CHAPTER 11

Customizing Prime Network Workflow

This chapter explains how to extend the Workflow Engine with custom tasks and Prime Network Workflow callbacks.

Topics include:

Coding

This section includes the following topics:

Custom Tasks

The procedure for developing custom tasks is explained in the Autonomy documentation.

The following is the definition of a simple task class:

```
package samples;

import com.dralasoft.workflow.Key;
import com.dralasoft.workflow.SynchronousTask;

public class CustomTask1 extends SynchronousTask {
    public CustomTask1(Key _key) {
        super(_key);
    }

    public void perform() {
        System.err.println("Hello from CustomTask1");
    }
}
```

Custom Task Panel Factories

A TaskPanelFactory implementation class can be assigned to each custom task class. This allows you to create custom property sheets for the custom class. This factory class should implement the interface *com.dralasoft.gui.common.ext.TaskPanelFactory*. A simple way to do this is to extend *com.dralasoft.gui.common.ext.DefaultPanelFactory* and override some of its methods.

Workflow Callbacks Class

It is possible to develop a class that implements the Prime Network Workflow callbacks. This class should implement the interface *com.sheer.client.workflowstudio.IWorkflowEditorCallbacks*, which includes the following methods:

```
/**
* Called before the deploy template action
* @param templateName
* @return true if deploy template action should proceed
*/
public boolean preDeployTemplate(String templateName);

/**
* Called after the deploy template action
* @param templateName
*/
public void postDeployTemplate(String templateName);

/**
* Called before the delete template action
* @param templateName
* @return true if delete template action should proceed
*/
public boolean preDeleteTemplate(String templateName);

/**
* Called after the delete template action
* @param templateName
*/
public void postDeleteTemplate(String templateName);

/**
* @param templateName
* @return true if this template should be displayed, false if not
*/
public boolean shouldDisplayTemplate(String templateName);
```

Packaging for Deployment

Classes and resources must be packaged into JAR files to be deployed on the server. A JAR file can contain multiple tasks and an optional workflow callback implementation.

In addition to the class files, each JAR file must contain a descriptor file named *extension-config.xml*. This file contains the XML block or blocks that define the tasks' appearance in the task palette, the tasks' custom panel factories, and, optionally, the Prime Network Workflow callback implementation.

Custom Tasks

In addition to the custom tasks class files, task icons should be included in the JAR file.

Task icons should measure 16 by 16 pixels and must be placed in the JAR file in the subdirectory, com/dralasoft/gui/common/images/16x16.

Custom Task Panel Factories

The full class name of the task panel factory should be added in an element called task-panel-factory-class inside the custom-task element. If this element is not specified, *com.dralasoft.gui.common.ext.DefaultPanelFactory* is used for the custom task.

Workflow Callbacks Class

Adding an editor-callbacks-class element to the extension-config element configures a class that implements the Prime Network Workflow callbacks. This element should be added to the extension-config.xml file in only one of the JAR files (if the element is present in more then one descriptor, one element's value would be used arbitrarily).

The following example of the contents of a task descriptor file defines two task types, one of which has a task panel factory, and an editor callback implementation:

```
<extension-config>
    <editor-callbacks-class>
        samples.EditorCallbacksImpl
    </editor-callbacks-class>

    <custom-task>
        <class-name>samples.CustomTask1</class-name>
        <label>Custom Task 1</label>
        <icon>task1.png</icon>
        <tooltip>Custom Task 1</tooltip>
        <menu-display>true</menu-display>
        <toolbar-display>true</toolbar-display>
        <task-panel-factory-class>
            com.sheer.client.workflowstudio.TestTaskPanelFactory
        </task-panel-factory-class>
    </custom-task>

    <custom-task>
        <class-name>samples.CustomTask2</class-name>
        <label>Custom Task 2</label>
        <icon>task2.png</icon>
        <tooltip>Custom Task 2</tooltip>
        <menu-display>true</menu-display>
        <toolbar-display>true</toolbar-display>
    </custom-task>
</extension-config>
```

Deploying

To deploy JAR files:

Step 1 On the Prime Network gateway, copy the JAR files to the /export/home/network38/dralasoft_extensions directory.

Note Create the Prime Network gateway if necessary.

Step 2 Run the script /export/home/network38/Main/scripts/installDralasoftExtensions.pl.

Note Run this script each time you want to add a JAR file, remove a JAR file, or replace an existing JAR file with a new version.

The script installs and uninstalls JAR files so that the set of installed JAR files is the same as the contents of the directory /export/home/network38/dralasoft_extensions. We therefore recommend that you keep files with a .jar extension here even after they have been deployed.

Note The script restarts the Workflow engine, so we recommend that you confirm that no workflows are currently running in the engine.

Step 3 When the script is done, run Prime Network Workflow again. The deployed JAR files are automatically downloaded, and the new tasks are included in the task toolbar.

About Workflow Migration

If you are upgrading from an earlier version of Cisco ANA to Prime Network 3.8, see the installation guide for that version.

There is no change to the workflow engine in Prime Network 3.8, and the supported workflow version remains 3.6.6. Prime Network workflow templates are stored in the database and are available after upgrading to Prime Network 3.8.

Existing workflow templates that execute BQL inventory commands might require a review to verify the correctness and validity of these commands. Review the IMO and BQL command changes as described in the *Cisco Prime Network Integration Developer Guide* and in the IMO documentation, available on Cisco Prime Network Technology Center.

Your existing Prime Network Workflow AVM (avm66.xml) is maintained during the upgrade to Prime Network 3.8 so that any user-defined entries or customizations are available in Prime Network 3.8.

Any workflow extensions stored under ~/dralasoft_extensions are maintained during the upgrade to Prime Network 3.8.

Note The workflow template must be saved or imported using the Export and Import options available in Prime Network Workflow. Saving or copying by any other method might not work as desired.