CHAPTER 28

Integrating Applications With Device Selector

The Device Selector is used to select devices to perform various device management tasks. This lists all devices in a group. The Display Name of the devices entered when you have added the devices in DCR is dispayed as the device name in the Device Selector.

The Device Selector is enhanced in Common Services 3.0.5 to address the uasbility issues. The enhancements to Device Selector include the following:

- Search and Advanced Search funtionality
- Group Customization and Group Ordering Feature
- Tooltips for device names
- Hyperlink to the message “x devices selected”

Refer to the User Guide for CiscoWorks Common Services 3.0.5 for more information on new features and enhancements.

DCR and Device Center is integrated with the enhanced Device Selector for performing the device management functions in Common Services 3.0.5.

Individual applications should integrate with the enhanced Device Selector delivered as part of Common Services 3.0.5 and User Interface Infrastructure kit 6.3 for their device management tasks.

This chapter explains you the following details of integration with New Device Selector:

- UII Integration
- Integration with Search feature
- Integration with Advanced Search Feature
- Integration with Tree Generator

Note Javadocs provides the details of all APIs and the complete set of customizations that can made to Serach and Advanced Search implementation. Applications can override few of the APIs to suit to their requirements.

UII Integration

UII tag libraries are enhanced and new tags are added to the the existing HTML Object Selector (HOS) to add the support for Search and Advanced search.

The following new properties are introduced in the uii:hosContentAreaSelector component.

- showSearch - The value must be set to true to enable the search feature.
- showAdvancedSearch -The value must be set to false to enable the advanced search feature
- searchHandler -This property refers to the implementation class that provides the implementation of search. In other words, this refers to the implementation class of com.cisco.nm.uii.hos.SearchHandler interface

UII provides the enhanced APIs for applications to set the Search results as a result of search or advanced search operation.

Applications must add the following code in their classes apart from the set of UII calls to enable the search behaviour:

```
else if(HOSRequestHelper.isRequestForSearch(request))
{
            return HOSRequestHelper.doSearch(request,mapping);
}
else if (HOSRequestHelper.isRequestForSaveSelections(request)) {
           return HOSRequestHelper.saveSelectedEntriesAndForward(request ,null ,mapping);
}
```

Refer to UII 6.3 SDK documentation for more information on UII Integration with Search and Advanced Search feature.

Integration with Search feature

You can enter your search criteria in the Search Input field of Device Selector and search for the devices using the Search icon. The search results are based on the display name of the devices added in DCR.

Common Services provides the default implementation for the SearchHandler interface of UII. Applications should extend this implementation class **com.cisco.nm.cmf.devsel.SearchHandlerImpl** to validate the search criteria specified, search the CMF database based on search criteria, get the list of devices managed by the local application and to filter the results based on the application contexts.

Applications should extend the implementation class to provide the following features:

- Perform device based authorization - You should set the OGSSecurityContect in order to mention the TaskId and ApplicationId values for performing device based authorization.
- Provide any custom search - If an application wants to search on another database in addition to DCR, it should extend the implentation class provided by CMF or implement the UII SearchHandler interface directly.

Refer to the Javadocs for the API specifications for all the classes provided by Common Services. Application can extend the Common Services classes and customize their implementation as required.

Configuring Property files

In addition to extending the Common Services classes, Applications need to configure few properties in the following properties files:

1. DeviceSelector.properties - You should specify a set of properties in this file for the search feature to function.
   a. OgsUrn - This property defines the URN of the OGS Server. Enter the value only when your application have a URN different from Common Services.
   b. OgsServerName - This property defines the name of the local OGS Server used for searching devices. The OGS Server Name used for Common Services is CMFOGSServer.
   c. OgsRelativeURL - Defines the relative URL of the CTMServlet for the local OGS Server. For example, you can enter the property value as /cwhp/CTMServlet.
   d. AllDevicesGroup - Defines the name of the All Devices group that is used for filtering based on aplplications or tasks after a device search.
   e. SetMDFIcon - Defines whether the icons needs to set for the Search results. Set the value as Yes, if you want to display the search results (device names) with their icons. Default value is No.
2. log4j-devsel.properties - Properties must be specified to define the log levels of the Common Services search or advanced search implementations.
   a. -log4j-devsel.properties: For defining the log levels of the CS search / advanced search implementations.
   b. log4j.category.com.cisco.nm.cmf.devsel=DEBUG, DevSel
   c. log4j.category.com.cisco.nm.cmf.devsel.util=DEBUG, DevSel
   d. log4j.additivity.com.cisco.nm.cmf.devsel=false
   e. log4j.appender.DevSel=org.apache.log4j.RollingFileAppender
   f. log4j.appender.DevSel.File=CSDeviceSelector.log
   g. log4j.appender.DevSel.layout=org.apache.log4j.PatternLayout
   h. log4j.appender.DevSel.layout.ConversionPattern=%d{dd/MMM/yyyy HH:mm:ss:SSS} %-4r [%t] %-5p %c %x %L - %m%n

Integration with Advanced Search Feature

You can use the Advanced Search icon to open the Advanced Search popup window and specify a set of rules for performing an Advanced search. The advanced search is based on the grouping attributes of the application's grouping server. For example, when you launch an Advanced Search from Campus Manager Device Selector, the attributes of the Campus Manager Grouping server appears.

Common Services provides the default implementation of View, Form and Action classes that connects to the respective local OGS Servers and provide the default rule based search.

Applications need to configure the following to integrate the Advanced Search feature:

1. Applications need to configure few property files to let the Common Services Implementation know about the details of the OGS Server. Refer to Configuring Property files for the list of property files and properties to be configured.
2. Applications should deliver the JSP files DeviceFilter.jsp and closeDeviceFilter.jsp.

3. Application should define the struts-config and sitemap XML files to configure the mapping between the form bean and action classes, and to set other parameters.

   a. Changes to struts-config.xml

      Form bean entries:

```
<!-- ####### DeviceSelector Specific ####### -->
<form-bean name="DeviceFilterForm" type="com.cisco.nm.cmf.devsel.DeviceFilterForm"
/>
<!-- ####### DeviceSelector Specific END ####### -->
```

      Global forward entries:

```
<!-- ######## DeviceSelector Specific ######## -->
<forward name="DeviceFilter" path="/WEB-INF/screens/popup.jsp?sid=DeviceFilter" />
<forward name="closeDeviceFilter"
path="/WEB-INF/screens/deviceselector/closeDeviceFilter.jsp" />
<!-- ######## DeviceSelector Specific END  ######## -->
```

      Action mapping entries:

```
<!-- ########## DeviceSelector Specific ########## -->
<action path="/DeviceFilter" scope="request" name="DeviceFilterForm"
validate="false" type="com.cisco.nm.cmf.devsel.DeviceFilterAction">
</action>
<!-- ########## DeviceSelector Specific END ########## -->
<action path="/hosTreeContent" name="nullFB"
type="com.cisco.nm.uii.hos.action.HOSContentAreaAction"/>
```

   b. Changes to sitemap.xml

      Add the following entries in the application pages containing the HOS component with the advanced search:

```
<appContentArea screenID="DeviceFilter" contentAreaTitle="Advanced Search"
helpTag="" fileRef="/WEB-INF/screens/deviceselector/DeviceFilter.jsp" />
```

4. Applications should define the Javascript function in their JSP files to launch the advanced search popup screen when you click the advanced search icon.

   The Javascript function is as follows:

```
function advancedSearchBtnOnClickHandler(osName) {
    window.open("/cwhp/DeviceFilter.do?treeID="+osName, "newer", width=580,
height=350, scrollbars=no,resizable=yes",true); }
function submitForm() {
    window.frames['contentFrame'].document.forms[0].submit(); }
```

Application can go for any custom implementation and create new action classes. To accomplish that applications should call an API in HOS to provide the search results. This API in HOS needs to be called only if the search operation is successful. Refer to UII 6.3 SDK documentation for more information.

Integration with Tree Generator

The Tree Generator is enhanced to provide a new default tree view to end users with new device groups such as Application Specific Groups and User Defined Groups in the tree. The Tree Generator also provides a customized tree view as per the configurations saved by end users while using the Group Customization and Group Ordering feature.

Common Services provides the Tree Generator implementation for the other applications to integrate.

This section explains the following:

Tree Generator Changes for Device Selector Nodes

Since Common Services does not have the “All Devices” Node or Group, the existing Tree Generator is extended to add an “All Devices” group.

Tree Generator now extends com.cisco.nm.xms.ogs.client.ostaglib.util.AbstractTreeStateManager class, which adds a new API to return the id of a group containing the devices of all application. Applications that have the device group “All Devices” can return the same group for this new API or can extend the existing Tree Generator classes.

Apart from this, Applications should extend the Tree Generator classes to add the application specific groups to the Device Selector or when they need a tree view specific to the application.

Tree Generator Changes for Search Implementations

Tree Generator sets the OGSSecurityContext object in the HttpSession as required by the Search and Advanced Search implementations and is used in all API calls to Object Grouping Services Server.

Tree Generator Changes for Group Customization and Group Ordering

Based on the Group Customization and Group Ordering settings entered by an end user, the Device Selector should load a customized tree view to the logged in user.

Common Services provides a base implementation for the new tree generator which uses the Group Customization and Group Ordering feature. Applications should use the Tree Generator class com.cisco.nm.cmf.devsel.DeviceSelectorTreeGenerator provided by Common Services.

Few classes are added to the Device Selector util and preference packages for the new Tree Generator. the Table below lists the classes added to Device Selector packages and their description.

Table 28-1 New Classes added to Device Selector packages for Tree Generator

Package	Class	Description
com.cisco.nm.cmf.devsel.util	Group	Stores the mapping between group id and group name
	GroupCollection	Collection class for the group object
	GroupMembership	Stores the membership of group and this is an extension class of Group
	GroupMembershipCollection	Collection class for the Group Membership object
com.cisco.nm.cmf.devsel.preference	DeviceSelectorPreferences	Stores the device selector settings entered by an end user.
	DeviceSelector	Stores and retrieves the device selector preferences entered. This is a Main class exposing the APIs.
	DeviceSelectorConstants	Defines the constants to be used in DeviceSelector class. For example, ALL_DEVICES.

Application needs to integrate this implementation to communicate with the respective local OGS Server, get the user preferences from the database, return the top-level groups and return the devices for the top-level groups.