HP OpenView Select Identity

For the Red Hat Enterprise Linux and
Windows 2003 Operating Systems

Software Version: 4.0

External Call Developer Guide

March 2006

invent

Trademark Notices

Support

Please visit the HP OpenView support web site at:

http://www.hp.com/managementsoftware/support

This web site provides contact information and details about the products, services, and support that HP OpenView offers.

HP OpenView online software support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valuable support customer, you can benefit by using the support site to:

- Search for knowledge documents of interest
- Submit enhancement requests online
- Download software patches
- Submit and track progress on support cases
- Manage a support contract
- Look up HP support contacts
- Review information about available services
- Enter discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and log in. Many also require a support contract.

To find more information about access levels, go to:

http://www.hp.com/managementsoftware/access_level

To register for an HP Passport ID, go to:

http://www.managementsoftware.hp.com/passport-registration.html

Preface

Welcome to the *HP OpenView Select Identity Workflow Studio Guide*. This guide provides detailed information regarding the use of HP OpenView Select Identity's (OVSI) Workflow Studio, which enables you to create a workflow template that automates the actions that approvers and systems management software must perform.

About This Guide

The *HP OpenView Select Identity Workflow Studio Guide* is designed to help you understand the concepts of workflows and how to create workflow templates.

Audience

This document is intended for administrators who will use the HP OpenVew Select Identity Workflow Studio to create and manage workflow templates.

Typographical Conventions

This guide uses the following typographical conventions:

Convention	**Description**
Bold	Used for user interface elements (menus, buttons, and so on), new terms, and URLs.
Italics	Used for variables, book titles, and emphasis.
`Monospace`	Used for code examples, directory and file names, commands, and user input.

Product Documentation

The HP OpenView 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 included in this release, 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.

- Detailed procedures for deployment and system management are documented in the *HP OpenView Select Identity Administrator Guide* and OVSI online help system. This guide provides detailed concepts and procedures for deploying and configuring the OVSI system. In the online help system, tasks are grouped by the administrative functions that govern them.
- The *HP OpenView Select Identity My Identity User Guide* provides detailed information for end-users about the My Identity function, which allows users to manage their identity information.
- 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.
- An *HP OpenView Connector Installation and Configuration Guide* is provided for each resource connector. These are located on the Select Identity Connector CD.
- The *HP OpenView Select Identity Attribute Mapping Utility User Guide* describes how to access the Attribute Mapping Utility, provides an overview to the utility's user interface, and describes how to define user and entitlements mappings. This guide is provided on the Select Identity Connector CD and is for use with the SQL and SQL Admin connectors only.
- 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 OVSI 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. This guide also describes the Web Service, which enables you to programmatically provision users in OVSI, providing an overview of the operations you can perform through use of the Web Service, including SPML examples for each operation. 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. Also, an independent, web-based help system is available for the Web Service 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.

Contents

1 Introduction to External Calls

HP OpenView Select Identity (Select Identity) supports the ability to invoke calls to external systems. External calls can be used to perform the following types of tasks:

- Generate a user ID or password. See Attribute Value Generation External Call Type on page 5.
- Provide a list of possible attribute values. See Attribute Value Constraint External Call Type on page 7.
- Validate the value of an attribute. See Attribute Value Validation External Call Type on page 8.
- Verify the value of an attribute. See Attribute Value Verification External Call Type on page 9.
- Query an external system for a list of approvers. See Approver Selection External Call Type on page 9.
- Perform a workflow task. See Workflow Action External Call Type on page 10.
- Retrieve a certificate from an external system. Certification Management Function External Call Type on page 13.

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

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 or later. 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 API and Workflow API in the `/docs/api_help/external_calls/Javadoc` directory on the Select Identity media. Refer to this web-based help for implementation and usage details.

The External Call API provides the following classes and interfaces, which are available for creating value external calls.

- **TAValueConstraintIntf**

This interface must be implemented by external call classes providing a set of possible values for an attribute. For example, implement this interface for an external call that provides a list of department codes to present options to an end user or to validate data. Note that SIConstraintIntf is an extension of this interface.

- **TAValueConstraintIntf.TAValueConstraintBeanIntf**

  External calls generating or providing a set of possible values for an attribute must return the values in a Java bean that implements this interface. For example, the department codes returned by the external call must be passed in a Java bean implementing this interface.

- **SIConstraintIntf**

  This interface must be implemented by classes generating possible values for an attribute.

- **TAValueGenerationIntf**

  This interface must be implemented by external call classes generating attribute values. For example, the class implementing this interface can generate a random password for a user.

- **TAValueValidationIntf**

  This interface must be implemented by external call classes validating attribute values. For example, if a value is present in a form (entered by an user), the implementing class can validate the value.

- **TAPolicyVerificationIntf**

  This interface must be implemented by external calls that validate the value of an attribute for a particular user. For example, you use a class implementing this interface to verify that a user's password that is stored externally (outside of Select Identity) is correct. Currently, only password attributes are verified this way.

- **TAAttributeDefinitionException** and **TAAttributeValueValidationException**

  Exceptions defined for use by external calls.

The Workflow API provides the following classes and interfaces to 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 an inactivated workflow instance and 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 invoked by workflow instances must implement this interface. This class returns a collection of approvers.

- **WfExternalCallException**

  Defines exceptions used 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.

Use the following classes to explore Select Identity requests 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 to 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**

  Lets you set and get parameter values.
- **TAFilter**

  A general class that stores filter criteria for a selection (search) procedure.
- **TARequestAction**

  Represents the action to take place on the target resource.
- **TARequestEvent**

  Represents the event for the request.
- **TARequestType**

  Represents the type of request.

2 Default External Calls

Select Identity provides default external calls to let you interact with external systems for workflow steps and approver lookups. Each external call is defined within one of the following call types:

- **Attribute Value Generation** — generates the name or ID of a user, the user's password, and any other attribute, such as the user's company or department. See Attribute Value Generation External Call Type on page 5.
- **Attribute Value Constraint** — provides a list of possible values for an attribute. See Attribute Value Constraint External Call Type on page 7.
- **Attribute Value Validation** — validates the value of an attribute. See Attribute Value Validation External Call Type on page 8.
- **Attribute Value Verification** — verifies the value of an attribute. See Attribute Value Verification External Call Type on page 9.
- **Approver Selection** — searches an external system for a list of users who can approve provisioning requests during a workflow. See Approver Selection External Call Type on page 9.
- **Workflow Action** — performs a task as part of a workflow, letting you integrate approval processes with external processes and systems. See Workflow Action External Call Type on page 10.
- **Certification Management** — enables you to retrieve a certificate from an external system. See Certification Management Function External Call Type on page 13.

Most external calls have predefined parameters that you can modify. The following sections list and describe the functionality of the external calls and their parameters, by call type.

Attribute Value Generation External Call Type

Attribute value generation external call types generate the name or ID of a user, the user's password, and any other attribute, such as the user's company or department. Attribute value generation external calls include the following:

- `IDValueGeneration`

- `PasswordValueGeneration`
- `UserIDValueGeneration`

IDValueGeneration External Call

The`IDValueGeneration` external call generates an attribute that is a unique number.

Parameters:

Parameter Name	Parameter Value Description
Suffix	Use after the number.
Prefix	Use before the number.

PasswordValueGeneration External Call

The `PasswordValueGeneration` external call generates a password that may contain letters and numbers. The password must contain at least one number, and the letters must be lowercase. the password value is constrained by the minimum and maximum parameters. Special characters ("/", "+", "-") are not permitted.

Parameters:

Parameter Name	Parameter Value Description
minLength	The smallest allowable length of the password.
maxLength	The largest allowable length of the password.

UserIDValueGeneration External Call

The `UserIDValueGeneration` external call generates a user ID based on another attribute.

Parameters:

Parameter Name	Parameter Value Description
MaxRetryAttempts	Maximum number of tries to create a unique ID.
Length	The number of characters (alphanumeric or special) in the generated ID.
AttributeName	Attribute name from which the user ID is generated (such as from email).

Attribute Value Constraint External Call Type

Attribute value constraint external call types provide a list of possible values for an attribute. Attribute value constraint external call types include the following:

- `Search Connector`
- `Search Table`

Search Connector External Call

The `Search Connector` external call constrains attributes based on the Select Identity resource name specified.

Parameters:

Parameter Name	Parameter Value Description
resource_name	Select Identity resource name.

Search Table External Call

The `Search Table` external call constrains attributes based on the specified `query` and `valuefield` parameters. The query is executed using the specified `poolname` parameter value.

Parameters:

Parameter Name	Parameter Value Description
valuefield	Value from the query to use for constraining the attribute.
query	Query invoked to dynamically lookup valid values from the data-base.
poolname	JNDI name for the data source and poolname for which the query is to be executed.

Attribute Value Validation External Call Type

Attribute value validation external call types validate values of attributes. Attribute value validation external call types include the following:

- `IsAlphaNumeric`
- `ManageExpireValidation`
- `PasswordValidation`
- `AttributeValueValidation`

IsAlphaNumeric External Call

The `IsAlphaNumeric` external call validates if the attribute value is alphanumeric. No parameters are editable with this external call.

ManageExpireValidation External Call

The `ManageExpireValidation` external call validates the value of the `ExpirationDate` attribute, which must be more than 30 days from the current date. An error message appears if the value of the `ExpirationDate` attribute is less than 30 days. No parameters are editable with this external call.

PasswordValidation External Call

The `PasswordValidation` external call validates that the password contains the minimum number of each type of characters specified.

Parameters:

Parameter Name	Parameter Value Description
Special Characters	Number of required non-alphanumeric or special characters.
List of Special Characters	Comma delimited list of valid special characters.
Lower Case Letters	Number of required lowercase letters
Upper Case Letters	Number of required uppercase letters.
Numerics	Number of required numeric values.
Letters	Number of required letters.

Attribute Value Verification External Call Type

The `AttributeValueVerification` external call verifies the value of an attribute. No parameters are editable with this external call.

Approver Selection External Call Type

Approver selection external call types search an external system for a list of users who can approve provisioning requests during a workflow. Approver selection external call type includes `WFGetApproverSampleExtCall`.

WFGetApproverSampleExtCall External Call

The `WFGetApproverSampleExtCall` external call is a sample external call that specifies a list of users to use for approvals.

Parameters

Parameter Name	Parameter Value Description
SampleApprovers	Comma delimited list of users to use for approvals.

Workflow Action External Call Type

Workflow external call types perform a task as part of a workflow, letting you integrate approval processes with external processes and systems. Workflow external calls include the following:

- `LoadUserServices`
- `UserEnableDisableWFExtCall`
- `WorkflowCertificateRequest`
- `ExclusionRuleCall`

LoadUserServices External Call

The `LoadUserServices` external call adds Services to a user based on context change. For an example of how to use this external call, see Scenario: Adding Services to a User within the *Workflow Studio Guide*.

Parameters:

Parameter Name	Parameter Value Description
ServicesRule	Specifies the rule name.

UserEnableDisableWFExtCall External Call

The `UserEnableDisableWFExtCall` external call enables or disables a user based on the value stored in a specified attribute.

Parameters:

Parameter Name	**Parameter Value Description**
AttributeName	Attribute name for which the value is checked.
EnableValue	If the value of the attribute of the user matches the `EnableValue`, the external call enables the user when the user is disabled.
DisableValue	If the value of the attribute of the user matches the `DisableValue`, the external call disable the user when the user is enabled.
UserName	Administrator with authority to modify users who will use this external call.
Password	Administrator's password.
url	Webservices URL.

WorkflowCertificateRequest External Call

The `WorkflowCertificateRequest` manages certificates. For information on using this external call, see Chapter 2 in the *HP OpenView Select Identity Workflow Studio Guide*.

Parameters:

Parameter Name	Parameter Value Description
DN_FieldName	Attribute name storing the user's distinguished name (DN) from the certificate.
CertificateFieldName	Challenge password assigned at the time of user registration.
EmailTemplateName	Default email template from Select Identity, to send email to the user.
CertificateProviderName	Certificate provider name. In the case of Verisign, the name must be "Verisign." In all other cases, the administrator can assign the name.
ExternalCallName	Name of the CA-specific Java class implementing validation and generation functions for the certificate.

ExclusionRuleCall External Call

The `ExclusionRuleCall` external call prevents users from receiving services when the users have existing or pending conflicting services or entitlements.

Parameters:

Parameter Name	Parameter Value Description
RuleName UniqueServicePrefix	Parameter specifying which rule to run.
WFVariableName $RuleExclusionMsg	Workflow variable specifying the result message.

Certification Management Function External Call Type

Certification management function external call types implement validation and generation functions for the certificate. Certification management function external call types include only `VerisignCertImpl`. For detailed information about Verisign certificate management, see Appendix E in the *HP OpenView Select Identity Workflow Studio Guide*.

VerisignCertImpl External Call

The `VerisignCertImpl` external call is called by the `WorkFlowCertificateRequest` external call, which validates certificate requests. This external call has no editable paramenters. For more information, see Chapter 2 in the *HP OpenView Select Identity Workflow Studio Guide*.

3 Creating an External Call

The sections in this chapter 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 HP OpenView Select Identity media for details about specific APIs.

Coding Attribute Value External Calls

To create an external call, follow these guidelines:

- Attribute Value Generation external calls must implement the `TAValueGenerationIntf` interface. See Attribute Value Generation External Call on page 25 for code examples.
- Attribute Value Constraint external calls must implement the `TAValueConstraintIntf` and `SIConstraintIntf` interfaces. See Attribute Value Constraint External Call on page 27 for code examples.
- Attribute Value Validation external calls must implement the `TAValueValidationIntf` interface. See Attribute Value Validation External Call on page 48 for code examples.
- Attribute Value Verification external calls must implement the `TAPolicyVerificationIntf` interface.
- Certification management external calls must implement the `CertificateMgmt interface` interface.

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

- To retrieve an external call parameter 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. For 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`. This example retrieves a single value attribute:

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

  To retrieve the values of a multivalue attribute, use the `getRequestParam` method. For example:

  ```
  Set attributeValues=(Set)requestTarget.

  getRequestParam("attributename").getRequestTArgetParamValue(
  )
  ```

- Throw `TAAttributeDefinitionException` if the external call is unsuccessful.
- If you write an external call function that must access data in the Select Identity database, the function can acccess the database using the JNDI name specified by the `truaccess.dataSource` property in the `truaccess.properties` file. However, the function can access the Select Identity database in other ways; it depends on how you construct the function. For more information about properties set in this file, see the *HP OpenView Select Identity Installation and Configuration Guide*.
- Use any logging mechanism to log messages and errors. The examples shown in this guide, however, use an internal logging mechanism 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. The call is required to return status indicating 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 and 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, the external call must return data that is valid in Select Identity.
- To retrieve an external call parameter defined on the `External Calls` page of the Select Identity client, use `containsKey` and `get` methods. The `containsKey` method verifies the existence of the specified parameter, and the `get` method retrieves the parameter. For 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 parameter is empty for a self-registration or system-generated request. 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`. For example, you may need to populate the values for each user attribute defined by a map in an external call called by a workflow template.

  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 a more extensive example, see Retrieving Request Object Data to Set Workflow Variables on page 61.

- 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");
  ```

- A workflow variable can be any Java object. However, if it is a persistent variable with its name preceded by `$`, you may either use HashMap to contain object attributes or use the Java object marked as Serializable. If the persistent variable is a Java object, you should also add following line in the Java class:

  ```
  static final long serialVersionUID = uid;
  ```

  where `uid` can be `0` for the newly created class or found by running JDK's server command.

- 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. The variable is assigned an integer value.
_joinCommand	Passed into the workflow by an external application invoked as an action in a workflow template activity. This variable is not persistent. Possible values include the following: • `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. 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).

Variable Name	Description
_pushVar	Passed into the workflow by an external application 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. You can specify any Java object as the value of this variable.
$_blockId	The block ID for the current block. You can assign any string value to this variable.
$_instId	The workflow instance ID. This variable is maintained by the engine and cannot be updated. An integer value is assigned to this variable.

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

Coding Workflow Action External Calls

For workflow action 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. The callout is required to return status indicating how Select Identity proceeds with the workflow.

General information about how to implement an workflow action external call, how to obtain to input parameters and Select Identity attributes, and what needs to be returned are provided:

- 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 action external call must return `WfExternalCallStatus`. In addition, the external call must return data valid in Select Identity.
- To retrieve an external call parameter 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`. This example retrieves a single value attribute:

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

  To retrieve the values of a multivalue attribute, use the `getRequestParam` method. For example:

  ```
  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 starting with `$` are 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, use the following:

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

  If the call is unsuccessful, use the following:

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

- 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.
_joinCommand	Passed into the workflow by an external application invoked as an action in a workflow template activity. This variable is not persistent. Possible values include • `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 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).

Variable Name	Description
_pushVar	Passed into the workflow by an external application 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. You can specify any Java object as the value of this variable.
$_blockId	The block ID for the current block. You can assign any string value to this variable.
$_instId	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. The examples shown in this guide, however, use an internal logging mechanism that is not available externally.

4 Examples

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

➤ All of the examples are included in the Select Identity EAR file deployed during installation. If you wish to modify and register one of the example calls, you must change the class name.

Attribute 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. The supporting class, `UserNameValueBean.java`, follows this one on page 27.

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

import java.util.StringTokenizer;
import java.util.Random;
import java.util.Map;
import com.trulogica.truaccess.request.model.RequestTarget;
import com.trulogica.truaccess.attribute.TAValueGenerationIntf;
import com.trulogica.truaccess.attribute.exception.
TAAttributeDefinitionException;
import com.trulogica.truaccess.util.logging.misc.Logger;
import com.trulogica.truaccess.util.logging.misc.Level;
import com.trulogica.truaccess.externalcall.generatefunction.
UserNameValueBean;
/**
 * Generates a String ID with the format Prefix + ID + Suffix
 * Prefix and Suffix are specified as parameters to the call
 */
```

```
public class SimpleValueGenerator implements
TAValueGenerationIntf
{
  public Object generateValue(String attribName,
           RequestTarget reqTarget, Map args)
           throws TAAttributeDefinitionException
  {

    try {

      if (mLogger.isLoggable(Level.FINEST))
mLogger.finest("Generating the
       value of :"+attribName);
      if (mLogger.isLoggable(Level.FINEST))
       mLogger.finest("RequestTarget:"+reqTarget.toString());

      String prefix = (String)args.get(PARAM_PREFIX);
             String suffix = (String)args.get(PARAM_SUFFIX);

             StringBuffer sb = new StringBuffer();

             sb.append((null==prefix)?"":prefix).append((new

Random()).nextInt()).append((null==suffix)?"":suffix);

             if (mLogger.isLoggable(Level.FINEST))
                 mLogger.finest("Returning a value of : " +
sb.toString() +
               " : for attribute" + attribName);
             return new UserNameValueBean(sb.toString());

    } catch (Throwable t) {
      if (mLogger.isLoggable(Level.WARNING))
       mLogger.log(Level.WARNING,"Unable to generate attribute
value for
       attribute:"+attribName,t);
      TAAttributeDefinitionException exp =  new
       TAAttributeDefinitionException("Unable to generate
attribute value
       for attribute:"+attribName+t.getMessage());
      exp.setCausedBy(t);
      throw exp;
```

```
        }
         }

         private static final Logger mLogger =
          Logger.getLogger(SimpleValueGenerator.class.getName());
         private static final String PARAM_PREFIX="prefix";
         private static final String PARAM_SUFFIX="suffix";
}
```

The value generation function aboce calls the following class:

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

import com.trulogica.truaccess.attribute.TAValueConstraintIntf.
TAValueConstraintBeanIntf;

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

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

Attribute Value Constraint External Call

The following examples implement a simple table-based lookup of possible constraint values. The examples use external arguments —poolname and SQL string— to query the database tables. The supporting class, `SearchResult.java`, follows the main classes.

Implementing the TAValueConstraintIntf Interface

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

import java.util.*;
import com.trulogica.truaccess.util.logging.misc.Logger;
import com.trulogica.truaccess.util.logging.misc.Level;

import com.trulogica.truaccess.base.TAFilter;
import com.trulogica.truaccess.attribute.TAValueConstraintIntf;
import com.trulogica.truaccess.attribute.exception.
TAAttributeDefinitionException;

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

import com.trulogica.truaccess.util.Tools;
import com.trulogica.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;

/**
 * This class implements simple table-based lookups of possible
constraint
 * values. It uses the external arguments poolname and sql string
to query
 * the database tables.
 */
public class SearchTable implements TAValueConstraintIntf {

  private static final Logger mLogger =
   Logger.getLogger(SearchTable.class.getName());

  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.
   * @param attribName
   * @param args
   * @return
   * @throws TAAttributeDefinitionException
   * @see
com.trulogica.truaccess.attribute.TAValueConstraintIntf
   */

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

  public List getValueConstraint(String attribName,TAFilter
filter, Map
  args) throws TAAttributeDefinitionException
  {
    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);
      if (mLogger.isLoggable(Level.FINEST))
mLogger.finest("Trying to
       locate values for attribute:"+attribName);
      String modFilter = "";
      boolean hasParam = false;
      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;
        }

        String query2Check = query;
        query2Check = query2Check.toUpperCase();

        if (modFilter.length() > 0) {
          if (query2Check.indexOf(" WHERE ") > 0) {
            query = query + " AND " + modFilter;

          } else {
            query = query + " WHERE " + modFilter;
          }
          hasParam = true;
        }
      }
      connection = this.getConnection();

      pStmt = connection.prepareStatement(query);
      if (mLogger.isLoggable(Level.FINEST))
       mLogger.finest("query:("+query+"):modfilter:"+modFilter);
      if (hasParam) {
        pStmt.setString(1,filter.getValue());
      }

      rs = pStmt.executeQuery();
      ResultSetMetaData rsMetaData = rs.getMetaData();
      int nameColumn=0;
      //Select the first String field whose column name doesnot
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) {
      if (mLogger.isLoggable(Level.WARNING))
       mLogger.log(Level.WARNING,"Unable to obtain the
values",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");
  }
}
```

The following is called by the class listed above:

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

import com.trulogica.truaccess.attribute.TAValueConstraintIntf;
```

```
public class SearchResult implements
TAValueConstraintIntf.TAValueConstraintBeanIntf
{
  private String name;
  private Object value;

  public SearchResult() {
  }

  public String getName() {
    return name;
  }

  public void setName(String _name) {
    name = _name;
  }

  public Object getValue() {
    return value;
  }

  public void setValue(Object _value)
  {
    value = _value;
  }
}
```

Implementing the SIConstraintIntf Interface

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

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.sql.Types;
import java.util.ArrayList;
import java.util.Collection;
import java.util.Iterator;
import java.util.List;
import java.util.Map;

import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.sql.DataSource;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;

import com.hp.ovsi.attribute.SIConstraintIntf;
import com.hp.ovsi.attribute.api.GetConstraintsCommand;
import com.hp.ovsi.attribute.api.GetConstraintsResponse;
import com.hp.ovsi.base.Command;
import com.hp.ovsi.base.Response;
import com.trulogica.truaccess.attribute.exception.
  TAAttributeDefinitionException;
import com.trulogica.truaccess.base.TAFilter;
import com.trulogica.truaccess.base.TAQuery;
import com.trulogica.truaccess.base.exception.CommandException;
import com.trulogica.truaccess.util.DBTool;
import com.trulogica.truaccess.util.Tools;

public class SearchTable implements SIConstraintIntf //
TAValueConstraintIntf
{

    private static final Log LOGGER =
      LogFactory.getLog(SearchTable.class );

     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;

     /**
      * Default NoArg Constructor
      */
     public SearchTable() {
         //end <init>
     }

     public SearchTable(final Map args) throws
      TAAttributeDefinitionException{
         setParams(args);
         //end <init>
     }

     /**
      * Returns a List of all possible values. Because this list
 can be
     * potentially very large, it should be used sparingly.
      *
      * @param attribName
      * @param args
      *
      * @return
      *
      * @throws TAAttributeDefinitionException
      * @see
 com.trulogica.truaccess.attribute.TAValueConstraintIntf
      */
    public List getValueConstraint( String attribName, Map args )
 throws
      TAAttributeDefinitionException {
         return getValueConstraint( attribName, null, args );
     }

     /**
      * @see
 com.trulogica.truaccess.attribute.TAValueConstraintIntf#
     * getValueConstraint(java.lang.String,
 com.trulogica.truaccess.base.
     * TAFilter, java.util.Map)
      */
     public List getValueConstraint( String attribName, TAFilter
 filter,
      Map args ) throws TAAttributeDefinitionException {
         PreparedStatement pStmt = null;
         ResultSet rs = null;
         ArrayList ar = new ArrayList();
         //FIXME: Why store the params, when replaced after every
 call?
         setParams(args);

         if ( LOGGER.isTraceEnabled() ) {
             LOGGER.trace( "Trying to locate values for
 attribute:" +
            attribName );
         }
         String modFilter = "";
         boolean hasParam = false;
         if ( null != filter ) {

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

             String query2Check = this.query;

             query2Check = query2Check.toUpperCase();

             if (modFilter.length() > 0) {
                 if (query2Check.indexOf(" WHERE ") > 0) {
                    this.query = this.query + " AND " + modFilter;

                 } else {
                     this.query = this.query + " WHERE " +
 modFilter;
```

```
                }
                hasParam = true;
            }//end if (modFilter.length() > 0
        }//end if (null != filter
        final Connection connection = getConnection();
        try {

            pStmt = connection.prepareStatement( this.query );

            if ( LOGGER.isTraceEnabled() ) {
                LOGGER.trace( "query:(" + this.query + 
"):modfilter:" + 
              modFilter );
            }

            if ( hasParam ) {
                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(this.valuefield))) {
                    nameColumn = i;
                    break;
                }//end if
            }//end for

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

                sr.setValue(rs.getString(this.valuefield));
                if (nameColumn > 0) {
                    sr.setName(rs.getString(nameColumn));
                } else {
                    sr.setName((String) sr.getValue());
                }

                ar.add(sr);
            }
            return ar;
        } catch (SQLException e) {
            LOGGER.fatal("Unable to obtain the values", e);
            TAAttributeDefinitionException exp =
           new TAAttributeDefinitionException();
            exp.initCause(e);

            throw exp;
        } finally {
            DBTool.close(rs);
            DBTool.close(pStmt);
            DBTool.close(connection);
        }//end finally
        //end getValueConstraint
    }

    /**
     * Gets the Connection to the Database.
     * @return Connection to the database.
     * @throws TAAttributeDefinitionException Thrown if an
Exception
    * occurs looking up DataSource, or making Connection
     * to the Database.
     */
    private Connection getConnection() throws
     TAAttributeDefinitionException {
        InitialContext ctx = null;

        try {
            ctx = new InitialContext();
            DataSource ds = (DataSource)
ctx.lookup(this.poolname);
            return ds.getConnection();
        } catch (NamingException e) {
            final String msg = "Unable to retrieve Data
Connection: " +
            e.getMessage();
            LOGGER.fatal(msg, e);
            throw new TAAttributeDefinitionException(msg);
        } catch (SQLException e) {
            final String msg = "Unable to make Connection to
Database: " +
            e.getMessage();
            LOGGER.fatal(msg, e);
            throw new TAAttributeDefinitionException(msg);
        } finally {
            Tools.close(ctx);
        }
        //end getConnection
    }

    /**
     * Convert the Filter (if supplied) into its SQL format, and
return
     * the SQL Query.
     * @param query Base SQL Query to apply filter to.
     * @param valuefield
     * @param filter
     * @return
     * @deprecated
     * @see TAFilter#getPreparedStmtStr(String)
     */
    protected static String getFilteredQuery(final String query,
            final String valuefield,
            final TAFilter filter) {
        if ( null != filter ) {
            final StringBuffer modFilter = new StringBuffer(32);
            modFilter.append(' ');
            modFilter.append(valuefield);
            switch ( filter.getOperation() ) {
            case TAFilter.EQUALITY:
                modFilter.append(" = ?");
                break;
            case TAFilter.BEGINS_WITH:
                modFilter.append(" LIKE ?");
                filter.setValue( filter.getValue() + "%" );
```

```
                    break;
                case TAFilter.ENDS_WITH:
                    modFilter.append(" LIKE ?");
                    filter.setValue( "%" + filter.getValue() );
                    break;
                case TAFilter.CONTAINS:
                    modFilter.append(" LIKE ?");
                   filter.setValue( "%" + filter.getValue() + "%" );
                    break;
                }

                String query2Check = query;

                query2Check = query2Check.toUpperCase();

                if (modFilter.length() > 1) {
                    if (query2Check.indexOf(" WHERE ") > 0) {
                         return(query + " AND " + modFilter);
                    }// else
                    return(query + " WHERE " + modFilter);
                }//end if (modFilter.length() > 0
            }//end if (null != filter
            //else
            return(query);
            //end getFilteredQuery
        }

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

        /**
         * @see
com.hp.ovsi.attribute.SIConstraintIntf#getConstraints(com.hp.
        * ovsi.attribute.api.GetConstraintsCommand)
```

```
      */
     public GetConstraintsResponse
getConstraints(GetConstraintsCommand
      command) throws TAAttributeDefinitionException{
          //TODO: Normal getConstraintValue
          final TAQuery query = command.getQuery();
          final List filters = query.getTaFilterList();
          //FIXME: This only sends the first filter!
         final TAFilter filter = filters == null || filters.size()
< 1 ? null : (TAFilter)filters.get(0);
          final List constraints =
getValueConstraint(command.getAttrName(),
          filter, command.getParams());
          final GetConstraintsResponse response = new
          GetConstraintsResponse(constraints);
          return(response);
          //end getConstraints
     }

     /**
      * @see
com.hp.ovsi.attribute.SIConstraintIntf#getDisplayName(com.hp.
     * ovsi.base.Command)
      * INPUT = values
      * OUTPUT = DiaplayValues(Name)/Values pairs
      */
     public Response getDisplayName(Command command) throws
     TAAttributeDefinitionException{
          final TAQuery taQuery = command.getQuery();
          final Collection filters = taQuery.getTaFilterList();
          String sqlQuery;
          if (filters != null && filters.size() > 0 ) {
              sqlQuery = this.query + makeWhereClause(filters,
              taQuery.isFilterListAnded());
          } else {
              sqlQuery = this.query;
          }
          LOGGER.debug(sqlQuery);
          final Connection connection = getConnection();
          PreparedStatement ps = null;
          ResultSet rs = null;
          try {
              ps = connection.prepareStatement(sqlQuery);
              if (filters != null && filters.size() > 0 ) {
                  int idx = 1;
                  for (Iterator it = filters.iterator();
  it.hasNext();idx++) {
                      TAFilter filter = (TAFilter) it.next();
                     ps.setObject(idx, filter.getValueAsObject());
                  }//end for
              }//end if
              rs = ps.executeQuery();
              final int valCol = rs.findColumn(this.valuefield);
              // The Name column should be either the first or
  second column
             // retrieved.
             final int nameCol = rs.getMetaData().getColumnCount()
  > 1 ?
                  (valCol == 1 ? 2 : 1) : 0;
              final boolean namePresent = nameCol > 0;
              final List resultList = new ArrayList(32);
              while(rs.next()) {
                  final SearchResult sr = new SearchResult();
                  final Object name = namePresent ?
  rs.getObject(nameCol) :
                 null;
                 sr.setName(name == null ? null : name.toString());
                  final String value = rs.getString(valCol);
                  sr.setValue(value == null ? null :
  value.toString());
                  resultList.add(sr);
              }//end while
              return(new Response(resultList));
          } catch (SQLException e) {
              final String msg = "Exception retrieving SQL: " +
  e.getMessage();
              LOGGER.fatal(msg, e);
              throw new TAAttributeDefinitionException(msg);
          } finally {
              DBTool.close(rs);
              DBTool.close(ps);
              DBTool.close(connection);
          }
          //end getDisplayName
```

```
    }

    /**
     * Create the SQL Where clause based on the Filters.
     * @param filters The Filters to create the Where clause on.
     * @param anded Whether the filters should be Anded or OR'd
together.
     * @return SQL Where Clause.
     */
    protected StringBuffer makeWhereClause(final Collection
filters,
     final boolean anded) {
        final StringBuffer whereClause = new StringBuffer(1024);
        final StringBuffer inSegment = new
StringBuffer(filters.size() * 3);
        for (Iterator it = filters.iterator();
it.hasNext();it.next()) {
            inSegment.append("?,");
        }//end for
        final int length = inSegment.length();
        if (length > 0){
            inSegment.deleteCharAt(length - 1);
            inSegment.append(')');
            if (this.query.toUpperCase().indexOf("WHERE") < 0) {
                whereClause.append(" WHERE ");
            } else {
                whereClause.append(" AND ");
            }
            whereClause.append(this.valuefield);
            whereClause.append(" IN (");
            whereClause.append(inSegment);
        }
        return(whereClause);
        //end makeWhereClause
    }

    /**
     * @see
com.hp.ovsi.attribute.SIConstraintIntf#setParams(java.util.Map)
     */
    public void setParams(final Map args) throws
     TAAttributeDefinitionException{
        this.poolname = ( String ) args.get( PARAM_POOLNAME );
        this.query = ( String ) args.get( PARAM_SQLSTRING );
        this.valuefield = ( String ) args.get( PARAM_VALUEFIELD
);

        checkParam( this.poolname, PARAM_POOLNAME );
        checkParam( this.query, PARAM_SQLSTRING );
        checkParam( this.valuefield, PARAM_VALUEFIELD );
        //end setParams
    }

    /**
     * @see
com.hp.ovsi.attribute.SIConstraintIntf#size(java.lang.String)
     */
    public int size(final String attrName) throws
    TAAttributeDefinitionException {
        return(size(attrName, null));
    }

    /**
     * @see com.hp.ovsi.attribute.SIConstraintIntf#size(String,
TAFilter)
     */
    public int size(final String attrName, final TAFilter filter)
throws
    TAAttributeDefinitionException {
        if ( LOGGER.isTraceEnabled() ) {
            LOGGER.trace( "Trying to count # of values for
attribute:" +
            attrName );
        }

        final Connection connection = getConnection();
        PreparedStatement ps = null;
        ResultSet rs = null;
        try {
            ps =
connection.prepareStatement(getSizeQuery(filter));
            if (filter != null) {
                ps.setString( 1, filter.getValue() );
            }//end if
            rs = ps.executeQuery();
            if (rs.next()) {
                return(rs.getInt(1)); //Only one column, the
count, returned
            } //else
            throw new TAAttributeDefinitionException("Count not
            retrieved.");
        } catch (SQLException e) {
            final String msg = "Exception retrieving Row Count
from the
            Database: " + e.getMessage();
            LOGGER.fatal(msg, e);
            throw new TAAttributeDefinitionException(msg);
        } finally {
            DBTool.close(rs);
            DBTool.close(ps);
            DBTool.close(connection);
        } //end finally
        //end size
    }

    /**
     * Create the Query to compute the size of the retrieved
result set.
     * @param filter Filter to use to filter the results.
     * @return SQL Query based on the supplied filter.
     */
    protected String getSizeQuery(TAFilter filter) {
        if (this.query == null) {
            throw new IllegalStateException("Query not set.");
        }
        String query = this.query.toUpperCase();
        final int fromIdx = query.indexOf("FROM");
        int toIdx = query.indexOf("ORDER");

        if (fromIdx < 0) {
            throw new IllegalStateException("Illegal Query
Statement Set:
            " + this.query);
        }
        String substr = "";
        if (toIdx > 0){
         substr = this.query.substring(fromIdx, toIdx);
         }
         else {
         substr = this.query.substring(fromIdx);
         }
         query = "SELECT COUNT(*) " + substr ;
        return(getFilteredQuery(query, this.valuefield, filter));
         //end getSizeQuery
    }

    /**
     * @see
com.hp.ovsi.Commandable#execute(com.hp.ovsi.base.Command)
     */
    public Response execute(Command command) throws
    UnsupportedOperationException, CommandException {
         final int commandCode = command.getCommand();
         try {
             if (commandCode ==
GetConstraintsCommand.COMMAND_VALUE) {
             final GetConstraintsCommand gcc =
             (GetConstraintsCommand)command;
             setParams(gcc.getParams());

return(getConstraints((GetConstraintsCommand)command));
             } else if (commandCode ==
SIConstraintIntf.DISPLAY_COMMAND) {
                 try {
                 final GetConstraintsCommand gcc =
                 (GetConstraintsCommand)command;
                 setParams(gcc.getParams());
                     return(getDisplayName(command));
                 } catch (TAAttributeDefinitionException e) {
                    throw new CommandException(e.getMessage(), e);
                 }
              } else if (commandCode ==
SIConstraintIntf.SIZE_COMMAND) {
                 final GetConstraintsCommand gcc =
                 (GetConstraintsCommand)command;
                 setParams(gcc.getParams());
                 final TAQuery query = gcc.getQuery();
```

```
                final List filterList = query == null ? null :
               query.getTaFilterList();
               final TAFilter filter = filterList == null ? null
:
                    filterList.size() > 0 ?
(TAFilter)filterList.get(0) :
                   null;
               final int size = size(gcc.getAttrName(), filter);
                return(new Response(new Integer(size)));
            }
        } catch (TAAttributeDefinitionException e) {
            throw new CommandException(e.getMessage(), e);
        }
        //else
        throw new UnsupportedOperationException("Command not
understood.");
        //end execute
    }

}//end SearchTable
```

The following is called by the class listed above:

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

import com.trulogica.truaccess.attribute.TAValueConstraintIntf;

public class SearchResult implements
TAValueConstraintIntf.TAValueConstraintBeanIntf {
    private String name;
    private Object value;

    public SearchResult() {
        //Default Constructor }

    public String getName() {
        return this.name; }

    public void setName( String name ) {
        this.name = name; }

    public Object getValue() {
        return this.value; }
    public void setValue( Object value ) {
        this.value = value; }
}
```

Attribute Value Validation External Call

This class determines whether the value is an alphanumeric.

```
package com.trulogica.truaccess.externalcall.validation;
import com.trulogica.truaccess.util.logging.misc.Logger;
import com.trulogica.truaccess.util.logging.misc.Level;
import com.trulogica.truaccess.attribute.TAValueValidationIntf;
import com.trulogica.truaccess.attribute.exception.
TAAttributeValueValidationException;
import java.util.*;
public class IsAlphaNumeric implements TAValueValidationIntf
{
  private static final Logger mLogger =
   Logger.getLogger("com.trulogica.truaccess.externalcall.
   validation.IsAlphaNumeric");
  public void validateValue(String attribName,Object value, Map
args)
   throws TAAttributeValueValidationException
  {
   String password = (String ) value;
   if (mLogger.isLoggable(Level.FINE)) mLogger.fine("Entering
the
    IsAlphaNumeric method ");
   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 (mLogger.isLoggable(Level.FINE)) mLogger.fine("The return 
of the
    containsDigit is  " + containsDigit);
   if (mLogger.isLoggable(Level.FINE)) mLogger.fine("The return 
of the
    containsLetter is  " + containsLetter);
   if (containsLetter && containsDigit)
   {
     if (mLogger.isLoggable(Level.FINE)) mLogger.fine("The 
external call
      vaidation is fine. For is digit and alphabets present. ");
   }
   else
   {

     if (mLogger.isLoggable(Level.FINE))
     {
       mLogger.fine("The external call vaidation is NOT fine. For 
is digit
         and alphabets not present. ");
     }
   throw new TAAttributeValueValidationException(
     "The field:" + attribName + " is not alpha numeric. It must 
contain at
      least one numeric and one non-numeric field", attribName);
   }
 }
}
```

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 Action 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 [] availEntilements,
    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.trulogica.truaccess.wfenginesvcs.wfexternalcall.support;

import
com.trulogica.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 the determines whether a user's department (attribute) changed. If the user's department 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.trulogica.truaccess.wfenginesvcs.wfexternalcall.support;

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

import com.trulogica.truaccess.request.model.*;
import com.trulogica.truaccess.util.logging.misc.*;
import com.trulogica.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 TAAttribute 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[] attrbRec1, 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
          //reconcilation 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);
    }
}
```

Retrieving Request Object Data to Set Workflow Variables

The following example provides the `WorkflowRequestCallout` class, which looks for a map called `$AttributeResourceMap` (defined in the `MAP_NAME` variable). The map must have a name-value pair whose name is `FieldName` (defined in the `FIELD_NAME` variable) and whose value is the field to retrieve from the request target (`requestTarget` class). This class also retrieves the value from the request target and stores it in a workflow variable called `WorkflowRequest` (defined in the `WORKFLOW_REQUEST` variable).

```
package com.trulogica.truaccess.workflow.external;

import java.util.ArrayList;
import java.util.Map;
import java.util.HashMap;
import java.util.Set;
import java.util.Iterator;

import
com.trulogica.truaccess.wfengine.wfexternalcall.WfExternalCall;
```

```
import
com.trulogica.truaccess.wfengine.wfexternalcall.AttributeRecord;
import
com.trulogica.truaccess.wfengine.wfexternalcall.ChangeRecord;
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.base.constants.UserAttributeConstants;
import com.trulogica.truaccess.util.logging.misc.Logger;
import com.trulogica.truaccess.util.logging.misc.Level;
import com.trulogica.truaccess.wfengine.client.IWfQuery;
import com.trulogica.truaccess.wfengine.client.IWfDataUpdate;
import com.trulogica.truaccess.transport.protocol.ejb.client.
  StatelessServiceObjectFactory;

public class WorkflowRequestCallout implements WfExternalCall
{

    private static final Logger mLogger =
      Logger.getLogger(WorkflowRequestCallout.class.getName());
    public static final String WORKFLOW_REQUEST =
"WorkflowRequest";
    public static final String FIELD_NAME = "FieldName";
    public static final String MAP_NAME =
"$AttributeResourceMap";

/*
 * Default public constructor.
 */
public WorkflowRequestCallout()
    {
    }

/*
 * Processes the external call request from Select Identity for
the person number.
 *
 * @param stageId                 The stage to process
 * @param requestTarget          Object containing fixed roles,
groups, and
 *                                 entitlements
 * @param availGrp               Groups that can be assigned to
the user
 * @param availRole               Roles that can be assigned to
the user
 * @param availEntitlements      Entitlements to be assigned to
the user
 * @param attrs                 Additional attributes needed by
the call
 *
 * @return  Object containing the person number
 *
 * @exception Throws WFExternalCallException if an error occurs.
 */
public WfExternalCallStatus process(String stageId,
                                    RequestTarget requestTarget,
                                    AttributeRecord [] availGrp,
                                   AttributeRecord [] availRole,
                                    AttributeRecord []
availEntitlements,
                                    Map attrs) throws
WfExternalCallException
    {
        try
        {
            WfExternalCallStatus ecs = new
WfExternalCallStatus(stageId);

            IWfQuery query = (IWfQuery)
               StatelessServiceObjectFactory.create(
IWfQuery.class );
            mLogger.finest("WorkflowRequestCallout:
WF_PARAM_WORKFLOWINSTID
               is " + attrs.get(WF_PARAM_WORKFLOWINSTID));

            Integer workFlowInstanceID = (Integer)
               attrs.get(WF_PARAM_WORKFLOWINSTID);
            mLogger.finest("WorkflowRequestCallout:
workFlowInstanceID is "
               + workFlowInstanceID);
            int instanceId = workFlowInstanceID.intValue();
           mLogger.finest("WorkflowRequestCallout: instanceId is
" +
               instanceId);

           Map varMap = query.getCurrentVariableMap(instanceId);
           mLogger.finest("WorkflowRequestCallout: varMap is " +
varMap);

            HashMap attributeRequestMap = (HashMap)
varMap.get(MAP_NAME);

            if (attributeRequestMap != null)
            {
                String field = (String)
attributeRequestMap.get(FIELD_NAME);

                if (field != null)
                {
                    IWfDataUpdate du =

(IWfDataUpdate)StatelessServiceObjectFactory.
                        create(IWfDataUpdate.class);
                    String value =

(String)requestTarget.getSingleRequestParamStr(field);

                du.setWorkflowVar(instanceId, WORKFLOW_REQUEST,
field);

                    mLogger.finest("WorkflowRequestCallout:
Setting status
                        to approved...");

                    // Set the status to approved

ecs.setStatus(WfExternalCallStatus.STATUS_APPROVED);
                }
            }
            else
            {
                   mLogger.severe("WorkflowRequestCallout:  Setting 
status to
                        rejected...");
                    // Set the status to rejected

ecs.setStatus(WfExternalCallStatus.STATUS_REJECT_TERMINATE);
               }

               return ecs;
          }
          catch(Exception e)
          {
               mLogger.severe("WorkflowRequestCallout:  Exception 
occurred: " +
                   e.getMessage());
               throw new WfExternalCallException( e );
          }
     }
}
```

glossary

Acronyms

A

AC

access control

AD

Adaptive Connector

ACL

access control list

AD Connector

Active Directory Connector

ADK

application development kit

ADO

ActiveX Data Objects

ANSI

America National Standards Institute

APA

auto port aggregation

API

application program interface

ARPA

Advanced Research Projects Agency

ASCII

American Standard Code for Information Interchange

B

BSIM

Business Service Identity Management

D

DBA

database analyst

DLL

dynamic-link library

DNS

domain name system

DHCP

dynamic host configuration protocol

DHTML

dynamic hypertext markup language

DSN

data source name

DSN

digital switched network

DTD

document type definition

E

EJB

enterprise java bean

F

FQDN

fully qualified domain name

FTP

file transfer protocol

G

GIF

graphics interchange format

GUID

globally unique identifier

H

HSRP

Hot Standby Router Protocol

HTTP

hypertext transfer protocol

HTTPS

Hypertext Transfer Protocol Secure

I

IBM

International Business Machines

IP

internet protocol

ISO

International Organization for Standardization

J

J2C

Java 2 connector

J2SDK

Java 2 software developer kit

J2EE

Java 2 enterprise edition

JAR

Java application resource

JCA

Java connection architecture

JDBC

Java database connectivity

JMS

Java messaging services

JNDI

Java naming directory interface

JSP

Java server protocol

JVM

Java virtual machine

K

KB

kilobyte

L

LDAP

lightweight directory access protocol

LDIF

lightweight data interchange format

LLB

local location broker

M

MAPI

messaging application programming interface

MB

megabyte

MHz

megahertz

MSSQL

MicroSoft Structured Query Language

MTA

message transfer agent

OVSI

HP OpenView Select Identity

P

PDF

portable document format

R

RAR

resource adapter archive

RDF

reporting data feeder

RPC

remote procedure call

S

SDK

software developer kit

SHA

secure hash algorithm

SMTP

simple mail transfer protocol

SNMP

simple network management protocol

SOAP

simple object access protocol

SP

service pack

SSL

secure socket layer

SS0

single sign on

SPML

service provisioning markup language

SQL

structured query language

T

TCP / IP

transmission control protocol / internet protocol

U

URI

uniform resource identifier

URL

uniform resource locator

UTF-8

unicode transformation format (eight-bit character conversion)

V

VM

virtual machine

VNC

virtual network computing

W

WAS

web application server

WAR

web application repository

X, Y, Z

XML

extensible markup language

XSD

XML schema definition

XSL

Extensible Style Sheet Language

Terms

A

Access Control List (ACL)

An abstraction that organizes entitlements and controls authorization. An ACL is list of entitlements and users that is associated with a secured object, such as a file, an operation, or an application. In an ACL-based security system, protected objects carry their protection settings in the form of an ACL.

access management

The process of authentication and authorization.

action

When the context is a user action in the user interface, an operation that can be carried out by an OpenView application. Actions are typically performed on managed object. Select manually executed actions through a menu item or tool bar buttons. Actions can also be configured to automatically occur in response to an event, message, or a change in information in the management database.

When the context is based on OVSI policy, an actions is an operation carried out as a result of the activation of a reconciliation policy and the successful evaluation of a rule or conditions within that policy.

See also: capability

activate

To make active or functional

activity

A logical step in a process; A task that may occur when a workflow template is executed (in Workflow Studio). Activities are the core components of workflow templates; they do the work necessary to provision users. An activity can set a property to be used throughout the workflow, track approvals, start a subworkflow, send email, call an external application, and so on.

adapter

Software that allows information interpretation between two or more software products or components.

AD Connector

Active Directory Connector. A type of interface used to connect HP OpenView Select Identity with the applications it serves on servers that communicate using the Active Directory protocol.

admin role

A template that defines the administrative actions performed by a user. Create an Administrative Service to provide access to roles so that users gain access to the Service. Users with administrative roles may grant their set of roles to another administrator within their Service context.

advanced customization

Less common types of customization which are more flexible in their capabilities and complex in their implementation than typical customizations. As with other customizations, advanced customizations are done to meet the needs and preferences of a particular customer or user.

agent

A program or process running on a remote device or computer system that responds to management requests, performs management operations, or sends performance and event notification. An agent can provide access to managed objects and MIB variables, interpret policy for resources and do configuration of resources; The component of an agent-based connector that resides in the same system as the resource. It listens for a changes in the user data made in the resources, then reports that change to HP OpenView Select Identity by communicating through a connector interface.

agent-based connectors

Two-way connector interface. There are two components: the connector that resides in the same system as HP OpenView Select Identity, and the agent, which resides in the same system as the resource. The agent listens for changes made in the resource, and contacts the resource about changes made in Select Identity.

agentless connectors

One way connectors. Connectors reside in the Select Identity server and does the communication brokering with the resource.

application

Packaged software that provides functionality that is designed to accomplish a set of related tasks. An application is generally more complex than a tool.

application deployment

The installation and activation of application components so that they work in the business environment.

Application Program Interface (API)

A set of routines, protocols, and tools used to build a software application; An interface that enables programmatic access to an application.

approval process

The process of approving the association, modification, or revocation of entitlements for an identity. This process is automated of these through workflow templates.

approver

A Select Identity administrator who has been given approval actions through an Admin Role.

assigned policy

A policy that has been assigned to one or more resources in the computing environment but which has not yet been deployed or installed on those resources.

asynchronous subprocess

A process that proceeds at its own pace independent of other processes and subprocesses.

attribute

An individual field that helps define an identity profile. For each identity, an attribute has a corresponding value. For example, an attribute could be “department” with possible values of “IT,” “sales,” or “support.”

attribute external call

Small programs that are written to generate values automatically for that attribute (value generation), define constraint values for the attribute (value constraint), or validate the value that is entered for that attribute (data validation). Each attribute can have each of these types of external calls.

attribute name-value pair

An attribute name-value pair is combination of an attribute identifier and the value of that attribute for a specific object. An example of an attribute-name-value pair for a person would be Name: John Smith.

audit engine

logs and stores all audit-related activities, e.g., when changes are made and who made them.

Audit Report

A report that provides regular account interaction information.

authentication

Verification of an identity's credentials.

authoritative source

A resource that has been designated as the "authority" for identity information. Select Identity accounts can be reconciled against accounts in an authoritative source.

automatic action

A pre-configured program or script that is executed in response to an event, message, or a change in information in the management database. without operator intervention.

B

bandwidth

The transmission capacity of an electronic line such as a communications network, computer bus, or computer channel. It is expressed in bits per second (for example, 56 kbps), bytes per second or in Hertz (cycles per second). When expressed in Hertz, the frequency may be a greater number than the actual bits per second, because the bandwidth is the difference between the lowest and highest frequencies transmitted. (TECH).

block

A special type of activity that serves two purposes: to define information to be used by a subset of activities (block-level properties) and to provide block-level reporting. For example, you might define a block that submits an approval request, waits for the response, and returns the status of the request to the workflow. In other words, think of a block as a process within a template.

block type

A property assigned to a block in a workflow template using the blockType property in end block activity. The report template uses this property to identify how block information is rendered in the resulting report.

Boolean operator

A logical operator that defines the context in which attribute values are compared to satisfy a query or policy. For example:

AND - Both conditions have to be satisfied.

OR - At least one condition has to be satisfied.

NOT - No instance of this condition is allowed.

browser

A module within a work space that presents one or more views of objects and provides functionality for interacting with the objects and the views.

business service

A product or facility offered by, or a core process used by, a business in support of its day-to-day operations. Example business services could include an online banking service, the customer support process, and IT infrastructure services such as email, calendaring, and network access.

See also: service

Business Service Identity Management (BSIM)

An organizational model that introduces new abstractions that simplify and provide scale to the business processes associated with identity management. These abstractions are modeled after elements that exist in businesses today and include Services and Service Roles.

C

capability

Actions that can be performed within the HP OpenView Select Identity client.

See also: action

challenge and response

A method of supplying alternate authentication credentials, typically used when a password is forgotten. Select Identity challenges the end user with a question and the user must provide a correct response. If the user answers the question correctly, HP OpenView Select Identity resets the password to a random value and sends email to the user. The challenge question can be configured by the administrator. The valid response is stored for each user with the user's profile and can be updated by an authenticated user through the Self Service pages.

client

When the context is network systems, a computer system on a network that accesses a service from another computer (server).

When the context is software, a program or executable process that requests a service from a server

client console

An instance of the user interface that appears on the client system while the application runs on a server.

condition type

An abstraction or categorization of a condition that determines to the particular kind of data that is valid for the parameter values in the condition and how those values will be used.For example a condition type could be Source IP Address which indicates that values must have 4 numbers separated by decimals with the value for each number being in the range of 0 to 255. Since the condition type is "Source" IP Address, the IP addresses will only be evaluated for sources not destinations.

configuration file

A file that contains specifications or information that can be used for determining how a software program should look and operate.

configuration

In a hardware context, a particular set of inter-related components that make up a computer system. For example the components of a computer system may include a keyboard, pointing device, memory, disk drives, modem, operating system, applications and printer. The configuration of the computer system determines the way that it works and the way that it is used.

In a network context, the complete set of inter-related systems, devices and programs that make up the network. For example the components of a network may include computer systems, routers, switches, hubs, operating systems and network software. The configuration of the network determines the way that it works and the way that it is used.

In a software context, the combination of settings of software parameters and attributes that determine the way the software works, the way it is used, and how it appears.

configure

To define and modify specified software settings to fulfill the requirements of a specified environment, application or usage.

Configuration Report

A report that provides current system information for user, administrator, and Service management activities.

connection

A representation of a logical or physical relationship between objects.

connector

A J2EE connector interface that communicates with the system resource applications that contain your identity profile information.

console

An instance of the user interface from which the user can control an application or set of applications.

context

An HP OpenView Select Identity concept that defines a logical grouping of users that can access a Service.

credential

A mechanism or device used to verify the authenticity of an identity. For example, a user ID and password, biometrics, and digital certificates are considered credentials.

customization

The process of designing, constructing or modifying software to meet the needs and preferences of a particular customer or user.

customize

To design, construct or modify software to meet the needs and preferences of a particular customer or user.

customize

To design, construct or modify software to meet the needs and preferences of a particular customer or user.

D

database

A repository of data that is electronically stored. Typically databases are organized so that data can be retrieved and updated.

data file

An SPML file that enables you to define user accounts to be added to Select Identity through Auto Discovery or Reconciliation.

data type

A particular kind of data; for example

deactivate

To deliberately stop a component or object from working.

delegated administration

The ability to securely assign a subset of administrative roles to one or more users for administrative management and distribution of workload. Select Identity enables role delegation through the Self Service pages from one administrator to another user within the same Service context.

delegated registration

Registration performed by an administrator on behalf of an end user.

See also: self-registration

deploy

To install and start software, hardware, capabilities, or services so that they work in the business environment.

deployed application

An application and its components that have been installed and started to work in the business environment.

deployed policy

A policy that is deployed on one or more resources in the computing environment.

deployment

The process of installing and activating software, hardware, capabilities or services so that they work in the business environment.

deployment package

A software package that can be deployed automatically and installed on a managed node.

deprecate

To lower the status of a hardware or software object to indicate that it can be taken out of use in the future

device

A generic term for a piece of hardware equipment that can be attached to a computer or a network. Examples of a device are a printer, a router, a switch, a load-balancer, a disk drive or a modem.

disable

To make unable to be used.

dismiss

Dismiss is an action that causes a message or other notification associated with a problem or situation to be removed from the browser. Messages are typically dismissed when the operator has resolved the situation that led to the message.

disown

The act of relinquishing responsibility for resolving a problem or situation associated with a message or other notification.

DNS domain

A set of computers and other network devices that are collectively addressable by a portion of an IP address or by the highest subdivision of the domain name that indicates the entity owning the address. For example all computers whose host name share the suffix .hp.com are in the same DNS domain.

domain

A set of computers and other network devices that are treated or managed as a unit.

double-click

To press and release a pointing device's button twice in rapid succession. Double-clicking is a time-dependent action. Clicking twice in the same location at slow speed (click-delay-click) is not a double-click.

downtime

The amount or percentage of time that a service, software, or hardware resource remains non-functional.

dynamic parameters

Parameters whose values are determined during program execution.

E

enable

To make able to use.

end user

A role associated to every user in the Select Identity system that enables access to the Self Service pages.

entitlement

An abstraction of the resource privileges granted to an identity. Entitlements are resource-specific and can be resource account IDs, resource role memberships, resource group memberships, and resource access rights and privileges. Entitlements are also considered privileges, permissions, or access rights.

event

An event is an unsolicited notification such as an SNMP trap or WMI notification generated by an agent or process in a managed object or by a user action. Events usually indicate a change in the state of a managed object or cause an action to occur.

event attribute

A characteristic or property of an event.

event correlation

The evaluation of multiple events or notifications that are related to a single incident or problem, to produce a single message. Event correlation is used to reduce the number of messages that are presented to an operator in a message browser.

event creation time

The time an event was created in Universal Coordinated Time (UTC)

event syntax

The rules governing the structure and content of an event.

event type

A classification of an event into a particular category that further defines the nature of the event.

export

To format and move information from the current application to a location outside the current application.

expression

A combination of workflow variables and constant values to be evaluated. An expression can be assigned to a new variable or passed to an application as an argument. If you are familiar with a programming language, an expression used in a workflow template is like C or Java expression. Example of expressions can be found in action input parameters, application return values, and transition conditions.

extend

The act of increasing the capabilities, scope, or effectiveness of a program.

extensible

Capable of being extended.

external call

A programmatic call to a third-party application or system for the purpose of validating accounts or constraining attribute values.

external system ID

An identifier that uniquely identifies a principal that is an external system.

F

filter

A software feature or program that functions to screen data so that only a subset of the data is presented or passed. Filters allow matching-relevant information to be extracted and acted on while non-matching-irrelevant information is held back.

find

The act of seeking of specific data or objects within the management application or set management applications based on specified criteria.

form

An electronic document used to capture information from end users. Forms are used by Select Identity in many business processes for information capture and system operation; A presentation mechanism that contains information and controls for obtaining user input (for example, text fields, radio buttons, lists).

foundation

A program that acts as the basic structure to support other software modules or programs that provide additional functionality for the user.

function

A general term for a portion of a program that performs a specific task.

H

hierarchy

Elements organized in successive levels with each lower level being subordinate to the one above.

HP OpenView

A family of network and system management products, and an architecture for those products. HP OpenView includes development environments and a wide variety of management applications.

I

icon

An on-screen image that represents objects that can be monitored or manipulated by the user or actions that can be executed by the user.

icon class

The portion of an icon that identifies the type or classification of the object being represented by the icon. For example, the network object class is represented by a circle surrounding a more complex image.

ID

identifier

identifier

A name that within a given scope that uniquely identifies the object with which it is associated.

identity

The set of authentication credentials, profile information, and entitlements for a single user or system entity. Identity is often used as a synonym for “user,” although an identity can represent a system and not necessarily a person.

identity management

The set of processes and technologies involved in creating, modifying, deleting, organizing, and auditing identities.

import

To format and move information from a location outside the current application into the current application.

install

To load a product or component of a product onto a computer system or other network or system device. Installation typically involves running initial configuration scripts that are part of the installation process.

instance

See: workflow instance

internationalization

The design of software so that a single binary can support the varied cultural and linguistic conventions that exist in different countries or locales. Internationalized software allows users to interact with the software in the user's native language including the input and output of data in the native language, as well as support for the conventions and rules applicable to the user's locale. The ANSI locale model is used in internationalized software.

J

Java

Object oriented programming language.

JCA

Java Connection Architecture. Architecture used to build interfaces between J2EE compliant products and other resources.

JVM

Java Virtual Machine. A platform independent execution environment that conversant Java bytecore into machine language then executes it.

L

LDIF

File that modifies and deletes directory objects.

list

If the context is a GUI, a set of selectable items. If the context is data, a variable-length ordered set of values all of the same data type.

locale

The locale collectively represents the location or country of the user, the language of the user, and the code set in which the user's data is represented. The locale is related to the language sensitive presentation of applications.

locale model

The software through which the user declares their desired language at application start up. The local model determines the set of files, tables, or collection of programs that are used to initialize an application so that it is sensitive to the user's language.

localization

Localization refers to the set of tasks that need to be accomplished to enable a product to work acceptably in a specific locale. The localization tasks include translating documentation, translating text and graphics that are presented to the user, and providing locale specific fonts and other functionality when needed.

M

management

The ongoing maintenance of an object or set of objects, including creating, modifying, deleting, organizing, auditing, and reporting.

message key

A message attribute that is a string used to identify messages that were triggered from particular events.The string summarizes the important characteristics of the event. Message keys can be used to allow messages to acknowledge other messages, and allows for the identification of duplicate messages.

N

node

When the context is network, a computer system or device (for example, printer, router, bridge) in a network.

When the context is a graphical point to point layout, a graphical element in a drawing that acts as a junction or connection point for other graphical elements.

notifications

The capability that enables you to create and manage templates that define the messages that are sent when a system event occurs.

P

package

A set of related programs or software files grouped together as a single object for a common purpose.

password reset

The ability to set a password to a system-generated value. Select Identity uses a challenge and response method to authenticate the user and then allow the user to reset or change a password.

persistent variable

A variable that is persisted after an instance is passivated. To extend the variable life cycle to the entire instance, you must create the variable to be persistent. This enables the variable to be created before a wait activity, and it will be accessible after the workflow instance resumes. To make a variable persistent, precede the name with $. For example, the $retryCount variable is persistent while retryCount is not.

See also: workflow variable

policy

A set of regulations set by an organization to assist in managing some aspect of its business. For example, policy may determine the type of internal and external information resources that employees can access.

policy management

The process of controlling policies (for example, creating, editing, tracking, deploying, deleting) for the purposes of network, system or service management.

port

If the context is hardware, a location for passing information into and out of a network device.

process

A repeatable procedure used to perform a set of tasks or achieve some objective. Whether manual or automated, all processes require input and generate output. A process can be as simple as a single task or as complicated a multi-step, conditional procedure.

See also: approval process

profile

Descriptive attributes associated with an identity, such as name, address, title, company, or cost center.

property

See:workflow property

provisioning

The process of assigning authentication credentials to identities.

R

reconciliation

The process by which Select Identity accounts are synchronized with a system resource. Accounts can be added to the Select Identity system through the use of an SPML data file.

registration

The process of requesting access to one or more resources. Registration is generally performed by an end user seeking resource access, or by an administrator registering a user on a user's behalf.

See also: delegated registration, self-registration

request

An event within the Select Identity system for the addition, modification, or removal of a user account. Requests are monitored through the Request Status capability.

resource

Any single application, database, or information repository. Resources typically include applications, directories, and databases that store identity information.

role

A simple abstraction that associates entitlements with identities. A role is an aggregation of entitlements and users, typically organized by job function.

See also: admin role

rule

A programmatic control over system behavior. Rules in Select Identity are typically used for programmatic assignment of Services. Rules can also be used to detect changes in system resources.

S

self-registration

Registration performed by an end user seeking access to one or more resources.

See also: deploy

self service

The ability to securely allow end users to manage aspects of a system on their own behalf. Select Identity provides the following self-service capabilities: registration, profile management, and password management (including password change, reset, and synchronization).

service

A business-centric abstraction representing resources, entitlements, and other identity-related entities. Services represent the products and services that you offer to customers and partners.

service attribute

A set of attributes and values that are available for or required by a Service. Attributes are created and managed through the Attributes pages.

See also: attribute

service role

A Select Identity abstraction that defines how a logical grouping of users will access a Select Identity Service. The Select Identity Service is a superset of all the identity management elements of a business service.

service view

A restricted view of a Service that is valid for a group of users. Views enable you to define a subset of Service registration fields, change field names, reorder fields, and mask field values for specific users.

single sign-On (SSO)

A session/authentication process that permits a user to enter one set of credentials (name and password) in order to access multiple applications. A Web SSO is a specialized SSO system for web applications.

SPML Data File

See: data file

submodule

A portion of a software module that provides a subset of the functionality provided by the module. A sub-module performs a specific task or presents a specific set of data.

suspend

To halt for a time a computer operation preserving the state of that operation.

synchronous subprocess

A process that must complete before the invoking process can proceed.

syntax

The rules governing the structure and content of a language or the description of an object.

system administrator

The role of a person who does configuration and maintenance on a computer system or the software on the system.

T

template

See: workflow template

trace log

An output file containing records of the execution of application software

transit delay

The difference between current time and the event's creation time.

transition

The definition of a relationship between activities. You can define that one activity always follows another, or you can define a condition that must be met before the workflow transitions from an activity to one or more others. For example, you can define a transition that only allows the workflow to progress if at least two administrators approve a request. If the request is not approved, the workflow can transition to an activity that sends email notification to an administrator.

U

URL

Acronym for Uniform Resource Locator or Universal Resource Locator, the address of a computer or a document on the Internet.

user import

The process of adding user accounts to the Select Identity system for a specified Service through the use of a data file.

users

The functionality that provides consistent account creation and management across Services.

V

variable

See: workflow variable

variable expression

See: external call

W

Web Service Definition Language (WSDL)

File format that the Application Definition file uses to define a web service application to be a workflow application. The workflow engine reads the web service invocation parameters through WSDL. A web service can reference a WSDL URL remotely or download it first as a local file and then read the file locally at run-time.

workflow engine

A system component that executes workflows and advances them through their flow steps.

workflow external call

A “subroutine” that is called during the workflow process. This could be an external application invocation such as a small custom application that calls external processes outside of the normal workflow process

workflow instance

An invocation of a workflow template. An instance starts when it is created and ends when it completes (when the last activity is executed). An instance’s status and other associated information can be viewed once an instance is created.

workflow process

The tasks, procedural steps, organizations or people involved, and required input and output information needed for each step in a business process. In identity management, the most common workflows are for provisioning and approval processes.

workflow property

A name-value pair, where the value is a text string. A property stores static data that cannot be changed at runtime. It can be accessed by the workflow API and report template. There are three levels of properties: global, block, and activity.

workflow studio

The functionality that enables you to create and manage workflow templates.

workflow template

A model of the provisioning process that enables Select Identity to automate the actions that approvers and systems management software must perform.

workflow variable

A name-value pair that can be created or changed at runtime in a workflow instance through actions, a workflow API call, or returned by an application invocation. It can be accessed by workflow API, workflow template, and report template. There are levels of variables: global, block, and activity.

See also: persistent variable

Index