Technical Note

Policy Manager 7.x Custom Actions

SOA Software, Inc.

12100 Wilshire Blvd, Suite 1800
Los Angeles, CA 90025
866-SOA-9876
www.soa.com
info@soa.com

Table of Contents

1 Overview

This technical note provides instructions for configuring a Policy Manager custom action.

A Custom Action is an additional function that can be added to a Policy Manager "Actions Portlet" (via an item's "Details" page) or "Actions drop-down list box" (via an item's "Summary" page). Custom Actions are typically used to launch web user interfaces from within the Policy Manager "Management Console" that extend the capabilities provided by the Policy Manager product.

2 Custom Action Development Steps

This section describes the steps for configuring and activating a Custom Action.

Step	Procedure
1.	Determine the type of Custom Action you would like to define, and the Policy Manager item type that will be associated with the action.
2.	Create a new or use an existing OSGi bundle that will provide the actions.
3.	Configure the elements of the new Custom Action XML file(s) to meet your requirements. Refer to the "Custom Action XML File Construction" section for more information about the elements used in the Custom Action XML file. Alternatively, define Action OSGi services and publish them. Refer to the "Custom Action OSGi Services" section for more information.
4.	Write the code to publish ActionMap containing your actions as OSGi services. This can be done in a bundle activator like in the com.soa.example.console.action bundle, or alternatively can be done using the ActionExporter bean using Spring DM.
5.	Deploy or Redeploy the OSGi bundle containing the Custom Action(s) using the container's Administration Console.

3 Custom Action XML File Construction

This section provides an overview of the Custom Action file location, construction, and sample files.

A Custom Action XML file is composed of the following elements.

Elements	Description	
`ActionMap type`	A parent element that specifies the object type that the Custom Action is associated with. This is a reserved element. Type values that are supported include:	
	`root`	Used for actions performed in the Root Organization.
	`organization`	Used for actions performed in the Sub-Organizations.
	`service`	Used for actions performed in the services folder "Summary" page.
	`container`	Used for actions performed in the container folder "Summary" page.
	`containers`	Used for actions performed on the containers "Details" page.
	`containerlistener`	Used for actions performed in the containers "Inbound Listeners Portlet.
	`alert`	Used for actions performed in all "Alert Portlets" or "Alerts > Monitoring" pages in the "Management Console."
	`consumedcontract`	Used for actions performed on the Contracts > Consumed Contracts "Summary" page.
	`providedcontract`	Used for actions performed on the Contracts > Provided Contracts "Summary" page.
	`contract`	Used for actions performed in the container folder "Summary" page.
	`contracts`	Used for actions performed on the containers "Details" page.
	`dashboard`	Used for actions performed via the "Dashboard" tab.

Elements	Description	
Actions	A child element that includes elements for configuring the properties of a Custom Action.	
	id	An attribute of the "Action" element that is used to specify the internal ID for the Custom Action. This ID is internally referenced by Policy Manager when the action is invoked.
	type	An attribute of the "Action" element that is used to specify the type of entity that the Custom Action will invoke. Two options can be specified (*wizard* or *link*). The "wizard" option is used if the Custom Action will be launching a web application. The "link" option is used if the Custom Action will be referencing and loading a URL address.
	label	An attribute of the "Action" element that is used to specify the name of the action (i.e., text label) that will display on the "Actions Portlet" or "Actions Drop-down List Box."
	decorators	An attribute of the "Action" element that is used to display icons preceding the action. The comma separated value(s) point to image resource(s) relative to the web application root.
	URL	A child element of the "Action" element used to specify how an action is displayed.
	target	An attribute of the "URL" element that is used to specify the window type that the web application or link specified in the Custom Action "type" element will open in. The default is `self`. Other commonly used HTML options include `top` and `blank`.
	path	A child element of the "URL" element that is used to specify the path of the web application or link that the Custom Action will invoke. Within the "path" there can be an "{OBJECT_ID}" placeholder that the system uses to store the UDDI Key for Custom Actions that invoke web applications that are designed to perform actions on a specific Policy Manager object type. If the web application is designed to

Elements	Description	
		perform more general actions and does not perform actions on a specific Policy Manager object type, the Object ID option is not used.
	`WizardProperties`	A child element of the "Action" element that is used to specify the display properties of the "type=wizard" element. Configurable display properties include `left`, `top`, `width`, `height`, `resizable`, `scrollable`, `locationbar`, `menubar`, `statusbar`, and `toolbar`. This option does not apply to the actions of "type=link".
	`Security`	An optional child element of "Action" element that is used to control visibility of action to end users.
	`ResourceType`	An attribute of the "Security" element that is used to specify the Policy Manager object type that the permission will be applied to. Value types supported include `organization`, `policy`, `container`, `contract`, and `service`. *The "`dashboard`" ActionMap type value is associated with administrator system/read security and does not utilize a Security ResourceType.*
	`BusinessAction`	An attribute of the "Security" element that is used to specify the valid permissions for the Custom Action. The default permission is `Read`. Refer to the "Security" section of the "Management Console" (for Root or Sub-Organizations) to determine your specific permission requirements.

The following is a Custom Action sample XML file. It is also available in the META-INF/resources folder of the com.soa.examples.console.action sample named service_action_example.xml.

```
01) <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
02) <Actionmap type="service"
xmlns="http://www.soa.com/uri/products/Actions/xsd/1.0">
03)   <Actions>
04)     <Action id="service.example" type="wizard" label="Test Service (Beta)"
decorators="/actionexample/resources/images/img1.gif,/actionexaxmple/resources/imag
es/img2.gif">
05)       <URL target="_self"
path="/actionexample/action_example_service.jsp?serviceKey={OBJECT_ID}"/>
06)       <WizardProperties left="10" top="10" width="900" height="580"
resizable="no" scrollable="no" locationbar="no" menubar="no" statusbar="yes"
toolbar="no"/>
07)       <Security ObjectKey="{OBJECT_ID}" ResourceType="Service"
BusinessAction="Read"/>
08)     </Action>
09)   </Actions>
10) </Actionmap>
```

In the above example, there is one action (lines 04 – 08) in the action map. It is a wizard as defined by the "type" attribute on line 04. The label in the Portlet will be "Test Service (Beta)" as defined by the "label" attribute. The URL element on line 05 states the URL that will be followed when the action is selected. The {OBJECT_ID} parameter will be replaced with the key of the service the action is displayed for. The Security element on line 07 states the security requirements for initiating the action. In this example the user must have read permissions to the service.

To deploy the actions in the actions file an ActionMapExporter is used. The ActionMapExporter references the actions file and exports it to the user interface framework. Below is a Spring file snippet illustrating the user of the ActionMapExporter.

```
01) <bean class="com.soa.console.features.actions.export.ActionMapExporter" init-
method="export">
02)   <property name="actionMaps>
03)     <list>
04)       <value>/META-INF/resources/actions/service_action_example.xml</value>
05)     </list>
06)   </property>
07) </bean>
```

4 Custom Action OSGi Services

An action can be defined as a simple Java object and then published as an OSGi service. The user interface framework subscribes to all action services and displays them accordingly. The only requirement of the action object is that it implements the com.soa.console.action.Action Interface. A new implementation can be constructed or the com.soa.console.action.impl.ActionImpl class can be used. The following is a Spring-DM file snippet that creates an ActionImpl and then publishes it as an OSGi service.

```
01) <bean class="com.soa.console.action.impl.ActionImpl"
id="organization.action.example">
02)   <property name="decorators">
03)     <value>
04)       /actionexample/resources/images/img1.gif,
/actionexample/resources/images/img2.gif
05)     </value>
06)   </property>
07)   <property name="id" value="organization.example"/>
08)   <property name="type" value="organization"/>
09)   <property name="label">
10)     <value>Example Organizaion Action from OSGi service</value>
11)   </property>
12)   <property name="URL">
13)     <bean class="com.soa.console.action.impl.URLImpl">
14)        <property name="link" value="false"/>
15)        <property name="target" value="_self"/>
16)        <property name="path">
17)         <value>
18) /actionexample/action_example_organization.jsp?organizationKey={OBJECT_ID}
19)         </value>
20)        </property>
21)     </bean>
22)   </property>
23)   <property name="wizardProperties">
24)     <bean class="com.soa.console.action.impl.WizardPropertiesImpl">
25)        <property name="height" value="580"/>
26)        <property name="left" value="10"/>
27)        <property name="locationbar" value="false"/>
28)        <property name="statusbar" value="true"/>
29)        <property name="menubar" value="false"/>
30)        <property name="resizable" value="false"/>
31)        <property name="scrollable" value="false"/>
32)        <property name="toolbar" value="false"/>
33)        <property name="top" value="10"/>
34)        <property name="width" value="900"/>
35)      </bean>
36)   </property>
37)   <property name="security">
38)     <list>
39)        <bean class="com.soa.console.action.impl.SecurityImpl">
40)          <property name="objectKey" value="{OBJECT_ID}"/>
41)          <property name="resourceType" value="Organization"/>
42)         <property name="businessAction" value="Read"/>
43)       </bean>
44)     </list>
45)   </property>
46) </bean>
47)
```

```
48) <osgi:service interface="com.soa.console.action.Action">
49)   <osgi:service-properties>
50)     <entry key="name" value="com.soa.console.action.organization.1"/>
51)   </osgi:service-properties>
52)   <ref local="service.action.example"/>
53) </osgi:service>
```

The ActionImpl object (lines 01 – 46) is similar to the Action element in the Action XML file with only a few differences. The "type" property (line 08) specifies the object type the action is for, "service" or "organization." The indication of whether the action is a link or a wizard is determined by the “link” boolean property (line 14) in the URL property (lines 12 – 22).

For the user interface framework to deploy the action object it must be published as an OSGi service. The ActionImpl object is published using Spring-DM on lines 48 – 53.

5 About SOA Software

SOA Software is a leading provider of unified SOA governance and API Management products that enable organizations to successfully plan, build, and run enterprise services and Open APIs. The world's largest companies including Bank of America, Verizon, and Pfizer use SOA Software solutions to transform their business. For more information, please visit http://www.soa.com.