

HP OpenView Select Identity

External Call Developer Guide

Software Version: 3.0.1



October 2004

© 2004 Hewlett-Packard Development Company, L.P.

Legal Notices

Warranty

Hewlett-Packard makes no warranty of any kind with regard to this document, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

A copy of the specific warranty terms applicable to your Hewlett-Packard product can be obtained from your local Sales and Service Office.

Restricted Rights Legend

Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause in DFARS 252.227-7013.

Hewlett-Packard Company
United States of America

Rights for non-DOD U.S. Government Departments and Agencies are as set forth in FAR 52.227-19(c)(1,2).

Copyright Notices

© 2004 Hewlett-Packard Development Company, L.P.

No part of this document may be copied, reproduced, or translated into another language without the prior written consent of Hewlett-Packard Company. The information contained in this material is subject to change without notice.

This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>). Portions Copyright (c) 1999-2003 The Apache Software Foundation. All rights reserved.

Select Identity uses software from the Apache Jakarta Project including:

- Commons-beanutils.
- Commons-collections.
- Commons-logging.
- Commons-digester.
- Commons-httpclient.

- Element Construction Set (ecs).
- Jakarta-poi.
- Jakarta-regexp.
- Logging Services (log4j).

Additional third party software used by Select Identity includes:

- JasperReports developed by SourceForge.
- iText (for JasperReports) developed by SourceForge.
- BeanShell.
- Xalan from the Apache XML Project.
- Xerces from the Apache XML Project.
- Java API for XML Processing from the Apache XML Project.
- SOAP developed by the Apache Software Foundation.
- JavaMail from SUN Reference Implementation.
- Java Secure Socket Extension (JSSE) from SUN Reference Implementation.
- Java Cryptography Extension (JCE) from SUN Reference Implementation.
- JavaBeans Activation Framework (JAF) from SUN Reference Implementation.
- OpenSPML Toolkit from OpenSPML.org.
- JGraph developed by JGraph.
- Hibernate from Hibernate.org.

This product includes software developed by Teodor Danciu (<http://jasperreports.sourceforge.net>). Portions Copyright (C) 2001-2004 Teodor Danciu (teodord@users.sourceforge.net). All rights reserved.

Portions Copyright 1994-2004 Sun Microsystems, Inc. All Rights Reserved.

This product includes software developed by the Waveset Technologies, Inc. (www.waveset.com). Portions Copyright © 2003 Waveset Technologies, Inc. 6034 West Courtyard Drive, Suite 210, Austin, Texas 78730. All rights reserved.

Portions Copyright (c) 2001-2004, Gaudenz Alder. All rights reserved.

Trademark Notices

HP OpenView Select Identity is a trademark of Hewlett-Packard Development Company, L.P. Microsoft, Windows, the Windows logo, and SQL Server are trademarks or registered trademarks of Microsoft Corporation.

Sun™ workstation, Solaris Operating Environment™ software, SPARCstation™ 20 system, Java technology, and Sun RPC are registered trademarks or trademarks of Sun Microsystems, Inc. JavaScript is a trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

This product includes the Sun Java Runtime. This product includes code licensed from RSA Security, Inc. Some portions licensed from IBM are available at <http://oss.software.ibm.com/icu4j/>.

IBM, DB2 Universal Database, DB2, WebSphere, and the IBM logo are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

This product includes software provided by the World Wide Web Consortium. This software includes xml-apis. Copyright © 1994-2000 World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University). All Rights Reserved. <http://www.w3.org/Consortium/Legal/>

Intel and Pentium are trademarks or registered trademarks of Intel Corporation in the United States, other countries, or both.

AMD and the AMD logo are trademarks of Advanced Micro Devices, Inc.

BEA and WebLogic are registered trademarks of BEA Systems, Inc.

VeriSign is a registered trademark of VeriSign, Inc. Copyright © 2001 VeriSign, Inc. All rights reserved.

All other product names are the property of their respective trademark or service mark holders and are hereby acknowledged.

Support

Please visit the HP OpenView web site at:

<http://openview.hp.com/>

There you will find contact information and details about the products, services, and support that HP OpenView offers.

You can go directly to the support web site at:

<http://support.openview.hp.com/>

The support web site includes:

- Downloadable documentation
- Troubleshooting information
- Patches and updates
- Problem reporting
- Training information
- Support program information

contents

Chapter 1	Introduction to External Calls	7
	Overview of the APIs	8
	Product Documentation	10
Chapter 2	Creating an External Call	12
	Coding Value External Calls	12
	Coding Approver Selection External Calls	14
	Coding Workflow External Calls	17
	Compiling and Deploying an External Call	20
Chapter 3	Examples	21
	Value Generation External Call	21
	Value Constraint External Call	24
	Value Validation External Call	27
	Approver Selection External Call	28
	Workflow External Call	29
	Adding a Generated Value to a User	30
	Retrieving and Changing Attributes and Entitlements	32

Introduction to External Calls

HP OpenView Select Identity supports the ability to invoke calls to external systems. Specifically, you can use external calls to perform the following:

- Value generation — generates the values of an attribute
- Value constraint — provides a list of possible values for an attribute
- Value validation — validates the value of an attribute
- Approver selection — queries an external system for a list of users who can approve provisioning requests during a workflow
- Workflow action — performs a task as part of a workflow, enabling you to integrate approval processes with external processes and systems

You must code the classes that are called by external calls using the External Call API and Workflow API, which define Java-based interfaces for creating external callouts. The Select Identity-facing portion of the interface must be written in Java.

This chapter provides an overview of the External Call and Workflow APIs and general guidelines for creating each type of external call.

Overview of the APIs

Creating an external call entails coding one or more Java classes. Thus, you must have an understanding of the Java Developer Kit (JDK), version 1.4. For information about the J2EE APIs, refer to <http://java.sun.com/j2se/1.4.2/docs/api/index.html>.

JavaDoc is provided for the External Call and Workflow APIs on the `docs/api_help/external_calls/Javadoc` directory on the Select Identity CD for details. Refer to this web-based help for implementation and usage details.

The following classes and interfaces are provided by the External Call API and are available for creating value external calls.

- **TValueConstraintIntf**

Defines classes that return a list of value constraints. External calls that generate possible attribute values must implement this interface.

- **TValueConstraintIntf.TValueConstraintBeanIntf**

Contains the constrained values. External calls the constrain attribute values should import this interface.

- **TValueGenerationIntf**

Defines a class that generates the value of an attribute. External calls that generate values must implement this interface.

- **TValueValidationIntf**

Defines a class that validates the value of an attribute. External calls that validate values must implement this interface.

- **TAttributeDefinitionException** and **TAttributeValueValidationException**

Exceptions defined for use by external calls.

The following classes and interfaces are provided by the Workflow API for use in approver selection and workflow external calls.

- **IWFClient**

Invokes a workflow template, thereby creating a workflow instance. This interface also enables you to resume a passivated workflow instance and to terminate an instance.

- **IWfQuery**
Retrieves runtime and configuration information about a workflow at the template, instance, block, or activity level.
- **IWfDataUpdate**
Updates workflow variables.
- **StatelessServiceObjectFactory**
Creates a component that implements the IWfClient, IWfQuery, and IWfDataUpdate interfaces and forces the interfaces to behave like stateless EJB while hiding EJB-specific code and deployment information.
- **WfExternalCall**
Provides the contract between Select Identity and an external stage in a workflow. Select Identity can invoke classes implementing this interface to perform actions in a workflow. External calls that are invoked by workflow instances must implement this interface. This class returns a collection of approvers.
- **WfExternalCallException**
Exception defined for use by the Workflow API.
- **WfExternalCallStatus**
Returns the status of an approval stage to Select Identity. Along with the status of the stage, it can also return changes to the user profile including attributes and entitlements.
- **WfSelectApproverIntf**
Provides the contract between Select Identity and an external entity to select an approver. Objects implementing this interface can be registered with Select Identity and used in approval stages to dynamically select an approver.

The following classes may be used to further explore the Select Identity's request and approval frameworks:

- **AttributeRecord**
Represents an attribute
- **ChangeRecord**
Represents a change in a attribute. Objects of this class can be used to communicate changes in a user profile from external calls.

- **Request**
Defines methods that retrieve and set all information related to an incoming request in the Select Identity system.
- **RequestJobItem**
Defines the job that handles the request in the Select Identity system.
- **RequestTarget**
Defines the target of a request.
- **RequestTargetParam**
Defines the parameters of a request target.
- **RequestTargetParamValue**
Enables you to set and get parameter values.
- **TAFilter**
A general class that stores filter criteria for a selection (search) procedure.
- **TAResourceAction**
Represents the action to take place on the target resource.
- **TAResourceEvent**
Represent the event for the request.
- **TAResourceType**
Represents the type of request.

Product Documentation

The Select Identity product documentation includes the following:

- Release notes are provided in the top-level directory of the HP OpenView Select Identity CD. This document provides important information about new features, known defects and limitations, and special usage information that you should be familiar with before using the product.
- For installation and configuration information, refer to the *HP OpenView Select Identity Installation Guide*. All installation prerequisites, system

requirements, and procedures are explained in detail in this guide. Specific product configuration and logging settings are included. This guide also includes uninstall and troubleshooting information.

- An *HP OpenView Connector Installation Guide* is provided for each resource connector. These are located on the Select Identity Connector CD.
- Detailed procedures for deployment and system management are documented in the *HP OpenView Select Identity Administrator Guide* and Select Identity online help system. This guide provides detailed concepts and procedures for deploying and configuring the Select Identity system. In the online help system, tasks are grouped by the administrative functions that govern them.
- The *HP OpenView Select Identity Workflow Studio Guide* provides detailed information about using Workflow Studio to create workflow templates. It also describes how to create reports that enable managers and approvers to check the status of account activities.
- The *HP OpenView Select Identity External Call Developer Guide* provides detailed information about creating calls to third-party applications. These calls can then be deployed in Select Identity to constrain attribute values or facilitate workflow processes. In addition, JavaDoc is provided for this API. To view this help, extract the `javadoc.jar` file in the `docs/api_help/external_calls/Javadoc` directory on the HP OpenView Select Identity CD.
- If you need to develop connectors, which enable you to connect to external systems for provisioning, refer to the *HP OpenView Select Identity Connector Developer Guide*. This document provides an overview of the Connector API and the steps required to build a connector. The audience of this guide is developers familiar with Java.

JavaDoc is also provided for the Connector API. To view this help, extract the `javadoc.jar` file in the `docs/api_help/connectors/Javadoc` directory on the HP OpenView Select Identity CD.

- The *HP OpenView Select Identity Web Service Developer Guide* describes the Web Service, which enables you to programmatically provision users in Select Identity. This guide provides an overview of the operations you can perform through use of the Web Service, including SPML examples for each operation.

An independent, web-based help system is available for this API. To view this help, double-click the `index.htm` file in the `docs/api_help/web_service/help` directory on the HP OpenView Select Identity CD.

Creating an External Call

The sections provided here describe how to implement an external call of each type. Refer to the JavaDoc provided in the `docs/api_help/external_calls/Javadoc` directory on the Select Identity CD for details about specific APIs.

Coding Value External Calls

To create an external call, follow these guidelines:

- Value Generation external calls must implement the `TAValueGenerationIntf` interface. See [Value Generation External Call on page 21](#) for code examples.
- Value Constraint external calls must implement the `TAValueConstraintIntf` interface. See [Value Constraint External Call on page 24](#) for code examples.
- Value Validation external calls must implement the `TAValueValidationIntf` interface. See [Value Validation External Call on page 27](#) for code examples.

In addition, the following provides general information about how to obtain to input parameters, Select Identity attributes, and what needs to be returned:

- To retrieve an external call parameter that is defined on the External Calls page of the Select Identity client, use the **containsKey** and **get** methods. The **containsKey** method verifies the existence of the specified parameter, and the **get** method retrieves the parameter. Here is an example:

```
if (attrs.containsKey("parametername"))
    String parameter = (String)
    attrs.get("parametername")
```

- To retrieve a Select Identity attribute, use one of the methods provided by the RequestTarget class, such as **getSingleRequestParamStr**. Here is an example that retrieves a single-value attribute:

```
String attributeValue=(String)requestTarget.
    getSingleRequestParamStr("attributename");
```

To retrieve the values of a multi-value attribute, use the following method:

```
Set attributeValues=(Set)requestTarget.
    getRequestParam("attributename").getRequestTargetParamValue()
```

- For Value Validation external calls, throw **TAAAttributeValueValidationException** if the call is unsuccessful.
- To log messages and errors, you can use any logging mechanism. Note, however, that the examples shown in this guide use an internal logging mechanism that is not available externally.

Coding Approver Selection External Calls

For approver selection external calls, the APIs support synchronous communication. The external system must complete its processing and provide status information as part of the call, which is required to return status that indicates how Select Identity will proceed with the workflow.

The following provides general information about how to implement an approver selection external call, how to obtain to input parameters, Select Identity attributes, and what needs to be returned:

- Approver selection external calls must implement the `WfSelectApproverIntf` interface.
- The main method of the external call must be called **getApprover**. Here is the call signature of the **getApprover** method:

```
public Collection getApprover(RequestTarget reqTarget,
                             HashMap attrs)
    throws WfExternalCallException;
```

- The workflow external call must return a collection of Select Identity user IDs. In addition, when an external call returns information, it must return data that is valid in Select Identity.
- To retrieve an external call parameter that is defined on the External Calls page of the Select Identity client, use **containsKey** and **get**. The **containsKey** method verifies the existence of the specified parameter, and the **get** method retrieves the parameter. Here is an example:

```
if (attrs.containsKey("parametername"))
    String parameter = (String)
    attrs.get("parametername")
```

- Select Identity defines the following standard parameters in the map:
 - `WfExternalCall.WF_PARAM_SERVICENAME` — for the service name. This is a String object.
 - `WfExternalCall.WF_PARAM_ADMINUSERID` — for the requestor's user name. This is a String object.
 - `WfExternalCall.WF_PARAM_REQUESTID` — for the request identifier. This is an Integer object.
 - `WfExternalCall.WF_PARAM_WORKFLOWINSTID` — for the workflow instance ID. This is an Integer object.

- To retrieve a Select Identity attribute, use one of the methods provided by the RequestTarget class, such as **getSingleRequestParamStr**. Here is an example that retrieves a single-value attribute:

```
String attributeValue=(String)requestTarget.  
    getSingleRequestParamStr("attributename");
```

To retrieve the values of a multi-value attribute, use the following method:

```
Set attributeValues=(Set)requestTarget.  
    getRequestParam("attributename").getRequestTargetParamValue()
```

- To retrieve workflow variables, use the IWfQuery interface as in the following example:

```
IWfQuery query =(IWfQuery)StatelessServiceObjectFactory.  
    create(IWfQuery.class);  
  
int instanceId = query.  
    getInstanceInfoByInstActivityId(instActivityId).getInstId();  
Map varMap = query.getCurrentVariableMap(instanceId);  
String comment = (String) varMap.get("$ApproverComments");
```

- To set and update a workflow variable, use the methods provided by the IWfDataUpdate interface, as follows:

```
IWfDataUpdate du =(IWfDataUpdate)StatelessServiceObjectFactory.  
    create(IWfDataUpdate.class);  
du.setWorkflowVar(instId, "workflowvariablename", "value");
```

- The following variables are available for use in your external call:

Variable Name	Description
activityId	The activity ID string for current activity in execution. This variable is maintained by the engine and cannot be updated in template. A string value is assigned to this variable.
instActivityId	An internal variable representing the instance activity ID for the current activity in execution. This variable is maintained by the engine and cannot be updated in template. It is assigned an integer value.

Variable Name	Description
_joinCommand	<p>Passed into the workflow by an external application that is invoked as an action in a workflow template activity. This variable is not persistent. Possible values include</p> <ul style="list-style-type: none"> • exit — unconditionally exits the block • exitAll — unconditionally exits the current block and all parent blocks • reset — resets the value of the joinCount property set in the block <p>For example, the application could pass this variable to the workflow to instruct it to exit a block unconditionally (even if, for instance, the _joinCount value is not met).</p>
_pushVar	<p>Passed into the workflow by an external application that is invoked as an action in a workflow template activity. The object is then added to workflow's internal pushList block variable. The pushed objects in the list can be displayed later in a tabular format in a report.</p> <p>You can specify any Java object as the value of this variable.</p>
\$_blockId	<p>The block ID for the current block. You can assign any string value to this variable.</p>
\$_instId	<p>The workflow instance ID. This variable is maintained by the engine and cannot be updated. An integer value is assigned to this variable.</p>

- To log messages and errors, you can use any logging mechanism. Note, however, that the examples shown in this guide use an internal logging mechanism that is not available externally.

Coding Workflow External Calls

For workflow external calls, the APIs support synchronous communication. Select Identity requires the external system to complete its processing and provide status information as part of the callout, which is required to return status that indicates how Select Identity will proceed with the workflow.

The following provides general information about how to implement an workflow external call, how to obtain to input parameters, Select Identity attributes, and what needs to be returned:

- Workflow external calls must implement the `WfExternalCall` interface.
- The main method of the external call must be called **process**. This method expects four input parameters. Here is the call signature of the **process** method:

```
WfExternalCallStatus process(String stageId,
    RequestTarget requestTarget,
    AttributeRecord [] availGrp,
    AttributeRecord [] availRole,
    AttributeRecord [] availEntilements,
    Map map) throws WfExternalCallException
```

- The workflow external call must return `WfExternalCallStatus`. In addition, when an external call returns information, it must return data that is valid in Select Identity.
- To retrieve an external call parameter that is defined on the External Calls page of the Select Identity client, use the **containsKey** and **get** methods. The **containsKey** method verifies the existence of the specified parameter, and the **get** method retrieves the parameter. Here is an example:

```
if (attrs.containsKey("parametername"))
    String parameter = (String)
    attrs.get("parametername")
```

- Select Identity defines the following standard parameters in the map:
 - `WfExternalCall.WF_PARAM_SERVICENAME` — for the service name. This is a `String` object.
 - `WfExternalCall.WF_PARAM_ADMINUSERID` — for the requestor's user name. This is a `String` object.

- `WfExternalCall.WF_PARAM_REQUESTID` — for the request identifier. This is an Integer object.
- `WfExternalCall.WF_PARAM_WORKFLOWINSTID` — for the workflow instance ID. This is an Integer object.
- To retrieve a Select Identity attribute, use one of the methods provided by the `RequestTarget` class, such as **`getSingleRequestParamStr`**. Here is an example that retrieves a single-value attribute:

```
String attributeValue=(String)requestTarget.  
    getSingleRequestParamStr("attributename");
```

To retrieve the values of a multi-value attribute, use the following method:

```
Set attributeValues=(Set)requestTarget.  
    getRequestParam("attributename").getRequestTargetParamValue()
```

- To retrieve workflow variables, use the `IWfQuery` interface as in the following example:

```
IWfQuery query =(IWfQuery)StatelessServiceObjectFactory.  
    create(IWfQuery.class);  
  
int instanceId = query.  
    getInstanceInfoByInstActivityId(instActivityId).getInstId();  
Map varMap = query.getCurrentVariableMap(instanceId);  
String comment = (String) varMap.get("$ApproverComments");
```

- To set and update a workflow variable, use the methods provided by the `IWfDataUpdate` interface, as follows:

```
IWfDataUpdate du =(IWfDataUpdate)StatelessServiceObjectFactory.  
    create(IWfDataUpdate.class);  
du.setWorkflowVar(instId, "workflowvariablename", "value");
```

Update a Select Identity attribute as follows:

```
WfExternalCallStatus ecs = new WfExternalCallStatus(stageId);  
ChangeRecord changeRecord = new ChangeRecord();  
  
// Update the attribute in the change record  
changeRecord.setName("attributename");  
changeRecord.addValue("newattributevalue");  
ecs.addAttributeChange(changeRecord);
```

You can also update workflow variables that need to be persisted using **`setStatus()`** provided by the `WfExternalCallStatus` class. All variable names that starts with \$ is assumed to be a workflow variable and persisted when the call returns.

- To set the return status of a workflow external call, use the `WfExternalCallStatus` class and methods, as follows:

```
ecs = new WfExternalCallStatus(stageId);
```

If the call returns successfully:

```
ecs.setStatus(WfExternalCallStatus.STATUS_APPROVED);
```

If the call is unsuccessful:

```
ecs.setStatus(WfExternalCallStatus.STATUS_REJECT_TERMINATE);
```

- The following variables are available for use in your external call:

Variable Name	Description
<code>_activityId</code>	The activity ID string for current activity in execution. This variable is maintained by the engine and cannot be updated in template. A string value is assigned to this variable.
<code>_instActivityId</code>	An internal variable representing the instance activity ID for the current activity in execution. This variable is maintained by the engine and cannot be updated in template. It is assigned an integer value.
<code>_joinCommand</code>	<p>Passed into the workflow by an external application that is invoked as an action in a workflow template activity. This variable is not persistent. Possible values include</p> <ul style="list-style-type: none"> exit — unconditionally exits the current block exitAll — unconditionally exits the current block and all parent blocks reset — resets the value of the <code>joinCount</code> property set in the block <p>For example, the application could pass this variable to the workflow to instruct it to exit a block unconditionally (even if, for instance, the <code>joinCount</code> value is not met).</p>

Variable Name	Description
<code>pushVar</code>	Passed into the workflow by an external application that is invoked as an action in a workflow template activity. The object is then added to workflow's internal <code>pushList</code> block variable. The pushed objects in the list can be displayed later in a tabular format in a report. You can specify any Java object as the value of this variable.
<code>\$_blockId</code>	The block ID for the current block. You can assign any string value to this variable.
<code>\$_instId</code>	The workflow instance ID. This variable is maintained by the engine and cannot be updated. An integer value is assigned to this variable.

- To log messages and errors, you can use any logging mechanism. Note, however, that the examples shown in this guide use an internal logging mechanism that is not available externally.

Compiling and Deploying an External Call

The following are general steps you must perform to compile and deploy the external call:

- 1 To compile the Java class(es) you create, use the following command:

```
javac -cp clientintf.jar <files>
```
- 2 Copy the class files and any associated file(s) to a directory on the Select Identity server. If you copy the files to a directory in the Select Identity installation path, you will save you a step in the deployment procedure.
 - ▶ Classes that implement external call interfaces cannot reside in the application server's classpath. Instead, you must specify the location in the class path of the external call definition.
- 3 Register the external call with Select Identity using the External Calls home page on the Select Identity client. Refer to the *HP OpenView Select Identity Administrator Guide* for more information.

Examples

This chapter provides an example for each type of external call. Refer to the Javadoc in the `docs/api_help/external_calls` and `workflow` directories on the Select Identity CD for information about the APIs.

Value Generation External Call

This class generates the password from a Social Security Number or ID. The configuration arguments to the function are the `SSN_FIELD` and `PERSONNUMBER_FIELD` attributes.

```
package com.trulogica.truaccess.externalcall.generatefunction;

import java.util.*;

import com.trulogica.truaccess.attribute.TAValueGenerationIntf;
import com.trulogica.truaccess.attribute.exception.*;
import com.trulogica.truaccess.attribute.concero.ejb.*;
import com.trulogica.truaccess.attribute.model.*;
import com.trulogica.truaccess.wfengine.wfexternalcall.*;
import com.trulogica.truaccess.externalcall.model.*;
import com.trulogica.truaccess.externalcall.ejb.*;
import com.trulogica.truaccess.externalcall.util.*;
import com.trulogica.truaccess.externalcall.*;
import com.trulogica.truaccess.request.model.*;
```

```

import com.truologica.truaccess.util.locator.ServiceLocator;
import com.truologica.truaccess.util.TAClassLoader;
import com.truologica.truaccess.attribute.TAValueConstraintIntf;
    TAValueConstraintBeanIntf;

public class PasswordFromSSNorPN implements TAValueGenerationIntf
{
    private TAExternalCallBI mExtCalleJB = null;
    private static final String SSN_FIELD="ssnfield";
    private static final String PERSONNUMBER_FIELD="personnumberfield";
    private static final String PASSWORD_SUFFIX="p2wd";

    public Object generateValue(String attribName, RequestTarget reqTarget,
Map args) throws TAAAttributeDefinitionException
    {
        try {
            String ssnField = (String)args.get(SSN_FIELD);
            if (null == ssnField) {
                throw new Exception("Unable to find the argument:"+SSN_FIELD);
            }

            String personNumberField = (String)args.get(PERSONNUMBER_FIELD);
            if (null == personNumberField) {
                throw new Exception("Unable to find the
                argument:"+PERSONNUMBER_FIELD);
            }

            String password = "";
            String ret = getLast4Chars(reqTarget, ssnField);
            if (ret.length() != 0) {
                password = PASSWORD_SUFFIX + ret;
                return new UserNameValueBean(password);
            }

            ret = getLast4Chars(reqTarget, personNumberField);
            if (ret.length() == 0) {
                throw new Exception("Unable to generate password");
            }

            password = PASSWORD_SUFFIX + ret;
            return new UserNameValueBean(password);
        } catch (Throwable t) {
            attribute:"+attribName+": "+t.getMessage();

            TAAAttributeDefinitionException exp = new
            TAAAttributeDefinitionException("Unable to generate attribute value
            for attribute:"+attribName);

            exp.setCausedBy(t);
        }
    }
}

```

```
        throw exp;
    }
}

private String getLast4Chars(RequestTarget reqTarget, String attrName)
    throws Exception
{
    String value = reqTarget.getParamValueString(attrName);
    if (value == null)
        return "";

    value = value.trim();
    if (value.length() < 4)
        return "";

    return value.substring(value.length() - 4);
}
}

class UserNameValueBean implements TAValueConstraintBeanIntf
{
    String value = null;
    public UserNameValueBean(String _value)
    {
        value = _value;
    }
    public String getName() {
        return null;
    }

    public Object getValue() {
        return value;
    }
}
}
```

Value Constraint External Call

This class implements a simple table-based lookup of possible constraint values. It uses the external arguments —poolname and SQL string— to query the database tables.

```
package com.truologica.truaccess.externalcall.constraintfunction;

import java.util.*;

import com.truologica.truaccess.base.TAFilter;
import com.truologica.truaccess.attribute.TAValueConstraintIntf;
import com.truologica.truaccess.attribute.exception.
    TAAAttributeDefinitionException;

import com.truologica.truaccess.externalcall.constraintfunction.
    SearchResult;

import com.truologica.truaccess.util.Tools;
import com.truologica.truaccess.util.DBTool;
import javax.naming.InitialContext;
import javax.sql.DataSource;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.Types;

public class SearchTable implements TAValueConstraintIntf {

    public static final String PARAM_POOLNAME = "poolname";
    public static final String PARAM_SQLSTRING = "query";
    public static final String PARAM_VALUEFIELD = "valuefield";

    private String poolname = null;
    private String query = null;
    private String valuefield = null;

    /**
     * Returns a List of all possible values.
     * Since this list can be very large it should be used sparingly.
     */

    public List getValueConstraint(String attribName, Map args)
        throws TAAAttributeDefinitionException
    {
        return getValueConstraint(attribName,null,args);
    }
}
```



```

public List getValueConstraint(String attribName, TAFilter filter,
    Map args) throws TAAAttributeDefinitionException
{
    Connection connection = null;
    PreparedStatement pStmt = null;
    ResultSet rs = null;
    ArrayList ar = new ArrayList();
    try {
        poolname = (String)args.get(PARAM_POOLNAME);
        query = (String)args.get(PARAM_SQLSTRING);
        valuefield = (String)args.get(PARAM_VALUEFIELD);

        checkParam(poolname, PARAM_POOLNAME);
        checkParam(query, PARAM_SQLSTRING);
        checkParam(valuefield, PARAM_VALUEFIELD);

        String modFilter = "";
        if (null != filter) {

            switch (filter.getOperation()) {
                case TAFilter.EQUALITY:
                    modFilter = " "+valuefield+" = ?";
                    break;
                case TAFilter.BEGINS_WITH:
                    modFilter = " "+valuefield+" LIKE ?";
                    filter.setValue(filter.getValue()+"%");
                    break;
                case TAFilter.ENDS_WITH:
                    modFilter = " "+valuefield+" LIKE ?";
                    filter.setValue("%"+filter.getValue());
                    break;
                case TAFilter.CONTAINS:
                    modFilter = " "+valuefield+" LIKE ?";
                    filter.setValue("%"+filter.getValue()+"%");
                    break;
            }

            query = query.toUpperCase();

            if (modFilter.length() > 0) {
                if (query.indexOf(" WHERE ") > 0) {
                    query = query + " AND " + modFilter;
                }
            }
        }
        connection = this.getConnection();

        pStmt = connection.prepareStatement(query);
    }
}

```

```

if (modFilter.length() > 0) {
    pstmt.setString(1,filter.getValue());
}

rs = pstmt.executeQuery();
ResultSetMetaData rsMetaData = rs.getMetaData();
int nameColumn=0;
//Select the first String field whose column name does not match
// the value field
for (int i = 1; i <= rsMetaData.getColumnCount();i++)
{
    if (((rsMetaData.getColumnType(i)==Types.VARCHAR)||
        (rsMetaData.getColumnType(i)==Types.CHAR))
        && (!rsMetaData.getColumnName(i).equalsIgnoreCase(valuefield))) {
        nameColumn=i;
        break;
    }
}

while (rs.next())
{
    SearchResult sr = new SearchResult();

    sr.setValue(rs.getString(valuefield));

    if (nameColumn>0)
        sr.setName(rs.getString(nameColumn));
    else
        sr.setName((String) sr.getValue());

    ar.add(sr);
}

return ar;
} catch (Exception e) {
    TAAttributeDefinitionException exp =
        new TAAttributeDefinitionException();
    exp.setCausedBy(e);
    throw exp;
} finally {
    DBTool.close(rs);
    DBTool.close(pstmt);
    DBTool.close(connection);
}
}

private Connection getConnection() throws Exception
{
    InitialContext ctx = null;

```

```

try {
    ctx = new InitialContext();
    DataSource ds = (DataSource)ctx.lookup(poolname);
    return ds.getConnection();
} finally {

    Tools.close(ctx);
}
}

private void checkParam(String value,String name) throws Exception
{
    if ((null == value)|| (value.trim().length() == 0))
        throw new Exception("The value of "+name+" is needed");
}
}

```

Value Validation External Call

This class determines whether the value is an alphanumeric.

```

package com.trulogica.truaccess.externalcall.validation;
import com.trulogica.truaccess.attribute.TAValueValidationIntf;
import com.trulogica.truaccess.attribute.exception.
    TAAAttributeValueValidationException;
import java.util.*;
public class IsAlphaNumeric implements TAValueValidationIntf
{
    public void validateValue(String attribName,Object value, Map args)
        throws TAAAttributeValueValidationException
    {
        String password = (String ) value;
        boolean containsDigit = false;
        boolean containsLetter = false;
        for (int i = 0, n = password.length(); i < n; i++)
        {
            // checkNum(word.charAt(i));
            if (Character.isDigit(password.charAt(i)))
            {
                containsDigit = true;
                break;
            }
        }

        for (int i = 0, n = password.length(); i < n; i++)
        {

```

```

        if (Character.isLetter(password.charAt(i)))
        {
            containsLetter = true;
            break;
        }
    }

    if( !containsLetter || !containsDigit )
    {
        throw new TAAAttributeValueValidationException("The password is not
        alphanumeric. Password has to contain at least one numeric and one
        non-numeric field");
    }
}
}
}

```

Approver Selection External Call

The following example creates a class that collects information from an existing request and concatenates the information to produce a new value for an attribute. The **getApprover()** function is called by Select Identity to get an approver.

```

package com.trulogica.truaccess.wfenginesvcs.wfexternalcall.support;

import
com.trulogica.truaccess.wfengine.wfexternalcall.WfSelectApproverIntf;
import
com.trulogica.truaccess.wfengine.wfexternalcall.WfExternalCallException;
import com.trulogica.truaccess.request.model.RequestTarget;
import com.trulogica.truaccess.util.logging.misc.*;
import java.util.*;

public class WfSelectApproverSample implements WfSelectApproverIntf
{
    /**
     * @param reqtarget is the Request target associated with the workflow
     * @param attrs a list of parameters (standard and custom) for the call
     * The standard parameters are
     *   serviceid: Internal identifier of the service
     *   servicename: Name of the service
     *   roleid: The role name of the approver
     * The specific parameter is SampleApprovers, which contains comma
     * separated user ids
     *
     */
}

```

```

    * @return The collection of Concero userid of the approver
    * @throws ApprovalStageException
    */
public Collection getApprover(RequestTarget reqTarget, HashMap attrs)
    throws WfExternalCallException
{
    ArrayList ret = new ArrayList();
    try
    {
        String value = (String) attrs.get("SampleApprovers");
        if (value != null)
        {
            StringTokenizer tokens = new StringTokenizer(value, ",");
            while (tokens.hasMoreTokens())
            {
                String user = tokens.nextToken();
                ret.add(user);
            }
        }
    }
    catch (Exception e)
    {
    }
    return ret;
}
}

```

Workflow External Call

Two examples are provided in this section. The first example provides two files: the first file generates an attribute value and adds it to a user, and the second file extends the first by adding an ID. The second example changes a user's attributes and entitlements based on whether the user's department has changed.

Adding a Generated Value to a User

This class generates the new attribute value and adds the value to the user in Select Identity.

```
package com.trulogica.truaccess.wfenginesvcs.wfexternalcall.support;

import com.trulogica.truaccess.wfengine.wfexternalcall.WfExternalCall;
import com.trulogica.truaccess.wfengine.wfexternalcall.AttributeRecord;
import com.trulogica.truaccess.wfengine.wfexternalcall.
    WfExternalCallStatus;
import com.trulogica.truaccess.wfengine.wfexternalcall.
    WfExternalCallException;
import com.trulogica.truaccess.request.model.RequestTarget;
import com.trulogica.truaccess.request.model.RequestTargetParam;
import com.trulogica.truaccess.request.model.RequestTargetParamValue;
import com.trulogica.truaccess.wfengine.wfexternalcall.ChangeRecord;
import com.trulogica.truaccess.base.constants.UserAttributeConstants;
import java.util.*;
//import com.trulogica.truaccess.workflow.external.util.*;

public abstract class WorkflowStepSample implements WfExternalCall
{
    String attribName = "";

    protected WorkflowStepSample(String _attribName)
    {
        attribName = _attribName;
    }

    public WfExternalCallStatus process(String stageId,
        RequestTarget requestTarget,
        AttributeRecord [] availGrp,
        AttributeRecord [] availRole,
        AttributeRecord [] availEntitlements,
        Map map) throws WfExternalCallException
    {
        try{

            WfExternalCallStatus ecs = new WfExternalCallStatus(stageId);
            String propPrepName = "", propName = "";

            propName = attribName + ".attributes";

            String attrs = System.getProperty(propName);
            if (null == attrs) {
                ecs.setStatus(WfExternalCallStatus.STATUS_REJECT_TERMINATE);
                return ecs;
            }
        }
    }
}
```

```

StringBuffer sb = new StringBuffer();
ChangeRecord changeRecord = new ChangeRecord();
changeRecord.setName(attribName);

StringTokenizer st = new StringTokenizer( attrs, ", " );
while( st.hasMoreTokens() ) {
    String attrName = st.nextToken();
    String value = requestTarget.getParamValueString(attrName);
    if (null == value) {
        throw new Exception("Unable to generate the value for
            attribute:"+attrName);
    }

    sb.append(value);
}

changeRecord.addValue(sb.toString());
ecs.addAttributeChange( changeRecord );

    getSingleRequestParamStr(UserAttributeConstants.FIRST_NAME );

ecs.setStatus(WfExternalCallStatus.STATUS_APPROVED);
return ecs;
}

catch( Exception e ) {
    throw new WfExternalCallException( e );
}
}
}

```

The following class extends the **WorkflowStepSample** class created above and defines a class that obtains the name of the user.

```

package com.truologica.truaccess.wfenginesvcs.wfexternalcall.support;
import com.truologica.truaccess.wfenginesvcs.wfexternalcall.support.
    WorkflowStepSample;

public class PersonNumberCallout extends WorkflowStepSample
{
    public PersonNumberCallout() {
        super("personId");
    }
}

```

Retrieving and Changing Attributes and Entitlements

The following code creates a class that determines whether a user's department (attribute) changed. If it has changed, the class removes entitlements associated with the old department, changes the cost center attribute, and adds new entitlements based on the new department.

```
package com.truologica.truaccess.wfenginesvcs.wfexternalcall.support;

import java.sql.*;
import java.util.*;
import javax.naming.*;
import javax.sql.*;

import com.truologica.truaccess.request.model.*;
import com.truologica.truaccess.util.logging.misc.*;
import com.truologica.truaccess.wfengine.wfexternalcall.*;

public class WfEntitlementChange
    implements WfExternalCall
{
    private static final Logger mLogger =
        Logger.getLogger(WfEntitlementChange.class.getName());

    public static final String PARAM_POOLNAME = "jdbc/TruAccess";

    //set the resource name here
    private static final String resourceName = "LDAP_70";

    private static final String USERNAMEATTRNAME = "UserName";
    private static final String DEPARTMENTATTRNAME = "Department";
    private static final String COSTCENTERATTRNAME = "CostCenter";
    private static final String ENTITLEMENTPOSTFIX = "_ENTITLEMENTS";

    private static final String findOldDeptQuery = "select A.stringValue
from TAAAttribute A , "
        + "TAUser B where A.identObjId = B.userId AND A.name = "
        + "? and B.conceroUserId = ?";

    /*
    * If the department has changed, assign the new cost center and
    * entitlements to the user. Use the following table to assign values.
    * Department Entitlements CostCenter
    * -----
    * Sales SA-505 101
    * Finance FIN-505 205
    * HR HR-101 308
    * Corporate CORP-3 409
    */
}
```



```

private static final String[] deptNames =
{
    "Sales", "Finance", "HR", "Corporate"};

private static final String[] costCenterNames =
{
    "101", "102", "103", "100"};

private static final String[] entNames =
{
    "SA-103", "FIN-101", "HR-102", "Corp-103"};

public WfExternalCallStatus process(String stageId, RequestTarget
reqTarget, AttributeRecord[] attrbRecl, AttributeRecord[] attrbRec2,
AttributeRecord[] attrbRec3, Map map) throws WfExternalCallException
{
    Connection connection = null;
    PreparedStatement pStmt = null;
    ResultSet rs = null;

    try
    {
        //get service name from map
        //String serviceName =
(String)map.get(WfExternalCall.WF_PARAM_SERVICENAME);

        WfExternalCallStatus status = new WfExternalCallStatus(stageId);

        //Get the new department and user name from the Request Target
        String newDept =
reqTarget.getParamValueString(DEPARTMENTATTRNAME);
        String userName =
reqTarget.getParamValueString(USERNAMEATTRNAME);

        //Get the old department from SI database
        String oldDept = null;
        connection = this.getConnection();

        pStmt = connection.prepareStatement(findOldDeptQuery);
        pStmt.setString(1, DEPARTMENTATTRNAME);
        pStmt.setString(2, userName);

        rs = pStmt.executeQuery();
        if (rs.next())
        {
            oldDept = rs.getString(1);

```

```

        if (null == oldDept)
        {
            oldDept = "";
        }
    }
    else
    {
        log("There is no old department");
        oldDept = "";
    }

    //Find whether the department has been changed by the
    //reconciliation process
    boolean changed = false;
    if (!oldDept.equalsIgnoreCase(newDept))
    {
        changed = true;
        //Build the map tables
    }
    Map entMap = new HashMap();
    Map costMap = new HashMap();

    for (int i = 0; i < deptNames.length; i++)
    {
        entMap.put(deptNames[i], entNames[i]);
        costMap.put(deptNames[i], costCenterNames[i]);
    }

    //Form the resource attribute name from the resource name.
    //Entitlement attribute names always ends with _ENTITLEMENTS.
    String resourceAttrib = resourceName + ENTITLEMENTPOSTFIX;

    if (changed)
    {
        //Create a change record to add the costcenter value
        //or replace the value if present.
        //CostCenter attribute is a single value attribute.

        ChangeRecord costCenterCR = new ChangeRecord();
        costCenterCR.setName(COSTCENTERATTRNAME);
        if(costMap.get(newDept) != null )
            costCenterCR.addValue( (String) costMap.get(newDept));

        status.addAttributeChange(costCenterCR);

        //Add entitlements to the user
        ChangeRecord entitlementsCR = new ChangeRecord();
        entitlementsCR.setName(resourceAttrib);
        if(entMap.get(newDept) != null )
            entitlementsCR.addValue( (String) entMap.get(newDept));
    }

```

```

    /*
     * If you need to add more entitlements, use:
     * cr.addValue("Some other entitlements");
     *
     * To delete attribute values, use:
     * cr.setOperation(ChangeRecord.DELETE);
     */

    //Entitlements are multi-value attributes. need to set change
    //operation ADD/MODIFY/... Default is ADD
    entitlementsCR.setChangeOperation(ChangeRecord.ADD);
    status.addAttributeChange(entitlementsCR);

    //set a work flow variable
    //workflow variable name starts with __
    //ChangeRecord _wfVar1 = new ChangeRecord();
    //_wfVar1.setName("__WFVAR");
    //_wfVar1.addValue("Var value");
}

status.setStatus(WfExternalCallStatus.STATUS_APPROVED);
return status;
}
catch (Throwable e)
{
    log("Unable to complete the department change", e);
    WfExternalCallException exp = new
        WfExternalCallException("Unable to complete the department
            change", e);
    throw exp;
}

finally
{
    //Close all database connections, statements and result sets
    try
    {
        if (null != rs)
        {
            rs.close();
        }
        rs = null;
    }
    catch (Throwable t)
    {
        log("Error in closing resultset", t);
    }
}

```

```
        try
        {
            if (null != pStmt)
            {
                pStmt.close();
            }
            pStmt = null;
        }
        catch (Throwable t)
        {
            log("Error in closing statement", t);
        }

        try
        {
            if (null != connection && !connection.isClosed())
            {
                connection.close();
            }
            connection = null;
        }
        catch (Throwable t)
        {
            log("Error in closing connection", t);
        }
    }
}

private Connection getConnection() throws Exception
{
    InitialContext ctx = null;
    try
    {
        ctx = new InitialContext();
        DataSource ds = (DataSource) ctx.lookup(PARAM_POOLNAME);
        return ds.getConnection();
    }
    finally
    {
        try
        {
            if (null != ctx)
            {
                ctx.close();
            }
            ctx = null;
        }
    }
}
```

```
        catch (Throwable t)
        {
            log("Error closing context", t);
        }
    }

    private void log(String msg, Throwable t)
    {
        System.out.println(msg + ":Caused By:");
        t.printStackTrace();
    }

    private void log(String msg)
    {
        System.out.println(msg);
    }
}
```