Oracle® Web Services Manager

Extensibility Guide

10*g* (10.1.3.3.0)

E10300-01

June 2007

ORACLE®

Oracle Web Services Manager Extensibility Guide, 10*g* (10.1.3.3.0)

E10300-01

Primary Author: Laureen Asato, Vrinda Kirloskar

Contents

Preface

This preface provides information on the following topics:

Audience

This manual provides information to programmers and system administrators on how to extend Oracle Web Services Manager (Oracle WSM) with custom steps.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at

`http://www.oracle.com/accessibility/`

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, seven days a week. For TTY support, call 800.446.2398.

Related Documents

For more information, see the following documents in the Oracle Web Services Manager10*g* (10.1.3.3.0) documentation set:

- *Oracle Web Services Manager Administrator's Guide*
- *Oracle Web Services Manager Deployment Guide*
- *Oracle Web Services Manager Quick Start Guide*
- *Oracle Web Services Manager Installation Guide*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

Developing and Deploying Custom Steps

Extensibility provides a developer with the ability to create custom steps. Custom steps are necessary when specific functionality is not provided with the standard policy steps that come with the product. For example, an application may require the use of a type of authentication that is not available in the Oracle WSM product. In this situation, a custom authentication step needs to be created. This manual provides information on how to develop, deploy, and test custom steps.

This chapter contains the following topics:

Custom Step Creation Overview

Oracle Web Services Manager (Oracle WSM) provides the following for creating a custom step:

- The Oracle WSM SDK
- A sample step template for developing a custom step
- The Oracle WSM step template schema

The Oracle WSM custom steps are supported by Java Development Kit (JDK) version 5.0 which comes bundled with the Oracle WSM product.

Complete the following tasks to develop a custom step:

1. Extend the `AbstractStep` class by writing the implementation of the `init`, `execute`, and `destroy` methods, and adding the `set` and `get` methods for the step properties.

   Note: The `execute` method is the only required method for implementing a custom step.

2. Define a step template and provide a unique ID, name, and configurable parameters for the step.
3. Deploy the custom step.

Developing a Custom Step

A custom step extends the `com.cfluent.policysteps.sdk.AbstractStep` class, implements the `execute` method, and may override the `init` and `destroy` methods. The following sections contain information about the Abstract step; the `execute, init,` and `destroy` methods; and the `IResult` states returned by the `execute` method.

About the execute Method

The basic characteristics and functionality of the `execute` method are as follows:

- Provides the program entry point for the custom step
- Can throw an exception of type `com.cfluent.policysteps.sdk.Fault`

  If a Fault is thrown by the `execute` method during exception processing, this Fault is cascaded back to the client application.
- Returns an object of type `IResult`

About Init and Destroy

To override the default `init` method, add initialization code specific to that custom step. You can also add code to the `destroy` method to ensure a clean end to the lifecycle of the custom step.

About IResult States

The custom step generates the `result` object using the `createResult` method. The int argument for the `createResult` method provides the state of the result. These are the possible states for the `execute` method:

- `IResult.FAILED` – The `execute` method failed.
- `IResult.SUCCEEDED` – The `execute` method succeeded.
- `IResult.SUSPENDED` – The `execute` method suspended execution.

> **Note:** Because the `execute` method is multithreaded (that is, it can be executed by multiple threads simultaneously), you should ensure that the implementation of the `execute` method is thread safe.

About the AbstractStep Class

The AbstractStep class provides the following default methods:

```
public final void setEnabled(boolean enabled)
public final boolean getEnabled()
public final void setStepName(String stepName)
public final String getStepName()
public void setFaultCodes(String[] faultCodes)
public String[] getFaultCodes()
public final AgentContext getAgentContext()
public final void setAgentContext(AgentContext context)
public abstract IResult execute(IMessageContext messageContext)
public void init()
protected void generateFault(String message, String qualifier, String actor,
Detail detail)
protected void generateFault (Fault fault)
```

```
protected IResult createResult (int status)
public void destroy()
```

Setting Outgoing HTTP Headers in a Custom Step

Oracle WSM allows you to configure a custom step to set outgoing custom HTTP headers in a gateway. You can perform this configuration by adding the following code to your custom step (Example 1–1):

Example 1–1 Setting Outgoing HTTP Headers in a Custom Step

```
HashMap httpHeaders = new HashMap();
httpHeaders.put("myheader1", "headervalue1");
httpHeaders.put("myheader2", "headervalue2");
messageContext.setProperty("customHttpHeaders", httpHeaders);
```

Modifying the Content of SOAP Messages in a Custom Step

Oracle WSM allows you to add, delete, or modify the contents of SOAP messages in a custom step. You can perform this configuration by adding the following code to your custom step (Example 1–2):

Example 1–2 Modifying the Content of SOAP Messages in a Custom Step

```
public IResult execute(IMessageContext messageContext) throws Fault {
    MessageContext msgCtxt = (MessageContext) messageContext;
    SOAPEnvelope senv = msgCtxt.getRequest().getAxisMessage().getSOAPEnvelope();
    ((org.apache.axis.message.SOAPEnvelope)senv).setDirty(true);
    Name header = factory.createName("CustomHeader",
"http://foo.com/custom/header");
    SOAPHeaderElement headerElement = senv.getHeader().addHeaderElement(header);
    headerElement.addTextNode("My custom header data");
    ...
}
```

Adding Logs to a Custom Step

To add logs to a custom step that are written to the log file of the gateway or the agent, add the following code to your custom step (Example 1–3):

Example 1–3 Adding Logs to a Custom Step

```
import com.cfluent.ccore.util.logging.*;

public class MyCustomStep {
  private static String CLASSNAME = MyCustomStep.class.getName();
  private static ILogger LOGGER = LogManager.getLogger(CLASSNAME);

  public IResult execute(IMessageContext messageContext) throws Fault {
    LOGGER.entering(CLASSNAME, "execute");
    ...
    LOGGER.log(Level.SEVERE, "This is a {0} log", "severe");
    if(LOGGER.isLoggable(Level.FINEST))
      LOGGER.log(Level.FINEST, "This is a {0} log", "finest");
    ...
  }
}
```

Log levels can be Level.SEVERE, Level.WARNING, Level.INFO, Level.FINE, Level.FINER, or Level.FINEST, which are listed in order from most severe to trace.

Defining a Custom Step Template

The Oracle WSM policy step framework requires that each Oracle WSM step refer to a step template. The step template defines the step ID and the step properties, and provides a brief description of each property.

To add a step template to the Oracle WSM framework, you must create a well-formed XML document. Refer to Example A–2 in Appendix A, "Custom Step Source Code and Step Template", when creating your XML sample step template. Refer to Appendix B, "Step Template Schema", for the step template schema.

Also refer to the `POLICY_MANAGER_OBJECTS.xml` and `POLICY_MANAGER_OBJECTS_4.0.xml` samples that come with the product. These sample files can be found in the following location: *`ORACLE_HOME`*`/config/db/common/PolicyRespository`

Table 1–1 contains descriptions of the tags for a typical step template:

Table 1–1 Step Template Tags

Tags	Description
`csw:StepTemplate name=`*`Step_Name`* `id=`*`XYZA123`*	Start tag of the XML document that defines the step name and a unique ID. The Oracle WSM system does not assign a unique ID—you must assign one. If the ID you assign is not unique to the system, Oracle WSM throws an exception.
`csw:StepTemplate package=?`	Defines the package name of the step implementing class. This is a required attribute.
`csw:StepTemplate timestamp=?`	Defines the time stamp of the creation of the step. This is a required attribute.
`csw:StepTemplate version=?`	Defines the step template version. This is an optional attribute.
`StepTemplate /csw:`*`Description`*	Provides a brief description of the step. This is an optional element.
`StepTemplate/csw:`*`Implementation`*	Identifies which class contains the implementation of the step. The framework loads the class defined by this step when the step is invoked, at runtime, by reflection. This is a required element.
`StepTemplate/csw:`*`Faults`*	Specifies the list of faults for the step. This is an optional element.
`StepTemplate/Faults/csw:`*`Fault/`*	Specifies the fault to be generated.
`StepTemplate/csw:PropertyDefinitions`	Defines the list of all the property sets for the step.
`StepTemplate/PropertyDefinitions/csw:PropertyDefinitionSet`	Defines each and every property set for the step.
`StepTemplate /PropertyDefinitions/PropertyDefinitionSet/csw:PropertyDefinition`	Defines the properties within a property set for the step.

Table 1–1 (Cont.) Step Template Tags

Tags	Description
`StepTemplate /PropertyDefinitions/ PropertyDefinitionSet/ csw:PropertyDefinition name=?`	The name of the property. This is a required attribute.
`StepTemplate/PropertyDefinitions/ PropertyDefinitionSet/ csw:PropertyDefinition type=?`	Defines the data type of a step property. Data types can be any of the following: boolean, string, boolean, int, long, float, short, string[], boolean[], int[], long[], short[], string[]. This is a required attribute.
`StepTemplate/PropertyDefinitions/ PropertyDefinitionSet/ csw:PropertyDefinitionisMultivalued=?`	Specifies if the property can hold multiple values or not. Possible values are "true" or "false".This is an optional attribute. The default value is “false”.
`StepTemplate/PropertyDefinitions/ PropertyDefinitionSet/ csw:PropertyDefinitionisRequired=?`	Specifies if the property is required or optional. Possible values are "true" or "false". This is an optional attribute. The default value is “false”.
`StepTemplate/PropertyDefinitions/ PropertyDefinitionSet/csw:Property Definition displayType=?`	Specifies how the property is to be displayed to the user in the graphical user interface (GUI). This is an optional field. The default value is “text”.
`StepTemplate/PropertyDefinitions/ PropertyDefinitionSet/csw:Property Definition displayType=”text”`	Displays a text box where the property value can be entered.
`StepTemplate/PropertyDefinitions/ PropertyDefinitionSet/csw:Property Definition displayType=”password”`	Displays a text box where the property value can be entered, but the value entered is not visible to the user in clear text.
`StepTemplate/PropertyDefinitions/ PropertyDefinitionSet/csw:Property Definition displayType=”checkbox”`	Requires providing enumeration values. Multivalue property should be set to true; if nothing is specified it is set, by default, to false. If one of these conditions is not met, a text box is displayed.
`StepTemplate/PropertyDefinitions/ PropertyDefinitionSet/csw:Property Definition displayType=”radio”`	Requires providing enumeration values. Multivalue property if specified should be set to false, if nothing specified it is set by default to false. If one of these conditions are not met a text box is displayed.
`StepTemplate/PropertyDefinitions/ PropertyDefinitionSet/csw:Property Definition displayType=”dropdown”`	Displays a drop-down list. Requires specifying enumeration values. If one condition fails, a text box is displayed.
`StepTemplate/PropertyDefinitions/ PropertyDefinitionSet/csw:Property Definition displayType=”textarea”`	Displays a text area where the property value can be specified. You can also specify the number of rows that span this text area. For example: textarea(rows=5) creates a text area spanning 5 rows. If the rows parameter is not specified, a text area spanning 3 rows is created by default.
`StepTemplate/PropertyDefinitions/ PropertyDefinitionSet/csw:Property Definition displayType=”hidden”`	This property is not visible to the user.
`StepTemplate/PropertyDefinitions/ PropertyDefinitionSet/csw:Property Definition displayType=”immutable”`	Specifies a property value that cannot be changed by the user.

Table 1–1 (Cont.) Step Template Tags

Tags	Description
`StepTemplate/PropertyDefinitions/ PropertyDefinitionSet/csw:Property DefinitiondisplayType="immutablePa ssword"`	Specifies a property value that cannot be changed by the user. This value is not visible to the user in clear text.
`StepTemplate/PropertyDefinitions/ PropertyDefinitionSet/PropertyDefi nition/csw:DisplayName`	Specifies the name of the property to be displayed to the user.
`StepTemplate/PropertyDefinitions/ PropertyDefinitionSet/PropertyDefi nition/csw:Description`	Provides a brief description of the property defined by the start tag. You can provide a description for the property being defined.
`StepTemplate/PropertyDefinitions/ PropertyDefinitionSet/PropertyDefi nition/DefaultValue/csw:Absolute`	Defines the default value(s) for the property being defined. The value may be Absolute or PropertyRef. A PropertyRef indicates to the framework to carry over an environment property, defined by ${<property_key>}. Currently only Absolute type is supported.
`StepTemplate/PropertyDefinitions/ PropertyDefinitionSet/PropertyDefi nition/csw:SimpleType`	Defines a simple type that determines the constraints on and information about the values of attributes or elements with text-only content. This is an optional element. Only one such element can be specified.
`StepTemplate/PropertyDefinitions/ PropertyDefinitionSet/PropertyDefi nition/csw:SimpleType name=?`	Defines the name of the SimpleType. This is a required attribute.
`StepTemplate/PropertyDefinitions/ PropertyDefinitionSet/PropertyDefi nition/SimpleType/csw:Restriction`	Restrictions are used to define acceptable values for elements or attributes.
`StepTemplate/PropertyDefinitions/ PropertyDefinitionSet/PropertyDefi nition/SimpleType/csw:Restriction base=?`	Defines the data type of the restriction values. This is a required attribute. Data types can be any of the following: boolean, string, boolean, int, long, float, short, string[], boolean[], int[], long[], short[], string[]. This is a required attribute.
`StepTemplate/PropertyDefinitions/ PropertyDefinitionSet/PropertyDefi nition/SimpleType/Restriction/ csw:length`	Specifies the exact number of characters or list items allowed. Must be equal to or greater than zero.
`StepTemplate/PropertyDefinitions/ PropertyDefinitionSet/PropertyDefi nition/SimpleType/Restriction/ csw:minLength`	Specifies the minimum number of characters or list items allowed. Must be equal to or greater than zero.
`StepTemplate/PropertyDefinitions/ PropertyDefinitionSet/PropertyDefi nition/SimpleType/Restriction/ csw:maxLength`	Specifies the maximum number of characters or list items allowed. Must be equal to or greater than zero.
`StepTemplate/PropertyDefinitions/ PropertyDefinitionSet/PropertyDefi nition/SimpleType/Restriction/ csw:enumeration`	Defines a list of acceptable values pattern and the exact sequence of characters that are acceptable.
`StepTemplate/PropertyDefinitions/ PropertyDefinitionSet/PropertyDefi nition/SimpleType/Restriction/ csw:totalDigits`	Specifies the exact number of digits allowed. Must be greater than zero.

Table 1–1 (Cont.) Step Template Tags

Tags	Description
`StepTemplate/PropertyDefinitions/PropertyDefinitionSet/PropertyDefinition/SimpleType/Restriction/csw:fractionDigits`	Specifies the maximum number of decimal places allowed. Must be equal to or greater than zero.
`StepTemplate/PropertyDefinitions/PropertyDefinitionSet/PropertyDefinition/SimpleType/Restriction/csw:minInclusive`	Specifies the lower bounds for numeric values (the value must be greater than or equal to this value).
`StepTemplate/PropertyDefinitions/PropertyDefinitionSet/PropertyDefinition/SimpleType/Restriction/csw:maxInclusive`	Specifies the upper bounds for numeric values (the value must be less than or equal to this value).
`StepTemplate/PropertyDefinitions/PropertyDefinitionSet/PropertyDefinition/SimpleType/Restriction/csw:minExclusive`	Specifies the lower bounds for numeric values (the value must be greater than this value).
`StepTemplate/PropertyDefinitions/PropertyDefinitionSet/PropertyDefinition/SimpleType/Restriction/csw:maxExclusive`	Specifies the upper bounds for numeric values (the value must be less than this value).
`StepTemplate/PropertyDefinitions/PropertyDefinitionSet/PropertyDefinition/SimpleType/Restriction/csw:whiteSpace`	Specifies how white space (line feeds, tabs, and carriage returns) is handled.

Deploying a Custom Step

Complete the following tasks to deploy a custom step:

1. Add the custom step to the Oracle WSM Policy Manager.
2. Add the minimum Java Archive (JAR) files required to compile the custom step. See "To add the JAR files" on page 1-8 for more information about adding or copying the JAR files.
3. Add the JAR files containing the implementation classes for the step to the `/lib/custom/` directory. See "To add the JAR files" on page 1-8 for more information about adding or copying the JAR files.

> **Note:** When you copy a new JAR file, or make changes to an existing JAR file, you must restart Oracle Application Server.

4. Add the custom step to the policy pipeline.
5. Add a help page for the custom step.

To add a custom step

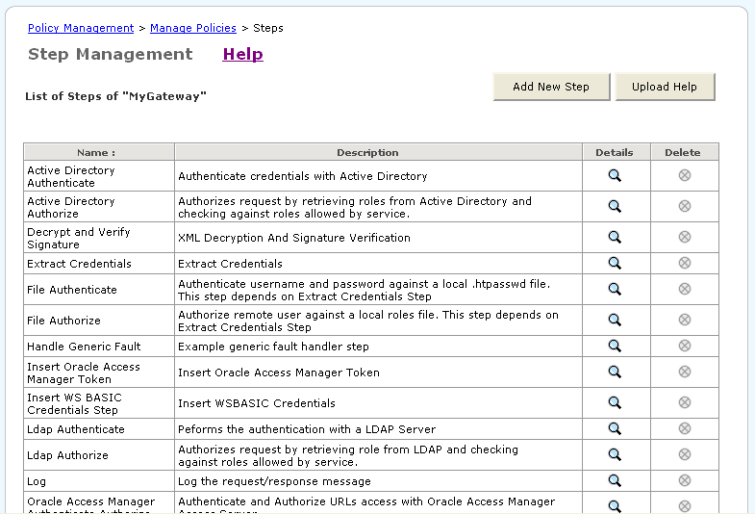
1. Log in to the Web Services Manager Control and select **Policy Management**, then select **Manage Policies.**

   The Web Services Manager Control displays the list of registered components.

2. In the row for the component where you want to add a custom step, click **Steps**.

   The Step Management page displays a list of existing steps (Figure 1–1).

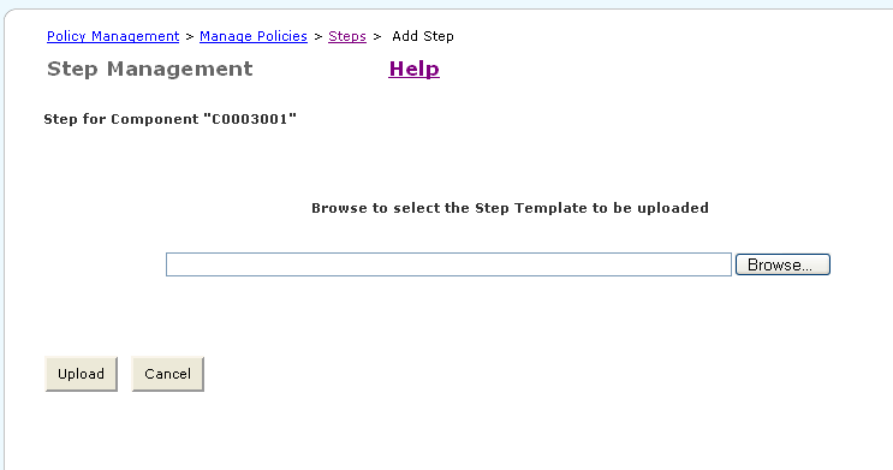
Figure 1–1 List of Policy Steps



3. Click **Add New Step**.

   The Web Services Manager Control displays the Step Management/Add Step page (Figure 1–2).

Figure 1–2 Step Management/ Add Step Page



4. Click **Browse** and select the newly created XML step template.
5. Click **Upload**.

   The Step Management page updates the list of currently available steps, including the newly added step.

> **Note:** There is a delete button for each custom step; however, default steps cannot be deleted.

To add the JAR files

The following two Java Archive (JAR) files are required to compile the custom step java code. Copy the files to the following locations:

ORACLE_HOME/owsm/lib/extlib/saaj.jar

ORACLE_HOME/owsm/lib/coresv-4.0.jar

Additionally, you need to copy the Java Archive (JAR) containing the implementation classes for the custom step to the computer where the gateway or agent component is installed. Copy the JAR files to the following location:

ORACLE_HOME/owsm/lib/custom

To add a step to a policy pipeline

1. From Web Services Manager Control, click **Policy Management**, then click **Manage Policies.**

   Web Services Manager Control displays the list of registered Oracle WSM components.

2. Click **Policies** for the component whose policy you want to modify.

   Web Services Manager Control displays the policies for the component.

3. Click the **Edit** icon for the policy you want to edit.

   The page displays the existing policy pipeline.

4. In the section of the pipeline immediately above where you want to add a step, click **Add Step Below**.

   A list displays the steps available for the selected component, including your new custom step.

5. Select the new step and click **Ok**.

6. Configure the step.

7. Click **Next**.

   The step is added to the policy pipeline.

8. Click **Save**.

9. Commit the changes to the policy by clicking **Commit Policy**.

> **Note:** The new policies go into effect in 10 seconds, if you use the default polling frequency. The gateway and agent components automatically retrieve the updated policies from the Oracle WSM Policy Manager.

To add a help page for a custom step

If you have created a custom step and want to add a help page for it, perform the following steps:

1. Navigate to the *OC4J_HOME*/owsm/lib/app/ccore/help/steps directory.

2. Create the HTML help page for the custom page in this directory. Name the file with the step name, replacing any spaces with underscores.

   For example, if the step name is "My Custom Step" then name the help page "My_Custom_Step.html."

   You can optionally create help pages in other languages. Place these help pages in directories according to the predefined language abbreviations, such as *es* (Spanish), *fr* (French), *de* (German), *it* (Italian), and *jp* (Japanese). These

abbreviations can be located in the language preference settings of a Web browser client.

3. Start the OC4J server.
4. Navigate to the *OC4J_HOME*/owsm/bin directory and perform the following command from the command prompt:

   ```
   wsmadmin deploy control
   ```

   This command repackages the Web Services Manager application with the newly created help page, and redeploys it to the OC4J container.
5. Log in to the Web Services Manager Control and verify that help for your custom step is available.

2

Understanding the Sample Custom Step

This chapter describes the custom authentication step which is provided as a sample with Oracle Web Services Manager (Oracle WSM) Release 3 (10.1.3.1), and describes procedures to compile, deploy, and run the step. Appendix A, "Custom Step Source Code and Step Template" contains the source code (Example A–1) and the step template (Example A–2) for this sample.

This chapter contains the following topics:

- Understanding the Custom Authentication Step
- Compiling and Running the Custom Authentication Step
- Checking Monitoring Data

Understanding the Custom Authentication Step

Oracle WSM Release 3 (10.1.3.1) includes a custom authentication step. The custom step authenticates the user against a user name and password that must be configured in the policy step pipeline. This step can be configured either in the gateway or the agent policy pipeline.

The sample step demonstrates the use of the following functions from the application programming interface (API):

- Stepping Through the Lifecycle Using the init, execute, and destroy Methods
- Accessing Data from the Message Context and Logging the Data
- Setting Result Status and Authenticated User Name in the Message Context
- Setting the Invocation Status for Operational Statistics
- Custom Step Template
- Defining Fault Codes in the Custom Step Template

Stepping Through the Lifecycle Using the init, execute, and destroy Methods

The policy is initialized when it is created for the first time or when changes are made to it and the policy is committed. The `init` method of a step is called when the policy is initialized. You can put any initialization code in the `init` method. In the source code for the sample application, the `init` method creates the PrintWriter object. This object is used by the `execute` method to log messages.

The `execute` method is called each time a message arrives. This method contains the main logic, accessing various parts of the message from the message context and logging the information to a file. You must make sure that this method is thread-safe

because multiple requests calling this function may be simultaneously executed by multiple threads.

The `destroy` method performs a clean ending to the step's lifecycle. This method is called when a policy is updated or during a graceful shutdown of the server.

The source code for the sample application demonstrates how to use the PrintWriter stream with this method.

Accessing Data from the Message Context and Logging the Data

The `execute` method takes an object of type `IMessageContext` as a parameter. This object contains all the information about the message. It also has access to protocol-specific objects to retrieve protocol-specific information.

The source code for our sample application demonstrates retrieving the user name and password from a SOAP message sent as clear text inside a WS-Security header. For more information, refer to the WS-Security specification.

The client is required to send the user name and password inside the SOAP header. The code retrieves the user name and password from this header after retrieving the SOAP message from the message context. This is one example of sending and retrieving the user name and password; other methods include sending the user name and password in an HTTP header as basic authentication, or sending them in a SOAP message as a custom header.

- Processing Stage Information

  You can receive processing stage information using the API method `messageContext.getProcessingStage().`

- Client IP address

  You can retrieve Client IP information from `messageContext` by first retrieving the `HttpServletRequest` object and then calling the `getHeader` function on this object.

```
HttpServletRequest httpServletRequest = (HttpServletRequest)

messageContext.getProperty("javax.servlet.request");

String remoteAddr = httpServletRequest.getHeader("Host");
```

  Note: The source code assumes that the request is being sent through HTTP. The Oracle WSM Test Web Service page also uses HTTP to send requests.

- User locale

  You can receive client user locale information using the API method `messageContext.getUserLocale.` This can be used to generate locale-specific faults or responses.

- SOAP message processing stage information

  You can receive processing stage information using the API method `messageContext.getRequestMessage.`

- SOAP message user credentials

Setting Result Status and Authenticated User Name in the Message Context

The source code for the sample application initializes the result object with a `FAILED` status:

```
result.setStatus(IResult.FAILED);
```

If the user is authenticated, the result status is set to `SUCCEEDED`:

```
result.setStatus(IResult.SUCCEEDED);
```

Setting the Invocation Status for Operational Statistics

Custom steps need to set the invocation status only if the step involves authentication or authorization. The source code for the sample application sets one of the following authentication statuses on the `InvocationStatus` object of the message context.

```
messageContext.getInvocationStatus().setAuthenticationStatus(InvocationStatus.FAILED)
```

```
messageContext.getInvocationStatus().setAuthenticationStatus(InvocationStatus.SUCCEEDED)
```

The Oracle WSM Monitor uses this information to display authentication graphs or bar charts in the Web Services Manager Control.

The `InvocationStatus` class includes `setInvocationStatus` which sets overall status and `setServiceStatus` which sets service status. These functions are handled by the Oracle WSM runtime; however, the authentication or authorization result is known only by the step implementation and needs to be set in the step itself.

Custom Step Template

Example A–2 in Appendix A, "Custom Step Source Code and Step Template" lists the step template for the custom authentication step, `CustomAuthenticationStep.xml`.

Defining displayType Properties in the Custom Step Template

The `displayType` has been set to "password" type (`displayType="password"`, shown in bold in Example 2–1) for the password property. When configuring this step with a user name and password, the text entered in the password field is not displayed in clear text. The text is obfuscated and displayed as a series of asterisks (*). For password security, the password display type is recommended for all fields in your step.

Example 2–1 Defining displayType Properties in the Custom Step Template

```
      <csw:PropertyDefinition name="Password" type="string" isRequired="true"
displayType="password">
        <csw:DisplayName>Password</csw:DisplayName>
        <csw:Description>Password used for authentication</csw:Description>
        <csw:DefaultValue>
          <csw:Absolute>test</csw:Absolute>
        </csw:DefaultValue>
      </csw:PropertyDefinition>
```

Defining Fault Codes in the Custom Step Template

You can define the fault code that the custom step throws inside the step definition file of the step. In the custom authentication step, the step throws the *AuthenticationFault*

fault code. Define the fault code in the step template, `CustomAuthenticationStep.xml`, as follows:

```
<csw:Faults>
  <csw:Fault xmlns:fns="http://www.example.com/ws/Faults">
fns:AuthenticationFault</csw:Fault>
</csw:Faults>
```

Custom Authentication Step Source Code

Example A–1 in Appendix A, "Custom Step Source Code and Step Template" provides the code for the custom authentication step sample.

The file containing this code, `CustomAuthenticationStep.java`, can be found in the *ORACLE_HOME*`\owsm\samples\customsteps\customAuthenticationStep` directory.

Compiling and Running the Custom Authentication Step

This section describes the procedures to compile and run the custom authentication step. It includes the following topics:

- Compiling the Custom Authentication Step
- Deploying the Custom Authentication Step Template
- Adding the Custom Authentication Step to a Policy
- Testing the Custom Authentication Step

Compiling the Custom Authentication Step

The *ORACLE_HOME*`/owsm/samples/customsteps/` directory contains the following files:

- `CustomAuthenticationStep.java` – This file contains the source code for the custom step.
- `CustomAuthenticationStep.xml` – This is the custom step template file.
- `build.xml` – This is an Ant build script used to build the JAR file.

Perform the following steps to compile the custom authentication step and copy it to the directory where it will be run:

1. Set the *ORACLE_HOME* environment variable.
2. Enter the following commands to navigate to the directory that contains the custom authentication step and compile it:

   ```
   cd ORACLE_HOME/owsm/samples/customsteps/
   customAuthenticationStep
   ```

   ```
   ant -f build.xml
   ```

   These commands generate the `customAuthenticationStep.jar` file.

Deploying the customAuthenticationStep.jar File

Perform the following step to deploy the `customAuthenticationStep.jar` file:

- Copy the `customAuthenticationStep.jar` file to the *ORACLE_HOME*`/owsm/custom/ directory`.

Deploying the Custom Authentication Step Template

Perform the following steps to deploy the step template:

1. Log in to the Web Services Manager Control.
2. Follow the procedure described in "Defining a Custom Step Template", and add the `customAuthenticationStepTemplate.xml` step template.
3. Verify that you can view the custom authentication step in the list of available steps.
4. Restart Oracle Application Server.

> **Note:** When you copy a new jar file, or make changes to an existing jar file, you must restart Oracle Application Server.

Adding the Custom Authentication Step to a Policy

Perform the following steps to configure the step for use in a policy:

1. Add the step to the gateway or agent policy.
2. Configure the step by setting a user name and password as desired.

   For example, you can enter *owsmadmin* as the user name and *oracle* as the password.
3. Save and commit the policy.

   You are now ready to execute the policy.

Testing the Custom Authentication Step

To test the step, you can use any client that can generate a WS-Security header, or you can write your own client. Oracle WSM provides the Test Web Service page for generating a WS-Security header and performing the test. For more information on using the Test Web Service page refer to *Oracle Web Services Manager Administrator's Guide*.

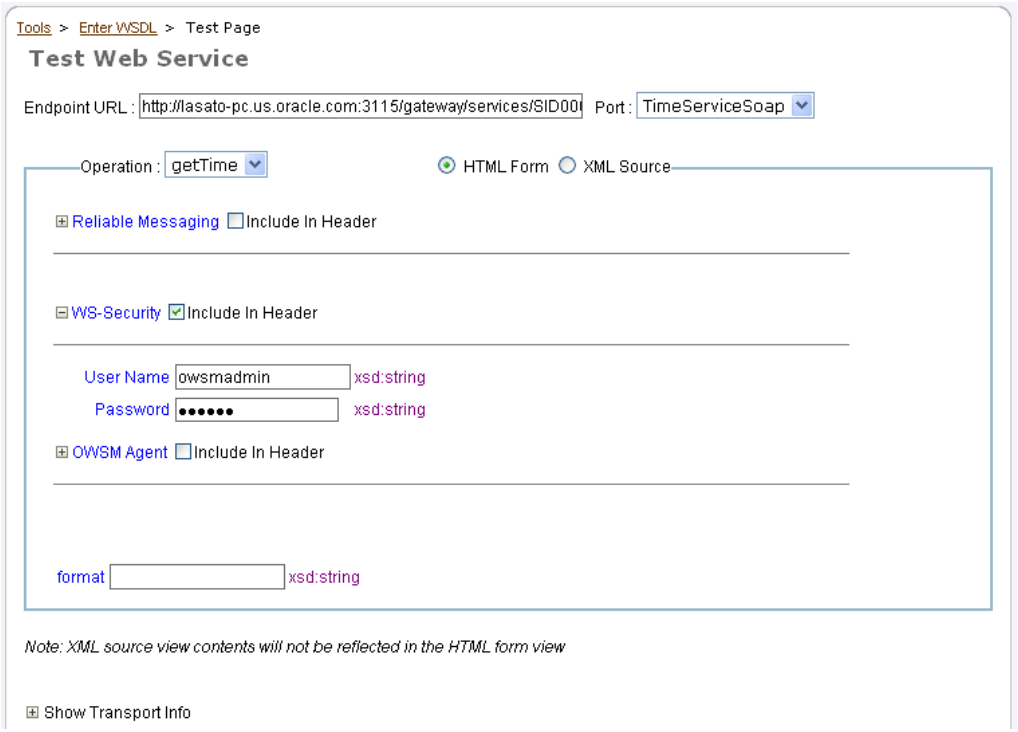
Perform the following steps to send messages and execute the policy:

1. Send a message without a security header (do not select the WS-Security "Include in Header" box or enter the values for User Name and Password.).

   The step gets executed and you see a "No Username supplied" fault message.
2. Perform the following steps to send a message with the user name and password in the SOAP header.
   - Access the Test Web Service page by selecting **Tools**, then **Test Page** from the navigation pane of Web Services Manager Control.
   - Enter the WSDL URL and click **Submit Query**.
   - When the page displays additional parameters (Figure 2–1), select the WS-Security *Include in Header* box and enter the values for User Name and Password.

     This allows a SOAP header to be inserted in the SOAP envelope.

Figure 2–1 Test Web Service Page with Additional Parameters



- Click **Invoke** to send the message as a SOAP request to the JAX-RPC Web service end point.
- Enter an incorrect password and observe the response.

The Test Result page displays the response from the service and generates the security header shown in Example 2–2:

Example 2–2 WS-Security Header

```
<soap:Header>
  <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/
oasis-200401-wss-wssecurity-secext-1.0.xsd">
    <wsse:UsernameToken>
      <wsse:Username>owsmadmin</wsse:Username>
      <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/
oasis-200401-wss-username-token-profile-1.0#PasswordText">oracle</wsse:Password>
    </wsse:Username Token>
  </wsse:Security>
</soap:Header>
```

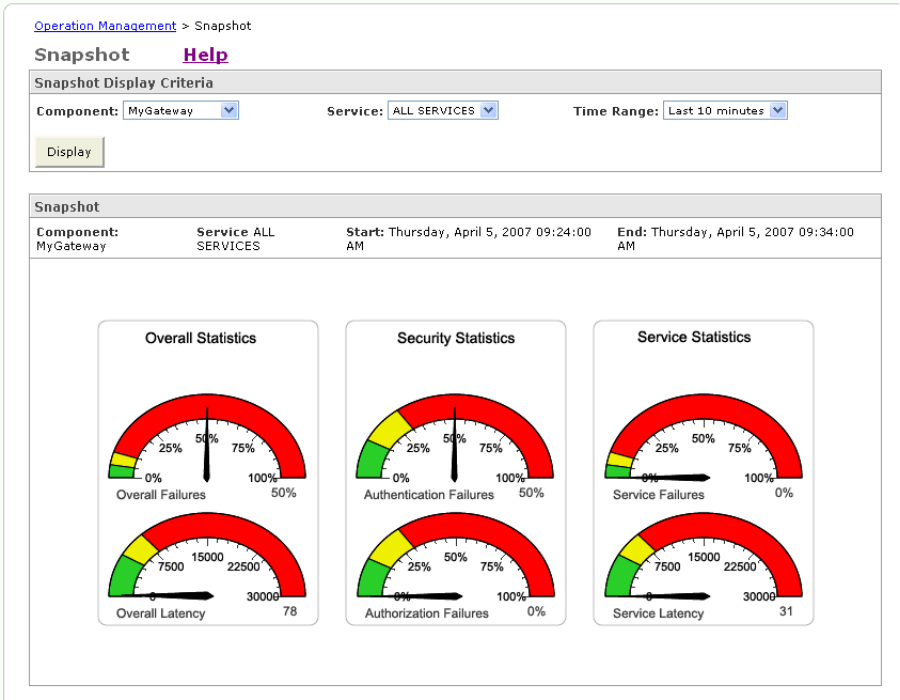
Note: In Example 2–2 the password is in clear text. For production environments you can use a mechanism such as #PasswordDigest instead of #PasswordText, as described in the WS-Security User Name Token profile.

Checking Monitoring Data

After you have sent the test messages, you can examine monitoring data from Web Services Manager Control to view authentication status and overall status:

1. From the navigation pane of Web Services Manager Control, click **Operational Management**, then click **Snapshot** (Figure 2–2).

Figure 2–2 Oracle WSM System Snapshot



2. Examine the Security Statistics authentication graph, and the rate of authentication failures.
3. Check the overall status of the Web services and view detailed information on Web services execution by clicking **Overall Statistics**.

For more information on viewing and managing performance of Web services, refer to Chapter 6, "Monitoring Oracle Web Services Manager" in *Oracle Web Services Manager Administrator's Guide*.

3

Policy Steps SDK Interfaces & Methods

This chapter contains reference information for the Oracle Web Services Manager (Oracle WSM) Step Software Development Kit (SDK). The Oracle WSM Step SDK contains the following interfaces and classes:

- IStep
- AbstractStep
- IContext
- IMessageContext
- IResult
- Result
- AgentContext
- InvocationStatus
- Fault

IStep

```
Interface IStep
```

Known Implementing Classes

`AbstractStep`

Declaration

```
public interface IStep
```

Description

A step is a basic execution block in the pipeline. A step is a Java Bean that is configured using the Java Beans framework. The `AbstractStep` class is an abstract class that provides partial implementation of this interface. A step may use resource pooling mechanisms to efficiently handle multiple requests, but it should not keep any per-invocation state.

Table 3–1 Methods of IStep

Method	Description
`setEnabled()`	Enables or disables this step.
`getEnabled()`	Gets the boolean value from setEnabled.

Table 3–1 (Cont.) Methods of IStep

Method	Description
`setStepName()`	Sets the name of this step.
`getStepName()`	Gets the step name.
`getAgentContext()`	Gets context information for the policy enforcement point.
`setAgentContext()`	Sets the agent context.
`execute()`	Executes this step.
`init()`	This method is called after the Java bean is created or after a Java bean property has changed.
`destroy()`	Ensures a clean ending to the step lifecycle.
`getFaultCodes()`	Gets the fault codes that are thrown by this step.
`setFaultCodes()`	Sets the fault codes that are thrown by this step.

setEnabled()

Declaration

```
public void setEnabled(boolean enabled)
```

Description

Enables or disables this step.

Parameters

`Enabled` – If true, the step is enabled.

getEnabled()

Declaration

```
public boolean getEnabled()
```

Description

Gets the boolean value from `setEnabled`.

Returns

One of the boolean values – true or false. If the step is enabled, this method returns the value, true.

setStepName()

Declaration

```
public void setStepName(java.lang.String stepName)
```

Description

Sets the name of this step.

Parameters

`stepName` – Name of the step.

getStepName()

Declaration

```
public java.lang.String getStepName()
```

Description

Gets the step name.

Returns

The name of this step.

getAgentContext()

Declaration

```
public AgentContext getAgentContext()
```

Description

Gets context information for policy enforcement point.

setAgentContext()

Declaration

```
public void setAgentContext(AgentContext context)
```

Description

Sets the agent context.

Parameters

`context` (of type `AgentContext`) – Represents the information of an agent during the pipeline execution.

execute()

Declaration

```
public IResult execute(IMessageContext context)
     Throws Fault
```

Description

Executes this step.

Parameters

`context` – message context containing the request and response.

Returns

`IResult` – On successful execution of this method.

`Fault` – Fault type exception, if it gets thrown.

init()

Declaration

```
public void init()
     throws java.lang.IllegalStateException
```

Description

This method is called after the Java bean is created or after a Java bean property has changed.

Throws

`java.lang.IllegalStateException`

destroy()

Declaration

```
public void destroy()
     throws java.lang.IllegalStateException
```

Description

Use the `destroy` method to ensure a clean ending to the step lifecycle.

setFaultCodes()

Declaration

```
void setFaultCodes(String[]faultCodes)
```

Description

Set the fault codes that are thrown by this step.

Parameters

`faultCodes` (of type `String[]`) – Fault codes thrown by the step.

getFaultCodes()

Declaration

```
String[]getFaultCodes()
```

Description

Get the fault codes that are thrown by this step.

Returns

The string array of the fault codes thrown by this step.

AbstractStep

`com.cfluent.policysteps.sdk.AbstractStep`

Declaration

```
public abstract class AbstractStep
```

extends

`java.lang.Object`

implements

`IStep`

Description

A default implementation of `IStep`. The implementation should at least override the `IStep` `execute` method.

Table 3–2 Methods of AbstractStep

Method	Description
`createResult()`	Called after the Java bean is created or after a Java bean property has changed.
`setEnabled()`	If set to true, the step is enabled.
`getEnabled()`	Returns the value of setEnabled; true if the step is enabled, otherwise false.
`setStepName()`	Sets the name of this step.
`getStepName()`	Gets the step name.
`getAgentContext()`	`getAgentContext` in interface `IStep`
`setAgentContext()`	Sets the agent context.
`execute()`	Throws `FaultExecute` for this step. Step implementations should override this method to perform desired tasks.
`init()`	Throws `java.lang.IllegalStateException`. This method is called after the Java bean is created or after a Java bean property has changed.
`getFaultCodes()`	Gets the fault codes thrown by this step.
`setFaultCodes()`	Sets the fault codes thrown by this step.
`destroy()`	Ensures a clean ending to the step lifecycle.
`generateFault()`	Generates faults for this step.
`createResult()`	Creates the result for this step.

createResult()

Declaration

```
protected IResult createResult(int status)
```

Parameters

`int status`

Description

This method is called after the Java bean is created or after a Java bean property has changed.

Table 3–3 Fields for createResult()

Variable	Description
m_name	Protected java.lang.String m_name.
m_enabled	Protected boolean m_enabled.
m_context	Protected AgentContext m_context.

setEnabled()

Declaration

```
public final void setEnabled(boolean enabled)
```

Description

Enables or disables this step.

Parameters

`boolean enabled`

Specified by

`setEnabled in interface IStep`

Parameters

`enabled` – If true, the step is enabled.

getEnabled()

Declaration

```
public final boolean getEnabled()
```

Description

Checks to see if the step is enabled.

Specified by

`getEnabled in interface IStep`

Returns

One of the boolean values – true or false. If the step is enabled, this method returns the value, true

setStepName()

Declaration

```
public final void setStepName(java.lang.String stepName)
```

Description

Sets the name of this step.

Specified by
`setStepName in interface IStep`

Parameters
`stepName` – Name of the step.

getStepName()

Declaration
```
public final java.lang.String getStepName()
```

Description
Gets the step name.

Specified by
`getStepName` in interface `IStep`

Returns
The name of this step.

getAgentContext()

Declaration
```
public final AgentContext getAgentContext()
```

Description
Retrieves the agent context for the step.

Parameters
`context`

Specified by
`getAgentContext` in interface `IStep`

setAgentContext()

Declaration
```
public final void setAgentContext(AgentContext context)
```

Description
Sets the agent context.

Specified by
`setAgentContext` in interface `IStep`

execute()

Declaration

```
public abstract IResult execute(IMessageContext messageContext)
```

Description

Executes the step. Step implementations should override this method to perform desired tasks.

Specified by

`execute` in interface `IStep`.

Parameters

`messageContext` – Message context containing the request and response.

Returns

Depending on result of execution, can return `IResult` or `Fault`.

init()

Declaration

```
public void init()
```

Description

Throws `java.lang.IllegalStateException`. This method is called after the Java bean is created or after a Java bean property has changed.

Specified by

`init` in interface `IStep`.

Throws

`java.lang.IllegalStateException`.

generateFault()

Declaration

```
protected void generateFault(Fault fault)
```

Description

Generates faults for this step.

Throws

`Fault`

destroy()

Declaration

```
public void destroy()
     throws java.lang.IllegalStateException
```

Description
Use the `destroy` method to ensure a clean end to the step lifecycle.

Specified By
`destroy` in interface `IStep`.

setFaultCodes()

Declaration
```
void setFaultCodes(String[] faultCodes)
```

Description
Sets the fault codes thrown by this step.

Specified By
`init` in interface `IStep`.

Parameters
`faultCodes` (of type `String[]`) – Fault codes thrown by the step.

getFaultCodes()

Declaration
```
String[] getFaultCodes()
```

Description
Gets the fault codes thrown by this step.

Specified By
`init` in interface `IStep`.

Returns
A string array of the fault codes thrown by this step.

createResult()

Declaration
```
protected IResult createResult(int status)
```

Description
Creates the result for this step.

IContext

`Interface IContext`

Subinterfaces
`IMessageContext`

Declaration

```
public interface IContext
```

Description

Information about the pipeline execution context with regard to a component.

Table 3–4 Methods of IContext()

Method	Description
`getProperty()`	Searches for a specific property and returns the value of the property (`propName`).
`getPropertyNames()`	Returns an iterator on all the property names.
`containsProperty()`	Checks whether a specified property (`propName`) exists.
`setProperty()`	Sets the value of a specified property (`propName`).
`removeProperty()`	Removes a specified property (`propName`).

getProperty()

Declaration

```
public java.lang.Object getProperty(java.lang.String propName)
```

Description

Searches for a specific property and returns the value of that property.

Returns

The value for `propName`.

getPropertyNames()

Declaration

```
public java.util.Iterator getPropertyNames()
```

Description

Returns an iterator on all the property names.

containsProperty()

Declaration

```
public boolean containsProperty(java.lang.String propName)
```

Description

Checks for the presence of a specific property.

Returns

One of the boolean values – true or false.

setProperty()

Declaration

```
public void setProperty(java.lang.String propName,
java.lang.Object obj,
boolean persistent)
```

Parameters

`propName`, `obj`, `persistent`.

Description

Sets a property to a specified value.

removeProperty()

Declaration

```
public void removeProperty(java.lang.String propName)
```

Parameters

`propName`

Description

Removes a specific property.

IMessageContext

`Interface IMessageContext`

Declaration

```
public interface IMessageContext
extends IContext
```

Description

Provides contextual information about messages that the step handles.

All Superinterfaces

`IContext`

Inheritance from interface com.cfluent.policysteps.sdk.IContext

`containsProperty`, `getProperty`, `getPropertyNames`, `removeProperty`, `setProperty`, `setProperty`.

Table 3–5 IMessageContext() Static Fields

Field	Description
STAGE_PREREQUEST	Indicates the prerequest processing stage.
STAGE_REQUEST	Indicates the request processing stage.
STAGE_RESPONSE	Indicates the response processing stage.
STAGE_POSTRESPONSE	Indicates the postresponse processing stage.

Table 3–5 (Cont.) IMessageContext() Static Fields

Field	Description
STAGE_SERVICE	Indicates that the message is at the service stage.
STAGE_SERVICE_ DEFINITION	Provides a Web service definition.
STAGE_SERVICE_WSDL	Identifies the Web service WSDL.

Table 3–6 Methods of IMessageContext()

Method	Description
`getGUID()`	Gets the global unique ID.
`getServiceID()`	Gets the service ID.
`getService URL()`	Gets the service URL.
`getRemoteUser()`	Gets the remote user.
`getRequestMessage()`	Gets the SOAP request.
`getResponseMessage()`	Gets the SOAP response.
`setRequestMessage()`	Sets the request message.
`setResponseMessage()`	Sets the response message.
`getProcessingStage()`	Gets the processing stage information.
`setProcessingStage()`	Sets the processing stage.
`getInvocationStatus()`	Gets the invocation status.
`setUserLocale()`	Sets the user locale. The protocol handlers use this method to look for locale information in the message properties and message headers. If the locale information is found, the locale is set. If the locale information is not found, the user locale defaults to the product locale.
`getUserLocale()`	Gets the user locale. The policy step implementation code uses the user locale to provide the SOAP faults in the user's specified locale.

MessageContext() Fields

STAGE_PREREQUEST

Declaration

```
public static final java.lang.String STAGE_PREREQUEST
```

Description

Indicates the prerequest processing stage.

STAGE_REQUEST

Declaration

```
public static final java.lang.String STAGE_REQUEST
```

Description
Indicates the request processing stage.

STAGE_RESPONSE

Declaration
```
public static final java.lang.String STAGE_RESPONSE
```

Description
Indicates the response processing stage.

STAGE_POSTRESPONSE

Declaration
```
public static final java.lang.String STAGE_POSTRESPONSE
```

Description
Indicates the postresponse processing stage.

STAGE_SERVICE

Declaration
```
public static final java.lang.String STAGE_SERVICE
```

Description
Indicates that the message is at the service.

STAGE_SERVICE_DEFINITION

Declaration
```
public static final java.lang.String STAGE_SERVICE_DEFINITION
```

Description
Provides a service definition.

STAGE_SERVICE_WSDL

Declaration
```
public static final java.lang.String STAGE_SERVICE_WSDL
```

Description
Identifies the service WSDL.

getGUID()

Declaration
```
public java.lang.String getGUID()
```

Description
Gets the global unique ID.

Returns
The global unique ID.

getServiceID()

Declaration
```
public java.lang.String getServiceID()
```

Description
Gets the service ID.

Returns
The ID of the service based on the service registration information in Oracle WSM.

getServiceURL()

Declaration
```
public java.lang.String getRemoteUser()
```

Description
Gets the service URL.

Returns
The service URL for this pipeline.

getRemoteUser()

Declaration
```
public java.lang.String getRemoteUser()
```

Description
Gets the ID of the service requestor.

Returns
Gets the remote user.

getRequestMessage()

Declaration
```
public com.cfluent.ccore.message.SOAPMessage getRequestMessage()
```

Description
Gets the SOAP request.

Returns
The SOAP request message.

getResponseMessage()

Declaration
```
public com.cfluent.ccore.message.SOAPMessage getResponseMessage()
```

Description
Gets the SOAP response.

Returns
The SOAP response message.

setRequestMessage()

Declaration
```
public void setRequestMessage(com.cfluent.ccore.message.SOAPMessage
requestMessage)
```

Description
Sets the request message.

setResponseMessage()

Declaration
```
public void setResponseMessage(com.cfluent.ccore.message.SOAPMessage
responseMessage)
```

Description
Sets the response message.

getProcessingStage()

Declaration
```
public java.lang.String getProcessingStage()
```

Description
Gets the processing stage information.

Returns
The information from the processing stage.

setProcessingStage()

Declaration
```
public void setProcessingStage(java.lang.String stage)
```

Description
Sets the processing stage.

getInvocationStatus()

Declaration
```
public InvocationStatus getInvocationStatus()
```

Description
Gets the invocation status.

Returns
The invocation status.

IResult

Declaration
```
public class IResult()
```

extends
`java.lang.Object`

Methods Inherited from class java.lang.Object
`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`

Constructor
```
public Result()
```

Table 3–7 Methods of IResult

Method	Description
`getStatus()`	Gets the execution status.
`setStatus()`	Set the execution status.
`getFault()`	Returns the fault that caused the request execution failure.
`setFault()`	Sets the fault that caused this request execution to fail. The framework sets this fault if all the applicable fault handlers have failed. Step implementations should only call the `generateFault` method.

getStatus()

Declaration
```
public int getStatus()
```

Description
Gets the execution status.

Specified by
`getStatus` in interface `IResult`.

Returns
The execution status.

setStatus()

Declaration
```
public void setStatus(int status)
```

Description
Sets the execution status.

Specified by
`setStatus` in interface `IResult`.

Parameters
`status` – Sets the execution status.

getFault()

Declaration
```
public Fault getFault()
```

Description
The fault that caused the request execution failure. The framework sets the fault if all the applicable fault handlers have failed.

Specified by
`getFault` in interface `IResult`.

Returns
The fault that caused the request failure.

setFault()

Declaration
```
public void setFault(Fault fault)
```

Description
Sets the fault that caused the request execution failure. The framework sets the fault if all the applicable fault handlers have failed. Step implementations should only call the `generateFault` method.

Specified by
`setFault` in interface `IResult`

Parameters
`fault`

Result

Declaration

```
public class Result()
```

extends

`java.lang.Object`

implements

`IResult`

Methods Inherited from class java.lang.Object

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`

Constructor

```
public Result()
```

Fields Inherited from interface com.cfluent.policysteps.sdk.IResult

FAILED, SUCCEEDED, SUSPENDED

Table 3–8 Methods of Result

Method	Description
`getStatus()`	Gets the execution status.
`setStatus()`	Set the execution status.
`getFault()`	Returns the fault that caused the request execution failure.
`setFault()`	Sets the fault that caused this request execution to fail. The framework sets this fault if all the applicable fault handlers have failed. Step implementations should only call the generateFault method.
`toString()`	Result in string form

getStatus()

Declaration

```
public int getStatus()
```

Description

Gets the execution status.

Specified by

`getStatus` in interface `IResult`.

Returns

The execution status.

setStatus()

Declaration

```
public void setStatus(int status)
```

Description

Sets the execution status.

Specified by

`setStatus` in interface `IResult`.

Parameters

`status` – Sets the execution status.

getFault()

Declaration

```
public Fault getFault()
```

Description

The fault that caused the request execution failure. The framework sets the fault if all the applicable fault handlers have failed.

Specified by

`getFault` in interface `IResult`.

Returns

The fault that caused the request failure.

setFault()

Declaration

```
public void setFault(Fault fault)
```

Description

Sets the fault that caused the request execution failure. The framework sets the fault if all the applicable fault handlers have failed. Step implementations should only call the `generateFault` method.

Specified by

`setFault` in interface `IResult`

Parameters

`fault`

toString()

Declaration

```
public java.lang.String toString()
```

Overrides

`toString` in class `java.lang.Object`

Returns

The result in string format.

AgentContext

Declaration

```
public class AgentContext
```

extends

`java.lang.Object`

Properties

`agentID`, `resolver`, `properties`.

Description

Provides methods to retrieve and work with agent context information.

Constructor

public `AgentContext`(`java.lang.String agentId`,

`com.cfluent.common.resource.IResourceResolver resolver`,

`java.util.Map` properties)

throws `java.io.IOException`

Table 3–9 AgentContext Static Fields

Field	Description
SERVER_ID	Component ID.
LOG_LOGBUNDLES	Bundles for the log.
LOG_LOGENABLED	Specifies whether logging is enabled.
COREMAN_ENABLED	Specifies whether the Oracle WSM Monitor is enabled.
POLICYSERVER_ ENDPOINT	Specifies the Oracle WSM Policy Manager endpoint.
POLICYSERVER_ ENABLED	Specifies that the Oracle WSM Policy Manager is enabled.
POLICYPACKS_ FILENAME	Policy packs file name (internal property).

Table 3–10 Methods of AgentContext

Method	Description
`getAgentID()`	Gets the agent ID as a string.
`getProperty()`	Gets the property value for a specified propName.
`getStringProperty()`	Gets the value of the specified property. Returns null if the property does not exist.

Table 3–10 (Cont.) Methods of AgentContext

Method	Description
`getIntProperty()`	Gets a property as an integer (primitive).
`getResourceResolver()`	Gets the resolver to a resource. The resolver is used to get the absolute path to the resource.
`getAllProperties()`	Gets all the properties as a key-value pair.

AgentContext Fields

SERVER_ID

Declaration

```
public static java.lang.String SERVER_ID
```

Description

Value for the component ID.

LOG_LOGBUNDLES

Declaration

```
public static java.lang.String LOG_LOGBUNDLES
```

Description

Bundles for the log.

LOG_LOGENABLED

Declaration

```
public static final java.lang.String LOG_LOGENABLED
```

Description

Property if logging is enabled.

COREMAN_ENABLED

Declaration

```
public static final java.lang.String COREMAN_ENABLED
```

Description

Property that specifies that the Oracle WSM Monitor is enabled.

POLICYSERVER_ENDPOINT

Declaration

```
public static final java.lang.String POLICYSERVER_ENDPOINT
```

Description
Property for Oracle WSM Policy Manager endpoint.

POLICYSERVER_ENABLED

Declaration
```
public static final java.lang.String POLICYSERVER_ENABLED
```

Description
Specifies that Oracle WSM Policy Manager is enabled.

POLICYPACKS_FILENAME

Declaration
```
public static final java.lang.String POLICYPACKS_FILENAME
```

Description
Property for policy packs file name.

getAgentID()

Declaration
```
public java.lang.String getAgentID()
```

Description
Gets the agent ID. The agent ID is configured in the `config.xml` file and is loaded by the constructor as a property during initialization.

Returns
The agent ID as a string.

getProperty()

Declaration
```
public java.lang.Object getProperty(java.lang.String propName)
```

Description
Gets the property value for a specified `propName`.

Parameters
`propName` – Name of the property for which the value needs to be retrieved.

Returns
The value of specified property. It returns a null if the property does not exist.

getStringProperty()

Declaration
```
public java.lang.String getStringProperty(java.lang.String propName,
```

```
java.lang.String defaultValue)
```

Description

Gets the property value as a string.

Parameters

`propName` – Name of the property for which the value needs to be retrieved.

Returns

The string representation of the value of the property in argument. It returns a null if the property is not available.

getIntProperty()

Declaration

```
public int getIntProperty(java.lang.String propName)
```

Description

Gets a property as an integer (primitive).

Parameters

`propName` – Name of the property for which the value needs to be retrieved.

Returns

An integer (primitive) value of the property name in the argument.

getIntProperty()

Declaration

```
public int getIntProperty(java.lang.String propName,
java.lang.string defaultValue)
```

Description

Fetches the value of a property in the argument.

Parameters

`propName` – Name of the property for which the value needs to be retrieved.

`defaultValue` – Default value that is returned if the property is not available.

Returns

The value of the property in the argument. It returns the value if the property exists; returns the default value in the argument, if the property is not available.

getResourceResolver()

Declaration

```
IResourceResolver getResourceResolver()
```

Description
Gets the resolver to a resource. The resolver is used to get the absolute path to the resource.

Returns
The object of type IResourceResolver.

getAllProperties()

Declaration
```
public java.util.Map getAllProperties()
```

Description
Gets all the properties as a key-value pair.

Returns
Then object of type Map, containing key-value pairs of the properties.

InvocationStatus

Declaration
```
public class InvocationStatus
```

extends
`java.lang.Object`

Implements
`java.io.Serializable`

Inherited from class java.lang.Object
`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`

Description
This class sends step execution status to the Oracle WSM Monitor. The Oracle WSM Monitor uses this information to interpret measurements.

Constructor
```
public InvocationStatus()
```

Table 3–11 Fields of InvocationStatus

Field	Description
SUCCEEDED	Invocation Succeeded flag.
FAILED	Invocation failure flag.
FAILEDOVER	Failed Over flag.
PENDING	Measurement or execution pending.
NA	Object not available.

Table 3–12 Methods of InvocationStatus

Method	Description
`getServiceName()`	Gets the name of the service being invoked.
`setServiceName()`	Sets the service name.
`getTime()`	Gets the time.
`setTime()`	Sets the time.
`getAuthenticationStatus()`	Gets the authentication status.
`setAuthenticationStatus()`	Sets the authentication status.
`getAuthorizationStatus()`	Gets the authorization status.
`setAuthorizationStatus()`	Sets the authorization status.
`getServiceStatus()`	Gets the service status.
`setServiceStatus()`	Sets the service status.
`getInvocationStatus()`	Gets the invocation status.
`setInvocationStatus()`	Sets the invocation status.
`getSize()`	Gets the size.
`setSize()`	Sets the size.
`getLatency()`	Gets the default value for overall latency for policy pipelines.
`setLatency()`	Sets the value for overall latency.
`getServiceLatency()`	Gets the value for service latency in milliseconds.
`setServiceLatency()`	Sets the value for service latency in milliseconds.
`getFlowID()`	Gets the flow ID.
`setFlowID()`	Sets the flow ID.
`getMethod()`	Gets the method.
`setMethod()`	Sets the method.
`getCorrelationContext()`	Gets the correlation.
`setCorrelationContext()`	Sets the correlation.
getMessageId()	Gets the message ID.
`setMessageId()`	Sets the message ID.
`getErrorMessage()`	Gets the error message for the failure.
`setErrorMessage()`	Sets the error message on failure.
`toString()`	String representation of the object.

InvocationStatus() Fields

SUCCEEDED

Declaration

```
public static final int SUCCEEDED
```

Description
Invocation succeeded flag.

FAILED

Declaration
```
public static final int FAILED
```

Description
Invocation failure flag.

FAILEDOVER

Declaration
```
public static final int FAILEDOVER
```

Description
Failed Over flag.

PENDING

Declaration
```
public static final int PENDING
```

Description
Measurement or execution pending.

NA

Declaration
```
public static final int NA
```

Description
Object not available.

getServiceName()

Declaration
```
public java.lang.String getServiceName()
```

Description
Gets the name of the service being invoked.

Returns
The name of the service being invoked.

setServiceName()

Declaration

```
public void setServiceName(java.lang.String name)
```

Description

Sets the service name.

getTime()

Declaration

```
public long getTime()
```

Description

Gets the time.

setTime()

Declaration

```
public void setTime(long time)
```

Description

Sets the time.

Parameters

`time`

getAuthenticationStatus()

Declaration

```
public int getAuthenticationStatus()
```

Description

Gets the authentication status.

setAuthenticationStatus()

Declaration

```
public void setAuthenticationStatus(int s)
```

Description

Sets the authentication status.

Parameters

`s` – Value for authorization status.

getAuthorizationStatus()

Declaration

```
public int getAuthorizationStatus()
```

Description

Gets the authorization status.

Returns

Value for authorization status.

setAuthorizationStatus()

Declaration

```
public void setAuthorizationStatus(int s)
```

Description

Sets the authorization status.

Parameters

`s` – Value for authorization status.

getServiceStatus()

Declaration

```
public int getServiceStatus()
```

Description

Gets the service status.

Returns

The value of service status.

setServiceStatus()

Declaration

```
public void setServiceStatus(int s)
```

Description

Sets the service status.

Parameters

`s` – Value for service status.

getInvocationStatus()

Declaration

```
public int getInvocationStatus()
```

Description
Gets the invocation status.

Returns
The value for invocation status.

setInvocationStatus()

Declaration
```
public void setInvocationStatus(int s)
```

Description
Sets the invocation status.

Parameters
`s` – Value for invocation status.

getSize()

Declaration
```
public int getSize()
```

Description
Gets the size.

Returns
The value of size.

setSize()

Declaration
```
public void setSize(int s)
```

Description
Sets the size.

Parameters
`s` – The value of size.

getLatency()

Declaration
```
public int getLatency()
```

Description
Gets the default value for overall latency used for policy pipelines.

Returns
The value for overall latency.

setLatency()

Declaration
```
public void setLatency(int l)
```

Description
Sets the value for overall latency.

Parameters
`l` – The value for overall latency.

getServiceLatency()

Declaration
```
public int getServiceLatency()
```

Description
Gets the service only latency.

Returns
The value for service latency (in milliseconds).

See Also
`getLatency()`

setServiceLatency()

Declaration
```
public void setServiceLatency(int l)
```

Description
Sets the value for service latency (in milliseconds).

getFlowID()

Declaration
```
public java.lang.String getFlowID()
```

Description
Gets the flow ID.

Returns
The value for flow ID.

setFlowID()

Declaration

```
public void setFlowID(java.lang.String flowid)
```

Description

Sets the flow ID.

Parameters

`flowid` – A string representing the flow ID.

getMethod()

Declaration

```
public java.lang.String getMethod()
```

Description

Finds and returns the method on the service.

Returns

The method on the service.

setMethod()

Declaration

```
public void setMethod(java.lang.String inMethod)
```

Description

Sets the method.

Parameters

`inMethod` – The WSDL operation.

getCorrelationContext()

Declaration

```
public java.lang.String getCorrelationContext()
```

Description

Gets the correlation.

setCorrelationContext()

Declaration

```
public void setCorrelationContext(java.lang.String inCorrelationContext)
```

Description

Sets the correlation.

Parameters
`inCorrelationContext` – The correlation context string.

getMessageId()

Declaration
```
public java.lang.String getMessageId()
```

Description
Gets the message ID.

setMessageId()

Declaration
```
public void setMessageId(java.lang.String inMessageId)
```

Description
Sets the message ID.

Parameters
`inMessageId` – The message ID string.

getErrorMessage()

Declaration
```
public java.lang.String getErrorMessage()
```

Description
Gets the associated error message for the failure.

setErrorMessage()

Declaration
```
public void setErrorMessage(java.lang.String m)
```

Description
Sets the error message on failure.

Parameters
`m` – The error message string.

toString()

Declaration
```
public java.lang.String toString()
```

Description
String representation of this object.

Overrides

`toString` in class `java.lang.Object`

Fault

Declaration

```
public class Fault extends BaseException
```

Description

The `Fault` class contains all the information about a step execution failure.

Table 3–13 Methods of Fault

Method	Description
`getFaultNS()`	Gets the fault namespace.
`getFaultString()`	Gets the fault string.
`getFaultDetail()`	Gets the fault detail.
`getFaultActor()`	Gets the fault actor.
`getQualifiedFaultCode()`	Gets the qualified fault code, for example, "Client.AuthenticationFailure."
`getFaultCodeQualifier()`	Gets the fault code qualifier.
`getFaultCode()`	Gets the fault code.
`getFaultDetailsAsString()`	Gets the fault detail as a string.

getFaultNS()

Declaration

```
public String getFaultNS()
```

Description

Gets the fault namespace.

Returns

The namespace as set or the default namespace.

getFaultString()

Declaration

```
public String getFaultString()
```

Description

Gets the fault string.

Returns

The fault string as set for the fault.

getFaultDetail()

Declaration

```
public Detail getFaultDetail()
```

Description

Gets the fault detail.

Returns

A detailed message for the cause of the fault.

getFaultActor()

Declaration

```
public String getFaultActor()
```

Description

Gets the fault actor.

Returns

The actor that experienced the fault.

getQualifiedFaultCode()

Declaration

```
public String getQualifiedFaultCode()
```

Description

Gets the qualified fault code, for example, "Client.AuthenticationFailure."

getFaultCodeQualifier()

Declaration

```
public String getFaultCodeQualifier()
```

Description

Gets the fault code qualifier.

getFaultCode()

Declaration

```
public String getFaultCode()
```

Description

Gets the fault code.

Returns

The fault code as set for this fault.

getFaultDetailsAsString()

Declaration

```
public String getFaultDetailsAsString()
```

Description

Gets the fault detail as a string.

Returns

A string representation of the fault detail.

A

Custom Step Source Code and Step Template

This appendix lists the source code (Example A–1) and step template (Example A–2) for the custom authentication step sample described in Chapter 2, "Understanding the Sample Custom Step".

Custom Step Source Code

Example A–1 is the source code for the custom authentication step sample. The file containing this code, `CustomAuthenticationStep.java`, can be found in the *ORACLE_HOME*`\owsm\samples\customsteps\customAuthenticationStep` directory.

Example A–1 Custom Authentication Step Source Code

```
package customsteps;

import com.cfluent.policysteps.sdk.*;
import javax.servlet.http.HttpServletRequest;
import javax.xml.soap.*;
import java.io.*;
import java.util.Locale;
import java.util.Iterator;

public class CustomAuthenticationStep extends AbstractStep {
  private String expectedUsername = null;
  private String expectedUserPassword = null;
  private PrintWriter out = null;
  public CustomAuthenticationStep() {
  }

  public void init() throws IllegalStateException {
    try {
      out = new PrintWriter(new BufferedWriter(new
FileWriter("log/CustomAuthenticationStep.log", true)));
    } catch (Exception e) {
      String errMsg = "Error in creating log file for custom authentication step:"
+ e.getMessage();
      System.err.println(errMsg);
      e.printStackTrace();
    throw new IllegalStateException(errMsg);
    }
  }
```

```
  public void destroy() {
    out.close();
  }

  public IResult execute (IMessageContext messageContext) throws Fault {
    log("********** Entering Custom Authentication execute method **********");
    Result result = new Result();
    result.setStatus(IResult.FAILED);//initialize result

    String processingStage = messageContext.getProcessingStage();
    log("Processing stage is " + processingStage);
    boolean isRequest =
      (IMessageContext.STAGE_REQUEST.equals(messageContext.getProcessingStage())
|| IMessageContext.STAGE_PREREQUEST.equals(messageContext.getProcessingStage()));

    if(!isRequest) {
      // This step is applicable only for request pipelines
      result.setStatus(IResult.SUCCEEDED);
      return result;
    }

    // get SOAP message to access various parts of it
    javax.xml.soap.SOAPMessage soapMessage = messageContext.getRequestMessage();
    // Log the request SOAP message
    logSOAPMessage(soapMessage);

    //get user locale
    Locale userLocale = messageContext.getUserLocale();
    log("User locale is " +userLocale.getDisplayName());

    // get HTTP Header containing client IP address
    HttpServletRequest httpServletRequest =
(HttpServletRequest)messageContext.getProperty("javax.servlet.request");
    String remoteAddr = httpServletRequest.getHeader("Host");
    log("Client ip address is " + remoteAddr);

    String verifiedUser = null;
    try {
      verifiedUser = authenticate(soapMessage);
      // set result status
      result.setStatus(IResult.SUCCEEDED);
    } catch (Exception ex) {
      String errMsg = ex.getMessage();
      log(errMsg);
      // set monitoring data for failed authentication
      messageContext.getInvocationStatus();setAuthenticationStatus
(InvocationStatus.FAILED);
      messageContext.getInvocationStatus().setErrorMessage(errMsg);
      // if you use this method, the Fault will be populated with appropriate
fault code
      // (AuthenticateFault for this custom step)
      generateFault(errMsg);
    }

    log("Verified user is " + verifiedUser);

    // set monitoring data for successful authentication
    messageContext.getInvocationStatus().setAuthenticationStatus
(InvocationStatus.SUCCEEDED);
```

```
    return result;
  }

  private String authenticate(SOAPMessage soapMessage) throws Exception {
    // Get username and password from Security header
    String username = null;
    String password = null;
    String namespaceURI ="http://docs.oasis-open.org/wss/2004/01/
oasis-200401-wss-wssecurity-secext-1.0.xsd";

    SOAPHeader soapHeader = soapMessage.getSOAPPart().getEnvelope().getHeader();
    if(soapHeader != null) {
      SOAPElement securityHeader = getToken(soapHeader, "Security", namespaceURI);
      if(securityHeader != null) {
        SOAPElement username TokenElement = getToken(securityHeader,
"UsernameToken", namespaceURI);
        if(usernameTokenElement != null) {
          SOAPElement usernameElement = getToken(username TokenElement,
"Username", namespaceURI);
          SOAPElement passwordElement = getToken(username TokenElement,
"Password", namespaceURI);
          if(usernameElement != null) username = usernameElement.getValue();
          if(usernameElement != null) password = passwordElement.getValue();
        }
      }
    }

    if(username == null) {
      String msg = "Authentication failed: Username not supplied";
      throw new Exception (msg);
    }

    // authenticate against the configured userid and password.
    // This can be against any custom datastore as well.
    if(!expectedUsername.equals(username) ||
!expectedUserPassword.equals(password)) {
      String msg = "Authentication failed for user " + username;
      throw new Exception(msg);
    }

    return username;
  }

  private SOAPElement getToken(SOAPElement element, String tokenName, String
nameSpaceURI)
  throws Exception
  {
      SOAPElement token = null;
      SOAPFactory factory = SOAPFactory.newInstance();
    Name localName = factory.createName(tokenName, "", nameSpaceURI);
    Iterator iter = element.getChildElements(localName);

    if(iter.hasNext()) token = (SOAPElement) iter.next();
    return token;
  }

  private void logSOAPMessage(javax.xml.soap.SOAPMessage soapMsg) {
    String msg = null;
    try {
      ByteArrayOutputStream baos = new ByteArrayOutputStream();
```

```
      soapMsg.writeTo(baos);
      msg = baos.toString();
      log("Request SOAP message is " + msg);
    } catch (Exception ex) {
      System.err.println("Exception encountered while converting SOAP message to a
String");
      ex.printStackTrace();
    }
  }

  private void log(String str) {
    try {
      out.println(str);
      out.flush();
    } catch (Exception ex) {
      System.err.println("Exception encountered while writing to file");
      ex.printStackTrace();
    }
  }

  public String getUsername() {
    return expectedUsername;
  }

  public void setUsername(String username) {
    this.expectedUsername = username;
  }

  public String getPassword() {
    return expected UserPassword;
  }

  public void setPassword(String password) {
    this.expectedUserPassword = password;
  }
}
```

Custom Step Template

Example A–2 lists the custom step template for the custom authentication step sample:

Example A–2 Step Template (CustomAuthenticationStep.xml)

```
<csw:StepTemplate xmlns:csw="http://schemas.confluentsw.com/ws/2004/07/policy"
name="Custom Authenticate step" package="customsteps" timestamp="Oct 31, 2005
05:00:00 PM" version="1" id="118970829">
  <csw:Description>Custom step that authenticates the user against the credentials
entered here.</csw:Description>
  <csw:Implementation>customsteps.CustomAuthenticationStep</csw:Implementation>
  <csw:Faults>
    <csw:Fault xmlns:fns="http://schemas.oblix.com/ws/2003/08/Faults">
fns:AuthenticationFault</csw:Fault>
</csw:Faults>
  <csw:PropertyDefinitions>
    <csw:PropertyDefinitionSet name="Basic Properties">
      <csw:PropertyDefinition name="Enabled" type="boolean">
        <csw:Description>If set to true, this step is enabled</csw:Description>
        <csw:DefaultValue>
          <csw:Absolute>true</csw:Absolute>
        </csw:DefaultValue>
```

```
      </csw:PropertyDefinition>
    </csw:PropertyDefinitionSet>
    <csw:PropertyDefinitionSet name="User Credentials">
      <csw:PropertyDefinition name="Username" type="string" isRequired="true">
        <csw:DisplayName>Username</csw:DisplayName>
        <csw:Description>Username used for authentication</csw:Description>
        <csw:DefaultValue>
          <csw:Absolute>test</csw:Absolute>
        </csw:DefaultValue>
      </csw:PropertyDefinition>
      <csw:PropertyDefinition name="Password" type="string" isRequired="true"
displayType="password">
        <csw:DisplayName>Password</csw:DisplayName>
        <csw:Description>Password used for authentication</csw:Description>
        <csw:DefaultValue>
          <csw:Absolute>test</csw:Absolute>
        </csw:DefaultValue>
      </csw:PropertyDefinition>
    </csw:PropertyDefinitionSet>
  </csw:PropertyDefinitions>
</csw:StepTemplate>
```

B

Step Template Schema

This appendix contains an Oracle Web Services Manager (Oracle WSM) step template schema.

Step Template Schema

The following is a step template schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://schemas.confluentsw.com/ws/2004/07/policy"
xmlns:csw="http://schemas.confluentsw.com/ws/2004/07/policy"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
   <xsd:complexType name="Restriction">
      <xsd:sequence>
         <xsd:element name="length" type="csw:Facet" minOccurs="0"
maxOccurs="unbounded"/>
         <xsd:element name="minLength" type="csw:Facet" minOccurs="0"
maxOccurs="unbounded"/>
         <xsd:element name="pattern" type="csw:Facet" minOccurs="0"
maxOccurs="unbounded"/>
         <xsd:element name="enumeration" type="csw:Facet" minOccurs="0"
maxOccurs="unbounded"/>
         <xsd:element name="totalDigits" type="csw:Facet" minOccurs="0"
maxOccurs="unbounded"/>
         <xsd:element name="fractionDigits" type="csw:Facet" minOccurs="0"
maxOccurs="unbounded"/>
         <xsd:element name="minInclusive" type="csw:Facet" minOccurs="0"
maxOccurs="unbounded"/>
         <xsd:element name="maxInclusive" type="csw:Facet" minOccurs="0"
maxOccurs="unbounded"/>
         <xsd:element name="minExclusive" type="csw:Facet" minOccurs="0"
maxOccurs="unbounded"/>
         <xsd:element name="maxExclusive" type="csw:Facet" minOccurs="0"
maxOccurs="unbounded"/>
         <xsd:element name="whiteSpace" type="csw:Facet" minOccurs="0"
maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="base" type="xsd:string" use="required"/>
   </xsd:complexType>

   <xsd:complexType name="Facet">
      <xsd:attribute name="value" type="xsd:string" use="required"/>
   </xsd:complexType>

   <xsd:complexType name="PropertyDefinitionType">
```

```
      <xsd:all>
         <xsd:element name="DisplayName" type="xsd:string" minOccurs="0"/>
         <xsd:element name="Description" type="xsd:string" minOccurs="0"/>
         <xsd:element name="DefaultValue">
            <xsd:complexType>
               <xsd:sequence>
                  <xsd:choice>
                     <xsd:element name="PropertyRef" type="xsd:string"/>
                     <xsd:element name="ResourceRef" type="xsd:string"/>
                     <xsd:element name="Absolute" type="xsd:string"/>
                  </xsd:choice>
               </xsd:sequence>
            </xsd:complexType>
         </xsd:element>
            <xsd:complexType>
               <xsd:all>
                  <xsd:element name="Restriction" type="csw:Restriction"
minOccurs="0"/>
               </xsd:all>
               <xsd:attribute name="name" type="xsd:string" use="required"/>
            </xsd:complexType>
         </xsd:element>
      </xsd:all>
      <xsd:attribute name="name" type="xsd:string" use="required"/>
      <xsd:attribute name="type" type="xsd:string" use="required"/>
      <xsd:attribute name="displayType" type="xsd:string" use="optional"/>
      <xsd:attribute name="isMultivalued" type="xsd:string" use="optional"/>
      <xsd:attribute name="isRequired" type="xsd:string" use="optional"/>
   </xsd:complexType>

   <xsd:element name="StepTemplate">
      <xsd:complexType>
         <xsd:all>
            <xsd:element name="Description" type="xsd:string" minOccurs="0"/>
            <xsd:element name="Implementation" type="xsd:string"/>
            <xsd:element name="PropertyDefinitions">
               <xsd:complexType>
                <xsd:sequence>
                   <xsd:element name="PropertyDefinitionSet" minOccurs="0"
maxOccurs="unbounded">
                     <xsd:complexType>
                       <xsd:sequence>
                        <xsd:element name="PropertyDefinition"
                          type="csw:PropertyDefinitionType" minOccurs="0"
maxOccurs="unbounded"/>
                          </xsd:sequence>
                          <xsd:attribute name="name" type="xsd:string"
use="optional"/>
                       </xsd:complexType>
                     </xsd:element>
                  </xsd:sequence>
               </xsd:complexType>
            </xsd:element>
            <xsd:element name="Faults" minOccurs="0">
               <xsd:complexType>
                <xsd:sequence>
                </xsd:sequence>
               </xsd:complexType>
            </xsd:element>
         </xsd:all>
```

```
         <xsd:attribute name="id" type="xsd:string" use="required"/>
         <xsd:attribute name="name" type="xsd:string" use="required"/>
         <xsd:attribute name="package" type="xsd:string" use="required"/>
         <xsd:attribute name="timestamp" type="xsd:string" use="required"/>
         <xsd:attribute name="version" type="xsd:nonNegativeInteger"
use="optional"/>
      </xsd:complexType>
   </xsd:element>
</xsd:schema>
```

Index

J

L

O

P

R

S

T