs/share/classes/client/com/cisco/nm/ms/ogs/client, under the ogs1_3 view.

Integrating OGS 1.3 With Your Application

To integrate OGS 1.3's enhanced client-side features with your application:

Step 1 Include all OGS 1.3-related jar files with the application.

Although all the changes were made in the ogs-client1.3.jar, we suggest that you refresh all jar files, to ensure that you have uniform versions throughout.

Step 2 Update the OGSClientUIDisplay.properties file with the requisite entries.

For a complete set of entries related to UI components, which you can configure, see Example 30-2 on page 30-55. This file can be placed under the WEB-INF/classes directory of the respective web application, under Tomcat.

Step 3 Define the TAGSTABLE definition in the XML file representing the Group Entries. Similarly, define the PROPERTYTABLE with the PROPERTY KEY as MEMBERSHIP_TYPE and the desired value.

For an example system-groups.xml file, see Example 30-3 on page 30-56. The file is currently available under the WEB-INF/classes directory of the respective web-app under Tomcat servlet.

Step 4 Make sure the string value provided for the MEMBERSHIP_TYPE key matches the value provided in the tag of the OGSClientUIDisplay.properties.

If the XML file has entries like these for representing Cisco Access Servers:

```
<GROUP>
    <NAME>/System Defined Groups/Cisco Access Servers</NAME>
    <DESCRIPTION>Group for Cisco Access Servers</DESCRIPTION>
    <TYPE value="DYNAMIC" />
    <OGS_RULE_CONTAINER>
        <OGSRULE> </OGSRULE>
    </OGS_RULE_CONTAINER>
    <TAGSTABLE  ref="devicegrouptags" />
    <READ_PERMISSION_LIST ref="read_user_list" />
```

```
    <WRITE_PERMISSION_LIST ref="write_user_list" />
    <EVALUATE_PERMISSION_LIST ref="evaluate_user_list" />
</GROUP>
<PROPERTYTABLE name="devicegrouptags">
    <PROPERTY>
        <KEY>MEMBERSHIP_TYPE</KEY>
        <VALUE>DEVICE</VALUE>
    </PROPERTY>
</PROPERTYTABLE>
```

Then the OGSClientUIDisplay.properties file can have entries like:

```
DEVICE.privateVisibility=disabled
```

The string “DEVICE” in the properties file needs to match the string defined in the XML file. However applications can use any convenient string that is relevant in their application space.

Step 5 Define the entries releated to the new screen in the struts-config.xml and site-map XMLfiles for the respective webapp.

The entries in struts-config.xml are:

```
forward name="ogsSkipMemberCreateProperties"
path="/WEB-INF/screens/uii/index.jsp?sid=ogsSkipMemberCreateProperties" />
             <forward name="ogsSkipMemberCreateRules"
path="/WEB-INF/screens/uii/index.jsp?sid=ogsSkipMemberCreateRules" />
             <forward name="ogsSkipMemberCreateSummary"
path="/WEB-INF/screens/uii/index.jsp?sid=ogsSkipMemberCreateSummary" />
<forward name="ogsSkipMemberEditProperties"
path="/WEB-INF/screens/uii/index.jsp?sid=ogsSkipMemberEditProperties" />
             <forward name="ogsSkipMemberEditRules"
path="/WEB-INF/screens/uii/index.jsp?sid=ogsSkipMemberEditRules" />
             <forward name="ogsSkipMemberEditSummary"
path="/WEB-INF/screens/uii/index.jsp?sid=ogsSkipMemberEditSummary" />
```

The entries in site-map.xml are:

```
<appWizard>
            <appWizardItem>
              <label>Properties</label>
              <appContentArea screenID="ogsSkipMemberCreateProperties"
             contentAreaTitle="Properties"
            helpTag="group_administration"
             fileRef="/WEB-INF/screens/OGS/ogsCreateProperties.jsp" />
            </appWizardItem>
            <appWizardItem>
              <label>Rules</label>
              <appContentArea screenID="ogsSkipMemberCreateRules"
             contentAreaTitle="Rules"
            helpTag="group_administration"
             fileRef="/WEB-INF/screens/OGS/ogsCreateRules.jsp" />
            </appWizardItem>
            <appWizardItem>
              <label>Summary</label>
              <appContentArea screenID="ogsSkipMemberCreateSummary"
             contentAreaTitle="Summary"
            helpTag="group_administration"
             fileRef="/WEB-INF/screens/OGS/ogsCreateSummary.jsp" />
            </appWizardItem>
          </appWizard>

<appWizard>
            <appWizardItem>
              <label>Properties</label>
```

```
    <appContentArea screenID="ogsSkipMemberEditProperties"
   contentAreaTitle="Properties"
  helpTag="group_administration"
   fileRef="/WEB-INF/screens/OGS/ogsEditProperties.jsp" />
  </appWizardItem>
  <appWizardItem>
    <label>Rules</label>
    <appContentArea screenID="ogsSkipMemberEditRules"
   contentAreaTitle="Rules"
  helpTag="group_administration"
   fileRef="/WEB-INF/screens/OGS/ogsEditRules.jsp" />
  </appWizardItem>
  <appWizardItem>
    <label>Summary</label>
    <appContentArea screenID="ogsSkipMemberEditSummary"
   contentAreaTitle="Summary"
  helpTag="group_administration"
   fileRef="/WEB-INF/screens/OGS/ogsEditSummary.jsp" />
  </appWizardItem>
</appWizard>
```

Example 30-2 OGSClientUIDisplay.properties

```
# Properties file for the OGS Client UI display
# This file is mainly used to control the display of UI components based on group types
# and is valid only for OGS Admin GUI.
# It is assumed that all the OGS GUI components are enabled by default.
# The respective components along with the group types for which they need to be disabled
# can be mentioned here.
# ----------------------------------
# Copyright (c) 2004 Cisco Systems, Inc.
# All rights reserved
#
# THIS SOFTWARE IS THE PROPERTY OF CISCO SYSTEMS INC.
# 2004
# THE RECIPIENT BY ACCEPTING THIS MATERIAL AGREES THAT THE
# MATERIAL WILL NOT BE USED, COPIED OR REPRODUCED IN  WHOLE OR
# IN PART NOR ITS CONTENTS REVEALED  IN ANY  MANNER WITHOUT THE
# EXPRESS WRITTEN PERMISSION OF CISCO SYSTEMS, INC.
# ----------------------------------
#
# The names which refer to the various components are mentioned below.
# creategroupSelect -> Group Select button in Properties:Create page
# createparentChange -> Change Parent button in Properties:Create page
# DynamicEvaluation -> Radio button for Automatic
# StaticEvaluation -> Radio button for Based Upon User Request
# publicVisibility ->Visibility scope radio button "Available to all users"
# privateVisibility -> Visibility scope radio button "Available to created user only"
# createORoperator ->  OR operator in the Rules:Create page
# createANDoperator -> AND operator in the Rules:Create page
# createEXCLUDEoperator -> EXCLUDE operator in the Rules:Create page
# createMembershipWizard ->Disabling the Membership page step in the Creation Wizard
# editORoperator -> OR operator in the Rules:Edit page
# editANDoperator -> AND operator in the Rules:Edit page
# editEXCLUDEoperator -> EXCLUDE operator in the Rules:Edit page
# editMembershipWizard ->Refers to the Membership page step in the Edition Wizard
# browseViewParentRule -> View Parent Rule button related to Properties:Details page
# browseMembershipDetails - > Membership Details button related to Properties:Details page
```

```
# EditGroupName -> Text field related to the Group Name in the Edit:Properties page
# The KEY value contains a string separated by "."
# The first token of the key should match the value given for the MEMBERSHIP_TYPE in the
# PROPERTY_TABLE pertaining to XML file containing the group definition.
# Sample examples are given below:
#
#DEVICE.createGroupSelect=disabled
#DEVICE.createParentChange=disabled
DEVICE.DynamicEvaluation=disabled
#DEVICE.StaticEvaluation=disabled
DEVICE.privateVisibility=disabled
DEVICE.publicVisibility=disabled
DEVICE.createAddRuleExpression=disabled
DEVICE.createSyntaxCheck=disabled
DEVICE.createViewParentRule=disabled
DEVICE.createORoperator=disabled
DEVICE.createANDoperator=disabled
DEVICE.createEXCLUDEoperator=disabled
DEVICE.editORoperator=disabled
DEVICE.editANDoperator=disabled
DEVICE.editEXCLUDEoperator=disabled
DEVICE.editAddRuleExpression=disabled
DEVICE.editSyntaxCheck=disabled
DEVICE.editViewParentRule=disabled
DEVICE.createMembershipWizard=disabled
DEVICE.editMembershipWizard=disabled
INTERFACE.createGroupSelect=disabled
INTERFACE.createParentChange=disabled
```

Example 30-3 System-Groups.xml

```
<?xml version="1.0"?>
<!DOCTYPE SYSTEMGROUPS [
<!ELEMENT SYSTEMGROUPS (GROUP+,PROPERTYTABLE+,USERLIST+) >
<!ELEMENT GROUP
(NAME,DESCRIPTION?,TYPE,OGS_RULE_CONTAINER,TAGSTABLE?,READ_PERMISSION_LIST,WRITE_PERMISSIO
N_LIST,EVALUATE_PERMISSION_LIST)>
<!ELEMENT NAME (#PCDATA)>
<!ELEMENT DESCRIPTION (#PCDATA)>
<!ELEMENT TYPE EMPTY>
<!ATTLIST TYPE value (DYNAMIC|STATIC) #REQUIRED >
<!ELEMENT OGS_RULE_CONTAINER (OGSRULE?)>
<!ELEMENT OGSRULE (#PCDATA) >
<!ELEMENT TAGSTABLE EMPTY>
<!ATTLIST TAGSTABLE ref CDATA #REQUIRED >
<!ELEMENT READ_PERMISSION_LIST EMPTY>
<!ATTLIST READ_PERMISSION_LIST ref CDATA #REQUIRED >
<!ELEMENT WRITE_PERMISSION_LIST EMPTY>
<!ATTLIST WRITE_PERMISSION_LIST ref CDATA #REQUIRED >
<!ELEMENT EVALUATE_PERMISSION_LIST EMPTY>
<!ATTLIST EVALUATE_PERMISSION_LIST ref CDATA #REQUIRED >
<!ELEMENT PROPERTYTABLE (PROPERTY+)>
<!ATTLIST PROPERTYTABLE
name  CDATA #REQUIRED >
<!ELEMENT PROPERTY (KEY,VALUE)>
<!ELEMENT KEY (#PCDATA)>
<!ELEMENT VALUE (#PCDATA)>
<!ELEMENT USERLIST (USER*)>
<!ATTLIST USERLIST
name CDATA  #REQUIRED >
```

```
<!ELEMENT USER (#PCDATA)>
]>
<SYSTEMGROUPS>
<GROUP>
    <NAME>/System Defined Groups/Unknown Devices</NAME>
    <DESCRIPTION>Group for devices whose MDFId is unknown</DESCRIPTION>
    <TYPE value="DYNAMIC" />
    <OGS_RULE_CONTAINER>
        <OGSRULE>:CMF:DCR:Device.MDFId equals "UNKNOWN" </OGSRULE>
    </OGS_RULE_CONTAINER>
      <TAGSTABLE  ref="devicegrouptags" />
    <READ_PERMISSION_LIST ref="read_user_list" />
    <WRITE_PERMISSION_LIST ref="write_user_list" />
    <EVALUATE_PERMISSION_LIST ref="evaluate_user_list" />
</GROUP>
<GROUP>
    <NAME>/System Defined Groups/Cisco Access Servers</NAME>
    <DESCRIPTION>Group for Cisco Access Servers</DESCRIPTION>
    <TYPE value="DYNAMIC" />
    <OGS_RULE_CONTAINER>
        <OGSRULE> </OGSRULE>
    </OGS_RULE_CONTAINER>
    <TAGSTABLE  ref="devicegrouptags" />
    <READ_PERMISSION_LIST ref="read_user_list" />
    <WRITE_PERMISSION_LIST ref="write_user_list" />
    <EVALUATE_PERMISSION_LIST ref="evaluate_user_list" />
</GROUP>
<GROUP>
    <NAME>/System Defined Groups/Cisco Cable Devices</NAME>
    <DESCRIPTION>Group for Cisco Cable Devices</DESCRIPTION>
    <TYPE value="DYNAMIC" />
    <OGS_RULE_CONTAINER>
        <OGSRULE> </OGSRULE>
    </OGS_RULE_CONTAINER>
      <TAGSTABLE  ref="interfacegrouptags" />
    <READ_PERMISSION_LIST ref="read_user_list" />
    <WRITE_PERMISSION_LIST ref="write_user_list" />
    <EVALUATE_PERMISSION_LIST ref="evaluate_user_list" />
</GROUP>
<GROUP>
    <NAME>/System Defined Groups/Cisco Cable Devices/Cisco 6900 Series Multiplexers</NAME>
    <DESCRIPTION>Group for Cisco 6900 Series Multiplexers</DESCRIPTION>
    <TYPE value="DYNAMIC" />
    <OGS_RULE_CONTAINER>
        <OGSRULE> </OGSRULE>
    </OGS_RULE_CONTAINER>
      <TAGSTABLE  ref="interfacegrouptags" />
    <READ_PERMISSION_LIST ref="read_user_list" />
    <WRITE_PERMISSION_LIST ref="write_user_list" />
    <EVALUATE_PERMISSION_LIST ref="evaluate_user_list" />
</GROUP>
<PROPERTYTABLE name="systemdefinedgrouptags">
    <PROPERTY>
        <KEY>SYSTEM_PREDEFINED</KEY>
        <VALUE>TRUE</VALUE>
    </PROPERTY>
</PROPERTYTABLE>
<PROPERTYTABLE name="customizablegrouptags">
    <PROPERTY>
        <KEY>CUSTOMIZABLE_GROUP</KEY>
        <VALUE>TRUE</VALUE>
    </PROPERTY>
</PROPERTYTABLE>
<PROPERTYTABLE name="hiddengrouptags">
```

```
    <PROPERTY>
        <KEY>HIDDEN</KEY>
        <VALUE>TRUE</VALUE>
    </PROPERTY>
</PROPERTYTABLE>
<PROPERTYTABLE name="devicegrouptags">
    <PROPERTY>
        <KEY>MEMBERSHIP_TYPE</KEY>
        <VALUE>DEVICE</VALUE>
    </PROPERTY>
</PROPERTYTABLE>
<PROPERTYTABLE name="interfacegrouptags">
    <PROPERTY>
        <KEY>MEMBERSHIP_TYPE</KEY>
        <VALUE>INTERFACE</VALUE>
    </PROPERTY>
</PROPERTYTABLE>
```

Using OGS 1.4 Enhancements

The following enhancements are part of OGS 1.4 release:

Template Based Groups

Template based groups simplify group creation. You can create frequently used groups easily. If you want to create a group based on a device attribute (such as Location) you must select the Location based groups template, and enter the location values for the rule.

However, for group creation with more than one attribute (such as Location and IP Address) you must use the regular Rule based group creation.

Group ID

Group ID is introduced to the OGSGroupDefinition (com.cisco.nm.xms.ogs.util) class. Group ID is unique for a particular group. However, when a group is deleted, and then recreated, the group will not have the same group ID. A Getter method (groupId()) is added to the OGSGroupDefinition class for the applications to use the group ID.

Only the evaluateGroup method in the OGSInterface is overloaded to use the group ID, instead of the group name. The others such as deleteGroup and copyGroup continue to use group name.

The evaluateGroup() API is used from the application backend, and require the group ID.

A new class – com.cisco.nm.xms.ogs.util.OGSGroupId is added.This will encapsulate the group ID of a particular group as a string.

Configurable Display Name for Class Names

In group administration UIs, the internal class names (:CMF:DCR:Device or Kilner:VASA:KilnerObject:Device:Host) are displayed. This is not very useful for the end users.

With configurable display name for class names, you can have more user-friendly UI display names (such as Device or Host).

OGSClient.properties contains the mapping definitions. Applications can enable this feature using a specified property (EnableClassNameMapping=true in OGSClient.properties), and provide the mapping from the internal class name to the external class name.

Configurable Root (Provider) Group

In a single-server setup (DCR in standalone mode), the root group name is of the format *Application*. In a multi-server setup (DCR in Master / Slave mode), the root group name is of the format *Application-CWHP Display Name*.

Both the formats are available as templates in a configuration file, *NMSROOT*/objects/groups/RootGroupTemplate.properties. The default content of the file is:

```
# This property file is for configuring the template to be used for
# root (provider) group of all group hierarchies in this server
ProviderTemplateForSingleServer = $ProductId$
ProviderTemplateForMultiServer = "$ProductId$-$CWHPServerName$"
```

When the CiscoWorks administrator changes the DCR mode from standalone to master/slave, the root (provider) group name of all group hierarchies will be changed according to the template.

This change will be done automatically, and does not need the daemons to be restarted. Similarly, the root group name will be changed when the DCR mode is changed from Master/Slave to Standalone.

In a multi-server setup, the template can be changed to `$CWHPServerName$ - $ProductId$`. This change is global, applies to all OGS servers in the CiscoWorks machine, and takes effect only after the daemons are restarted. Hence, the properties file is shipped by Common Services and there will be only one properties file in a CiscoWorks installation.

The Common Services root group hierarchy will always be CS, irrespective of whether the server is in single-server mode or multi-server mode. Since there will be only one CS group hierarchy in a multi-server setup, there is no need to uniquely identify the CS group hierarchy by adding the CWHP display name.

com.cisco.nm.xms.ogs.util.OGSUtil will be augmented with the following APIs:

- Public string getCommonRoot( )—Returns the common group root (root of the CS group hierarchy). The value, as explained, will be CS.
- Public string getLocalRoot( )—Returns the current root group name of the local hierarchy. For example, this API will return the RME root group name when called with the classpath of the RMEOGSServer.
- Public string getOldLocalRoot( )—Returns the old root group name of the local hierarchy.

Integrating OGS 1.4 With Your Application

To integrate the client-side features of OGS 1.4 with your application:

Step 1 Include all OGS 1.4 related jar files in the applications and ensure that they are shipped with the respective webapp.

Step 2 Update the OGSClient.properties file with the following entries in the respective webapp environment:

```
TemplateSupportEnabled = true
TemplateFile=\\cwhp\\WEB-INF\\classes\\Templates.ser
DefaultTemplateToBeSelected=<Template String which needs to be selected default>
```

(\\Indicates the relative location of the path from webapps directory).

The default location of OGSClient.properties file is the WEB-INF\classes directory, relative to MDC\tomcat\webapps*app-name*.

By default, the template support is disabled because it is not consumed by all applications.

Step 3 Configure an XML file that describes the templates.

The DTD for the XML file is provided here, along with sample entries. Applications should specify the entries accordingly so that they would be made available out of the box.

Note Skipping the Membership page and enabling template support cannot be done together for the same group type. However, backward compatibility will be maintained for the other group types for which template support is not enabled.

Sample entries for creating templates in a XML file. Here the ITM application has been taken as an example for integration.

DTD for the XML File

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

<!--*******************************************************-->
<!--         Copyright (c) 2005 Cisco Systems, Inc.      -->
<!--      All rights reserved.                           -->
<!--*******************************************************-->

<!DOCTYPE TEMPLATES [
<!ELEMENT TEMPLATES (TEMPLATE+,PROPERTYTABLE+) >
<!ELEMENT TEMPLATE (NAME,DESCRIPTION?,TEMPLATE_RULE_CONTAINER,DISPLAY_LABEL?,TAGSTABLE?)>
<!ELEMENT NAME (#PCDATA)>
<!ELEMENT DESCRIPTION (#PCDATA)>
<!ELEMENT TEMPLATE_RULE_CONTAINER (TEMPLATERULE?)>
<!ELEMENT TEMPLATERULE (#PCDATA) >
<!ELEMENT DISPLAY_LABEL (#PCDATA)>
<!ELEMENT TAGSTABLE EMPTY>
<!ATTLIST TAGSTABLE ref CDATA #REQUIRED >
<!ELEMENT PROPERTYTABLE (PROPERTY+)>
<!ATTLIST PROPERTYTABLE name  CDATA #REQUIRED >
<!ELEMENT PROPERTY (KEY,VALUE)>
<!ELEMENT KEY (#PCDATA)>
<!ELEMENT VALUE (#PCDATA)>
]>
<TEMPLATES>
<TEMPLATE>
    <NAME>Location Based template</NAME>
    <DESCRIPTION>This template is used for creating groups based upon
location</DESCRIPTION>
    <TEMPLATE_RULE_CONTAINER>
        <TEMPLATERULE>:ITM:VASA:ITMObject:Device.Location contains</TEMPLATERULE>
    </TEMPLATE_RULE_CONTAINER>
      <DISPLAY_LABEL>List of Locations</DISPLAY_LABEL>
        <TAGSTABLE ref="Locationtemplates"/>
</TEMPLATE>
<TEMPLATE>
    <NAME>Name Based template</NAME>
    <DESCRIPTION>This template is used for creating groups based upon Name</DESCRIPTION>
    <TEMPLATE_RULE_CONTAINER>
        <TEMPLATERULE>:ITM:VASA:ITMObject:Device.Name contains</TEMPLATERULE>
    </TEMPLATE_RULE_CONTAINER>
      <DISPLAY_LABEL>List of MDFIds</DISPLAY_LABEL>
        <TAGSTABLE ref="Nametemplates"/>
</TEMPLATE>
<TEMPLATE>
    <NAME>Subnet Based template</NAME>
    <DESCRIPTION>This template is used for creating groups based upon Subnet</DESCRIPTION>
    <TEMPLATE_RULE_CONTAINER>
        <TEMPLATERULE>:ITM:VASA:ITMObject:Device.IP.Netmask contains</TEMPLATERULE>
    </TEMPLATE_RULE_CONTAINER>
```

```
      <DISPLAY_LABEL>List of Subnets</DISPLAY_LABEL>
        <TAGSTABLE ref="Subnettemplates"/>
</TEMPLATE>
<TEMPLATE>
    <NAME>Service Based template</NAME>
    <DESCRIPTION>This template is used for creating groups based upon
Service</DESCRIPTION>
    <TEMPLATE_RULE_CONTAINER>
        <TEMPLATERULE>:ITM:VASA:ITMObject:Device.Type contains</TEMPLATERULE>
    </TEMPLATE_RULE_CONTAINER>
      <DISPLAY_LABEL>List of Series</DISPLAY_LABEL>
        <TAGSTABLE ref="Servicetemplates"/>
</TEMPLATE>
<TEMPLATE>
    <NAME>Model Based template</NAME>
    <DESCRIPTION>This template is used for creating groups based upon Model</DESCRIPTION>
    <TEMPLATE_RULE_CONTAINER>
        <TEMPLATERULE>:ITM:VASA:ITMObject:Device.Model contains</TEMPLATERULE>
    </TEMPLATE_RULE_CONTAINER>
      <DISPLAY_LABEL>List of Models</DISPLAY_LABEL>
        <TAGSTABLE ref="Modeltemplates"/>
</TEMPLATE>

<PROPERTYTABLE name="Locationtemplates">
        <PROPERTY>
                <KEY>ATTRIBUTE</KEY>
                <VALUE>Location</VALUE>
        </PROPERTY>
</PROPERTYTABLE>
<PROPERTYTABLE name="Nametemplates">
    <PROPERTY>
                <KEY>ATTRIBUTE</KEY>
                <VALUE>Name</VALUE>
        </PROPERTY>
</PROPERTYTABLE>
<PROPERTYTABLE name="Servicetemplates">
    <PROPERTY>
                <KEY>ATTRIBUTE</KEY>
                <VALUE>Type</VALUE>
        </PROPERTY>
</PROPERTYTABLE>
<PROPERTYTABLE name="Modeltemplates">
    <PROPERTY>
                <KEY>ATTRIBUTE</KEY>
                <VALUE>Model</VALUE>
        </PROPERTY>
</PROPERTYTABLE>
<PROPERTYTABLE name="Subnettemplates">
    <PROPERTY>
                <KEY>ATTRIBUTE</KEY>
                <VALUE>IP.NetMask</VALUE>
        </PROPERTY>
</PROPERTYTABLE>
<PROPERTYTABLE name="">
    <PROPERTY>
                <KEY>ATTRIBUTE</KEY>
                <VALUE></VALUE>
        </PROPERTY>
</PROPERTYTABLE>

</TEMPLATES>
```

Note The element TAGSTABLE is optional. TAGSTABLE is to add tags if the application needs to find the groups created based on a specific template.

Step 4 Define entries related to the new screen IDs in struts-config.xml file and site-map XML file of the respective webapp.

Entries for struts-config.xml

The following entries must be added in the OGS related sections of struts-config.xml.

```
<forward name="ogsBrowseProperties"
path="/WEB-INF/screens/uii/index.jsp?sid=ogsBrowseProperties" />
             <forward name="ogsBrowseRules"
path="/WEB-INF/screens/uii/index.jsp?sid=ogsBrowseRules" />
             <forward name="ogsBrowseMembership"
path="/WEB-INF/screens/uii/index.jsp?sid=ogsBrowseMembership" />
             <forward name="ogsBrowseSubgroups"
path="/WEB-INF/screens/uii/index.jsp?sid=ogsBrowseSubgroups" />
             <forward name="ogsCreateProperties"
path="/WEB-INF/screens/uii/index.jsp?sid=ogsCreateProperties" />
             <forward name="ogsTemplateCreateProperties"
path="/WEB-INF/screens/uii/index.jsp?sid=ogsTemplateCreateProperties" />
             <forward name="ogsCreateRules"
path="/WEB-INF/screens/uii/index.jsp?sid=ogsCreateRules" />
             <forward name="ogsDefaultCreateRules"
path="/WEB-INF/screens/uii/index.jsp?sid=ogsDefaultCreateRules" />

<forward name="ogsTemplateCreateRules"
path="/WEB-INF/screens/uii/index.jsp?sid=ogsTemplateCreateRules" />
             <forward name="ogsCreateMembership"
path="/WEB-INF/screens/uii/index.jsp?sid=ogsCreateMembership" />
             <forward name="ogsDefaultCreateMembership"
path="/WEB-INF/screens/uii/index.jsp?sid=ogsDefaultCreateMembership" />

<forward name="ogsTemplateCreateMembership"
path="/WEB-INF/screens/uii/index.jsp?sid=ogsTemplateCreateMembership" />
              <forward name="ogsCreateAccessControl"
path="/WEB-INF/screens/uii/index.jsp?sid=ogsCreateAccessControl" />
             <forward name="ogsCreateSummary"
path="/WEB-INF/screens/uii/index.jsp?sid=ogsCreateSummary" />
              <forward name="ogsDefaultCreateSummary"
path="/WEB-INF/screens/uii/index.jsp?sid=ogsDefaultCreateSummary" />

<forward name="ogsTemplateCreateSummary"
path="/WEB-INF/screens/uii/index.jsp?sid=ogsTemplateCreateSummary" />
              <forward name="ogsEditProperties"
path="/WEB-INF/screens/uii/index.jsp?sid=ogsEditProperties" />

<forward name="ogsTemplateEditProperties"
path="/WEB-INF/screens/uii/index.jsp?sid=ogsTemplateEditProperties" />
             <forward name="ogsEditRules"
path="/WEB-INF/screens/uii/index.jsp?sid=ogsEditRules" />
             <forward name="ogsTemplateEditRules"
path="/WEB-INF/screens/uii/index.jsp?sid=ogsTemplateEditRules" />
             <forward name="ogsEditMembership"
path="/WEB-INF/screens/uii/index.jsp?sid=ogsEditMembership" />
             <forward name="ogsTemplateEditMembership"
path="/WEB-INF/screens/uii/index.jsp?sid=ogsTemplateEditMembership" />
             <forward name="ogsEditAccessControl"
path="/WEB-INF/screens/uii/index.jsp?sid=ogsEditAccessControl" />
```

```
            <forward name="ogsEditSummary"
path="/WEB-INF/screens/uii/index.jsp?sid=ogsEditSummary" />
            <forward name="ogsTemplateEditSummary"
path="/WEB-INF/screens/uii/index.jsp?sid=ogsTemplateEditSummary" />
            <forward name="ogsMainSetup"
path="/WEB-INF/screens/uii/index.jsp?sid=ogsMainSetup" />
            <forward name="ogsRoleAccessControl"
path="/WEB-INF/screens/popup.jsp?sid=ogsRoleAccessControl" />

<forward name="ogsServerError" path="/WEB-INF/screens/uii/index.jsp?sid=ogsServerError" />

<forward name="ogsSkipMemberCreateProperties"
path="/WEB-INF/screens/uii/index.jsp?sid=ogsSkipMemberCreateProperties" />

<forward name="ogsSkipMemberCreateRules"
path="/WEB-INF/screens/uii/index.jsp?sid=ogsSkipMemberCreateRules" />

<forward name="ogsSkipMemberCreateSummary"
path="/WEB-INF/screens/uii/index.jsp?sid=ogsSkipMemberCreateSummary" />

<forward name="ogsSkipMemberEditProperties"
path="/WEB-INF/screens/uii/index.jsp?sid=ogsSkipMemberEditProperties" />

<forward name="ogsSkipMemberEditRules"
path="/WEB-INF/screens/uii/index.jsp?sid=ogsSkipMemberEditRules" />

<forward name="ogsSkipMemberEditSummary"
path="/WEB-INF/screens/uii/index.jsp?sid=ogsSkipMemberEditSummary" />
```

Entries for Sitemap XML File

The following entries must be added in the OGS-related section in the sitemap XML file of the respective webapp:

```
<?xml version="1.0"?>
<contentAreaSet>
     <navContentArea screenID="ogsMainSetup"
        contentAreaTitle="Group Administration"
        helpTag="group_administration"
        fileRef="/WEB-INF/screens/OGS/ogsMainSetup.jsp">
    </navContentArea>
    <appContentArea screenID="ogsBrowseProperties"
        contentAreaTitle="Property Details"
        helpTag="cs_groups_admin_view"
        fileRef="/WEB-INF/screens/OGS/ogsBrowseProperties.jsp" />
    <appContentArea screenID="ogsBrowseMembership"
        contentAreaTitle="Membership Details"
        helpTag="cs_groups_admin_view"
        fileRef="/WEB-INF/screens/OGS/ogsBrowseMembership.jsp" />
    <appContentArea screenID="ogsServerError"
        contentAreaTitle="Group Administration Server Error"
        helpTag="ogs_server_error"
        fileRef="/WEB-INF/screens/OGS/serverError.jsp" />
    <appContentArea screenID="ogsDCRSecondary"
        contentAreaTitle=""
                helpTag="ogs_dcr_secondary"
                fileRef="/WEB-INF/screens/OGS/ogsDCRSecondary.jsp" />
    <appContentArea screenID="ogsNewPage"
         contentAreaTitle=""
         helpTag="ogs_dcr_secondary"
         fileRef="/WEB-INF/screens/OGS/ogsNewPage.jsp" />

    <appWizard>
        <appWizardItem>
```

```
            <label>Properties</label>
            <appContentArea screenID="ogsTemplateCreateProperties"
                contentAreaTitle="Properties"
                helpTag="cs_groups_admin_create"
                fileRef="/WEB-INF/screens/OGS/ogsTemplateCreateProperties.jsp" />
        </appWizardItem >
    <appWizard>
        <appWizardItem>
            <label>Rules</label>
            <appContentArea screenID="ogsDefaultCreateRules"
                contentAreaTitle="Rules"
                helpTag="cs_groups_admin_create"
                fileRef="/WEB-INF/screens/OGS/ogsCreateRules.jsp" />
        </appWizardItem>
        <appWizardItem>
            <label>Membership</label>
            <appContentArea screenID="ogsDefaultCreateMembership"
                contentAreaTitle="Membership"
                helpTag="cs_groups_admin_create"
                fileRef="/WEB-INF/screens/OGS/ogsCreateMembership.jsp" />
        </appWizardItem>
        <appWizardItem>
            <label>Summary</label>
            <appContentArea screenID="ogsDefaultCreateSummary"
                contentAreaTitle="Summary"
                helpTag="cs_groups_admin_create"
                fileRef="/WEB-INF/screens/OGS/ogsCreateSummary.jsp" />
        </appWizardItem>
    </appWizard>
           <appWizard>
        <appWizardItem>
            <label>Templates</label>
            <appContentArea screenID="ogsTemplateCreateRules"
                contentAreaTitle="Templates"
                helpTag="cs_groups_admin_create"
                fileRef="/WEB-INF/screens/OGS/ogsTemplateCreateRules.jsp" />
        </appWizardItem>
        <appWizardItem>
            <label>Membership</label>
            <appContentArea screenID="ogsTemplateCreateMembership"
                contentAreaTitle="Membership"
                helpTag="cs_groups_admin_create"
                fileRef="/WEB-INF/screens/OGS/ogsCreateMembership.jsp" />
        </appWizardItem>
        <appWizardItem>
            <label>Summary</label>
            <appContentArea screenID="ogsTemplateCreateSummary"
                contentAreaTitle="Summary"
                helpTag="cs_groups_admin_create"
                fileRef="/WEB-INF/screens/OGS/ogsCreateSummary.jsp" />
        </appWizardItem>
    </appWizard>
    </appWizard>
    <appWizard>
        <appWizardItem>
            <label>Properties</label>
            <appContentArea screenID="ogsEditProperties"
                contentAreaTitle="Properties"
                helpTag="cs_groups_admin_modify"
                fileRef="/WEB-INF/screens/OGS/ogsEditProperties.jsp" />
        </appWizardItem>
        <appWizardItem>
            <label>Rules</label>
            <appContentArea screenID="ogsEditRules"
```

```
            contentAreaTitle="Rules"
            helpTag="cs_groups_admin_modify"
            fileRef="/WEB-INF/screens/OGS/ogsEditRules.jsp" />
    </appWizardItem>
    <appWizardItem>
        <label>Membership</label>
        <appContentArea screenID="ogsEditMembership"
            contentAreaTitle="Membership"
            helpTag="cs_groups_admin_modify"
            fileRef="/WEB-INF/screens/OGS/ogsEditMembership.jsp" />
    </appWizardItem>
    <appWizardItem>
        <label>Summary</label>
        <appContentArea screenID="ogsEditSummary"
            contentAreaTitle="Summary"
            helpTag="cs_groups_admin_modify"
            fileRef="/WEB-INF/screens/OGS/ogsEditSummary.jsp" />
    </appWizardItem>
    </appWizard>

<appWizard>
    <appWizardItem>
        <label>Properties</label>
        <appContentArea screenID="ogsCreateProperties"
            contentAreaTitle="Properties"
            helpTag="cs_groups_admin_modify"
            fileRef="/WEB-INF/screens/OGS/ogsCreateProperties.jsp" />
    </appWizardItem>
    <appWizardItem>
        <label>Rules</label>
        <appContentArea screenID="ogsCreateRules"
            contentAreaTitle="Rules"
            helpTag="cs_groups_admin_modify"
            fileRef="/WEB-INF/screens/OGS/ogsCreateRules.jsp" />
    </appWizardItem>
    <appWizardItem>
        <label>Membership</label>
        <appContentArea screenID="ogsCreateMembership"
            contentAreaTitle="Membership"
            helpTag="cs_groups_admin_modify"
            fileRef="/WEB-INF/screens/OGS/ogsCreateMembership.jsp" />
    </appWizardItem>
    <appWizardItem>
        <label>Summary</label>
        <appContentArea screenID="ogsCreateSummary"
            contentAreaTitle="Summary"
            helpTag="cs_groups_admin_modify"
            fileRef="/WEB-INF/screens/OGS/ogsCreateSummary.jsp" />
    </appWizardItem>
</appWizard>

<appWizard>
    <appWizardItem>
        <label>Properties</label>
        <appContentArea screenID="ogsTemplateEditProperties"
            contentAreaTitle="Properties"
            helpTag="cs_groups_admin_modify"
            fileRef="/WEB-INF/screens/OGS/ogsEditProperties.jsp" />
    </appWizardItem>
    <appWizardItem>
        <label>Templates</label>
        <appContentArea screenID="ogsTemplateEditRules"
            contentAreaTitle="Templates"
            helpTag="cs_groups_admin_modify"
```

```
            fileRef="/WEB-INF/screens/OGS/ogsTemplateEditRules.jsp" />
    </appWizardItem>
    <appWizardItem>
        <label>Membership</label>
        <appContentArea screenID="ogsTemplateEditMembership"
            contentAreaTitle="Membership"
            helpTag="cs_groups_admin_modify"
            fileRef="/WEB-INF/screens/OGS/ogsEditMembership.jsp" />
    </appWizardItem>
    <appWizardItem>
        <label>Summary</label>
        <appContentArea screenID="ogsTemplateEditSummary"
            contentAreaTitle="Summary"
            helpTag="cs_groups_admin_modify"
            fileRef="/WEB-INF/screens/OGS/ogsEditSummary.jsp" />
    </appWizardItem>
</appWizard>

<appWizard>
    <appWizardItem>
        <label>Properties</label>
        <appContentArea screenID="ogsSkipMemberCreateProperties"
            contentAreaTitle="Properties"
            helpTag=""
            fileRef="/WEB-INF/screens/OGS/ogsCreateProperties.jsp" />
    </appWizardItem>
    <appWizardItem>
        <label>Rules</label>
        <appContentArea screenID="ogsSkipMemberCreateRules"
            contentAreaTitle="Rules"
            helpTag=""
            fileRef="/WEB-INF/screens/OGS/ogsCreateRules.jsp" />
    </appWizardItem>
    <appWizardItem>
        <label>Summary</label>
        <appContentArea screenID="ogsSkipMemberCreateSummary"
            contentAreaTitle="Summary"
            helpTag=""
            fileRef="/WEB-INF/screens/OGS/ogsCreateSummary.jsp" />
    </appWizardItem>
</appWizard>

      <appWizard>
    <appWizardItem>
        <label>Properties</label>
        <appContentArea screenID="ogsSkipMemberEditProperties"
            contentAreaTitle="Properties"
            helpTag=""
            fileRef="/WEB-INF/screens/OGS/ogsEditProperties.jsp" />
    </appWizardItem>
    <appWizardItem>
        <label>Rules</label>
        <appContentArea screenID="ogsSkipMemberEditRules"
            contentAreaTitle="Rules"
            helpTag=""
            fileRef="/WEB-INF/screens/OGS/ogsEditRules.jsp" />
    </appWizardItem>
    <appWizardItem>
        <label>Summary</label>
        <appContentArea screenID="ogsSkipMemberEditSummary"
            contentAreaTitle="Summary"
            helpTag=""
            fileRef="/WEB-INF/screens/OGS/ogsEditSummary.jsp" />
    </appWizardItem>
```

```
    </appWizard>
</contentAreaSet>
```

Note You need to make these changes to the struts-config.xml and respective sitemap XML files only if template support is required. If template support is not required, you do not need to make any changes to the existing files.

Entry for Displaying Error Messages

Add the following entry in the appropriate property file for the webapp to display error messages:

ogs.noTemplateAttrValues=<Attribute value>

You must enter at least one value for this entry.

Generating Serialized File from the XML File

To generate a serialized file from the XML file, enter the following (where? CLI?):

com.cisco.nm.xms.ogs.client.templates.OGSTemplateGenerator *XML_file_name serialized_file_name*

To run this command (?), the following files should be available in the classpath:

NMSROOT\MDC\tomcat\shared\lib\mice.jar;

NMSROOT\www\classpath;

NMSROOT\lib\classpath;

NMSROOT\MDC\tomcat\shared\lib\MICE.jar;

NMSROOT\lib\classpath\jconn2.jar;

NMSROOT\MDC\tomcat\webapps\cwhp\WEB-INF\lib\log4j.jar;

NMSROOT\MDC\tomcat\shared\lib\xerces.jar;

NMSROOT\MDC\tomcat\shared\lib\xalan.jar;

NMSROOT\MDC\tomcat\webapps\cwhp\WEB-INF\lib\ogs-client1.2.jar;

NMSROOT\MDC\tomcat\webapps\cwhp\WEB-INF\lib\cogs1.2.jar;

Integrating Configurable Display Name for Class Names Feature with Applications

In OGSClient.properties, the following properties are added to enable this feature.

- EnableClassNameMapping = true
- ClassNameMappingFile = DCR-Class-Name-Mapping.xml

Setting EnableClassNameMapping property to true enables this feature. ClassNameMappingFile property specifies the XML file where the mapping between internal classnames and external (displayable) class names are specified.

This XML file should be present in the “WEB-INF/classes” directory. The applications need to make sure that all classnames are entered properly in the XML mapping file.

The following is the DTD for the classname mapping file:

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

```
<!DOCTYPE Mappings [
<!ELEMENT Mappings (ClassNameMap+)>
<!ELEMENT ClassNameMap (InternalClass, ExternalClass)>
<!ELEMENT InternalClass (#PCDATA) >
<!ELEMENT ExternalClass (#PCDATA) >
]>
```

The following is the sample classname mapping file:

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

<!DOCTYPE Mappings [
<!ELEMENT Mappings (ClassNameMap+)>
<!ELEMENT ClassNameMap (InternalClass, ExternalClass)>
<!ELEMENT InternalClass (#PCDATA) >
<!ELEMENT ExternalClass (#PCDATA) >
]>

<Mappings>
    <ClassNameMap>
        <InternalClass>:DFM:VASA:DFMObject</InternalClass>
        <ExternalClass>DFMObject</ExternalClass>
    </ClassNameMap>
    <ClassNameMap>
        <InternalClass>:DFM:VASA:DFMObject:AccessPort</InternalClass>
        <ExternalClass>AccessPort</ExternalClass>
    </ClassNameMap>
    <ClassNameMap>
        <InternalClass>:DFM:VASA:DFMObject:Device</InternalClass>
        <ExternalClass>Device</ExternalClass>
    </ClassNameMap>
    <ClassNameMap>
        <InternalClass>:DFM:VASA:DFMObject:Device:Cable</InternalClass>
        <ExternalClass>Cable</ExternalClass>
    </ClassNameMap>
    <ClassNameMap>
            <InternalClass>:DFM:VASA:DFMObject:Device:ContentNetworking</InternalClass>
        <ExternalClass>ContentNetworking</ExternalClass>
    </ClassNameMap>
</Mappings>
```

The following is the enhancement to Object Grouping Server (OGS)1.4.2 release:

Enhancements to evaluateGroup API

The evaluateGroup method is an overloaded API in the OGSInterface. The evaluateGroup() API now takes an additional argument, taglist, which is a String array. This String array returns the values of the tag lists passed, for each of the group.

This enhanced evaluateGroup() method can be used when the caller wants to evaluate the group to display the devices and subgroups, and to retreive the group properties as tag-value pairs.

The API signature is as follows:

```
public OGSObjectList evaluateGroup(OGSSecurityContext context,
                                   String task,
                                   OGSRuleContainer filter,
                                   String groupName,
                                   String[] attributeList,
                                   Integer membershipFormat,
                                   Boolean recomputeMembers,
                                   String[] tagList) throws OGSException;
```