Sun Java™ System

Identity Manager 6.0 Technical Deployment Overview

2005Q4M3

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 819-4486-10

Contents

LDAP Deployment Scenario

Preface

This guide provides an overview of the reference and procedural information you will use to customize Sun Java™ System Identity Manager for your environment.

How to Find Information in this Guide

The guide is organized in these sections:

- Chapter 1. Working with Attributes — Introduces Identity attributes and how to use this feature to streamline the data flow through your Identity Manager deployment.
- Chapter 2. Data Loading and Synchronization — Presents an overview of the reconciliation and other mechanisms for loading account information into Identity Manager. Reconciliation compares the set of users defined in Identity Manager to the set of accounts that are defined on an Identity Manager resource.
- Chapter 3. Data Loading Scenarios — Provides tips to consider when preparing to load account information into Identity Manager, including sample scenarios that illustrate some of the issues that you might encounter.
- Chapter 4. Configuring User Actions — Details how to add custom tasks to the Identity Manager Administrator Interface and configure user actions that you can execute from two areas of the interface.
- Chapter 5. Private Labeling of Identity Manager — Describes how to customize IDM colors, logos, and header and footer content to meet the style standards of your organization.
- Appendix A. Editing Configuration Objects — Provides an overview of configuration objects and a discussion of the UserUIConfig object.
- Appendix B. Enabling Internationalization — Provides information on configuring Identity Manager to use multiple languages or display a language other than English.
- Appendix C. LDAP Scenario — Describes an LDAP deployment scenario, which may be loaded into Identity Manager from scripts in the `sample` directory.

Related Documentation and Help

Sun Microsystems provides additional printed and online documentation and information to help you install, use, and configure Identity Manager:

- *Identity Manager Installation*

  Step-by-step instructions and reference information to help you install and configure Identity Manager and associated software.

- *Identity Manager Upgrade*

  Step-by-step instructions and reference information to help you upgrade and configure Identity Manager and associated software.

- *Identity Manager Administration*

  Procedures, tutorials, and examples that describe how to use Identity Manager to provide secure user access to your enterprise information systems.

- *Identity Manager Technical Deployment Overview*

  Conceptual overview of the Identity Manager product (including object architectures) with an introduction to basic product components.

- *Identity Manager Workflows, Forms, and Views*

  Reference and procedural information that describes how to use the Identity Manager workflows, forms, and views — including information about the tools you need to customize these objects.

- *Identity Manager Deployment Tools*

  Reference and procedural information that describes how to use different Identity Manager deployment tools; including rules and rules libraries, common tasks and processes, dictionary support, and the SOAP-based Web service interface provided by the Identity Manager server.

- *Identity Manager Technical Reference*

  Reference and procedural information that describes how to load and synchronize account information from a resource into Sun Java™ System Identity Manager.

- *Identity Manager Audit Logging*

  Reference and procedural information that describes how to load and synchronize account information from a resource into Sun Java™ System Identity Manager.

- *Identity Manager Tuning, Troubleshooting, and Error Messages*

  Reference and procedural information that describes Identity Manager error messages and exceptions, and provide instructions for tracing and troubleshooting problems you might encounter as you work.

- *Identity Manager Help*

  Online guidance and information that offer complete procedural, reference, and terminology information about Identity Manager. You can access help by clicking the Help link from the Identity Manager menu bar. Guidance (field-specific information) is available on key fields.

Product Support

If you have problems with Identity Manager, contact customer support using one of the following mechanisms:

- The online support web site at http://www.sun.com/service/online/us
- The telephone dispatch number associated with your maintenance contract

We'd Like to Hear from You!

We would like to know what you think of this guide and other documentation provided with Identity Manager. If you have feedback - positive or negative - about your experiences using this product and documentation, please send us a note:

Sun Microsystems.
5300 Riata Park Court
Austin, TX 78727
Attn: Identity Manager Information Development

Email: idm-idd@sun.com

We'd Like to Hear from You!

1 Working with Attributes

Attributes are name-value pairs that are used to define and manipulate characteristics of Identity Manager objects as well as external resources. Identity Manager components such as forms, workflows, and rules call attributes as an essential part of accessing and transforming data of their regular operations.

Related Information

Attributes are manipulated as part of many Identity Manager operations and are discussed throughout the documentation set. Substantial discussion of various attributes types are included in these chapters:

- *Views* chapter of *Identity Manager Workflows, Forms, and Views* for an extensive discussion of view attributes, registering attributes, and deferred attributes.
- *Identity Manager Resources Reference* for information about resource attributes

Types of Attributes

Although objects in an Identity Manager deployment can contain a variety of attributes, you are most likely to customize or work with the types introduced in the following table. Each attribute type is introduced at greater length in subsequent sections.

Attribute Type	Description
Summary attributes	Define attributes available when writing objects
Queryable attributes	Define attributes that are accessible for searches
Inline attributes	Optimize queries through the database.
User Extended attributes	Define optional name/value pairs that are not included with default object definitions but that are added by users
View attributes	Define name/value pairs of user account information that are assembled into a dynamic data model called a view.
Operational attributes	Required for the repository to work correctly

Attribute Type	Description
Resource User attributes	Define a characteristic of a user account on a resource.
Identity System User attributes	Define an Identity Manager value that corresponds to a Resource user attribute. Identity Manager displays Identity system user attributes on the left side of the schema map. This term is used interchangeably with account attributes.
Identity attributes	Provide a central point from which you can configure and view attribute flow within your Identity Manager deployment.

Summary Attributes

Every persistent object exposes a set of *summary attributes*. Each subclass can extend the default set by overriding the getSummaryAttributes method. Summary attributes contain the values that are returned for each item in the result of a list call.

Summary attributes are typically single-valued, because there is a limit to the total length of the summary attributes when serialized to a string.

You configure these attributes directly through the UserUIConfig object. For more information, see *UserUIConfig Object*.

Queryable Attributes

Every persistent object exposes a set of queryable attributes. Each subclass can extend the default set by overriding the getQueryableAttributes method.

Queryable attributes can be multi-valued, and contain the set of values used for filtering and matching.

You configure these attributes directly through the UserUIConfig object. For more information, see *UserUIConfig Object*.

Inline Attributes

You can designate up to five queryable attributes for each type as *inline attributes*. Designating an attribute as *inline* asks the data store to optimize the performance of queries against that attribute.

Identity Manager typically stores each value of a queryable attribute as a row in an attribute table that is separate from the main object table. The attribute table can be joined to the object table to select objects that match an AttributeCondition.

Identity Manager stores the value of an *inline* attribute, however, directly in the object table for that type. Designating an attribute as inline allows Identity Manager to generate more efficient SQL. A column expression on the main object table is faster than a JOIN to (or an EXISTS predicate against) the corresponding attribute table. This improves the performance of any query against the attribute.

- *An inline attribute must be single-valued* because its value is stored in a single column of the parent row in the object table.
- *Up to five queryable attributes can be inline for a type* because the object table contains only five columns that can be used to store arbitrary attribute values.
- *The same set of queryable attributes is designated as inline for every instance of a type* because the correspondence between the column value and the name of an attribute is specified only by the configuration. That is, the configuration of inline attributes for a type is the only way the repository knows which attribute is stored in which column.

User Extended Attributes

You can add attributes to the default list of extended attributes for a user object. These attributes, called *User Extended attributes*, are stored on the `User Extended Attributes` Configuration object in the repository.

This Configuration object defines additional attributes that are stored in the user objects in the repository. The system ignores any attribute value in the `accounts[lighthouse]` namespace of the User view that is not registered in the configuration object.

See the discussion of the accounts[lighthouse] attribute of the User view in the *Views* chapter of *Identity Manager Workflows, Forms, and Views* for more information on user extended attributes.

Operational Attributes

Identity Manager predefines several attributes that are required for the repository to work correctly. `ID`, `type`, and `name` are especially important.

ID, type and name

Every PersistentObject stored in the repository has a *globally unique internal identifier (ID)*. An ID value is unique across time and space, and a generated ID value is never re-used. (Some predefined Identity Manager objects have well known identifiers that are defined as program constants. These are known as *fake IDs*.) The repository promises that an object's ID will never change.

Objects of the same type typically *map* to the same Java class. That is, they are constructed as instances of the same Java class when *deserialized*. Where there is not a one-to-one correspondence between type and Java class, every object of the same type at least uses the same mechanism to look up the corresponding Java class. (For example, some types of objects expose a `class` attribute that contains the fully qualified class name).

An object's *name* must be unique within *type*. That is, only one object of a type can have a particular name. (However, another object of a different type can have the same name). Thus, each type effectively defines a subordinate namespace. You can change an object's name, but you cannot change an object's ID.

modified

A Java long, `modified` contains a value that is incremented whenever the object is modified through Identity Manager.

Immediately after the update, and until the next object of the same type is modified, this `modified` value will match the value of the `counter` attribute in the special LASTMODIFIED item for that type.

lockinfo

A Java String, `lockinfo` contains a value that represents a logical lock.

The `lockinfo` string includes the name of the locker and the time that the logical lock will expire.

counter

A Java long, counter contains an arbitrary value that can be incremented or set very efficiently *without requiring a logical lock on the object*.

Other Standard Attributes

MemberObjectGroups

Every persistent object belongs to at least one object group. Each value of this multi-valued attribute is the ID of an ObjectGroup object.

ObjectGroups are exposed as Organizations in the Identity Manager Administrator and User interfaces. ObjectGroup membership governs Session-level authorization (that is, administrator and end-user access to repository objects), but the repository itself ignores object group membership.

creator, createDate, lastModifier and lastModDate

These values record historical information about each object. These attributes are maintained (but are not used) by the repository.

PropertyList

Every persistent object can contain an arbitrary list of Properties. This feature is not widely used.

subType

Every persistent object can have a `subType` attribute. For example, Identity Manager uses Attribute.SUBTYPE to select separate lists of the available correlation rules and confirmation rules.

authType

The `authType` attribute allows authorization to be performed (that is, access to be scoped or restricted) for users who do not control any organization (object group). These subjects would otherwise have no access in Identity Manager's standard authorization scheme.

View Attributes

A *view* is a collection of name/value pairs that are assembled from one or more objects stored in the repository, or read from resources. The value of a view attribute can be atomic such as a string, a collection such as a list, or reference to another object.

Whenever you create or modify a user account from the Identity Manager Administrator or User interfaces, you are indirectly working with the User view. Workflow processes also interact with the User view. When a request is passed to a workflow process, the attributes are sent to the process as a view. When a manual process is requested during a workflow process, the attributes in the user view can be displayed and modified further.

Views are extensively documented in the *Views* chapter of *Identity Manager Workflows, Forms, and Views*.

Resource User Attributes

Resource User attributes (sometimes referred to as *account attributes*) map Identity Manager account attributes to resource account attributes in a schema map. The list of attributes varies for each resource. You can remove unused attributes from the schema map page. However, adding attributes might require editing the adapter code.

The Resource User attributes are used only when the adapter communicates with the resource.

Working with Resource User attributes are extensively documented in *Identity Manager Resources Reference*.

Identity System User Attributes

Identity System User attributes define an internal Identity Manager value that corresponds to a Resource user attribute. The Identity System User attributes can be used in rules, forms, and other Identity Manager-specific functions. Identity Manager displays these attributes on the left side of the schema map. This term is used interchangeably with account attributes.

Working with Identity System User attributes are extensively documented in *Identity Manager Resources Reference*.

AttributeCondition

An *attribute condition* is an expression that tests the value(s) of an attribute. Attribute conditions are typically used to select the subset of objects that match certain criteria.

Each attribute condition expresses a single criterion and consists of:

- an attribute name (of a queryable attribute)
- an operator (a type of check or comparison to be made)
- operand (a specified set of values)

Working with Identity Attributes

Identity attributes provide a central point from which you can configure and view attribute flow within your Identity Manager deployment, regardless of where the information originates from, or how many resources — for example, SPML, Active Sync resources, the User Interface.

For specific deployment needs, Identity attributes can provide an alternative to customizing workflows and forms. For example, you can create an Identity attribute named `fullname` that results from appending `firstname` to `fullname` instead of creating a workflow to derive the same value.

Identity Manager permits you to select which portions of your deployment environment will incorporate Identity attributes. You can use Identity attributes throughout your deployment or limit their implementation to “applications”, including the Identity Manager Administrator Interface, Identity Manager User Interface, or during Active Sync processing, for example.

You define Identity attributes from the **Configure > Identity Attributes** page in the Identity Manager Administrator Interface. Consult the online help that accompanies these pages for procedures on creating, deleting, and editing attributes.

Why Use Identity Attributes?

Identity attributes provide the following advantages for your deployment:

- consistent way to apply data mapping rules within a deployment
- separation of data-transformation functions of a form from its data-display capabilities. If you’ve worked with an Identity Manager deployment previously, you’ll know that Identity Manager forms serve the dual purpose of transforming data and displaying data. Using Identity attributes reduces the amount of form-based data transformations.
- centralized view of how data flows through an Identity Manager deployment

Architectural Overview

Implementing Identity attributes involves the following Identity Manager components. The more significant components are described below.

Component	Description
attribute source	Attribute sources can be resources, rules, transformations, other Identity attributes, or constants.
attribute target	Attribute targets can be resources, extended attributes, or operational Identity user attributes.
Meta View	Object that supports the creation and manipulation of Identity attributes. This view namespace exists within the User view. You can access this object, as you can other objects, with the Business Process Editor.
User Metaview form	Form that Identity Manager automatically generates after configuring or modifying Identity attributes.
UserUIConfig object	Object that stores information about the Identity Manager summary, queryable, and extended user attributes. For example, if you choose to store a newly created Identity attribute in the repository, Identity Manager adds the attribute to the *Extended User Attributes* Configuration object in the repository.

Attribute Sources and Targets

Each Identity attribute can use input from one or more source. Inputs can include:

- *resource attributes*. For example, you can assign an authoritative Active Sync resource to populate `firstname` on other resources.
- *transformations*. Attributes can be derived from other Identity attributes (or user view attributes) through a rule or transformation. Meta view transformations are applied uniformly throughout the system, regardless of where a change originates. Identity Manager applies these transformations whether an attribute was derived from an Active Sync event, a reconciliation process, or a user entering information into a form in the User Interface.
- *constant values*. Identity attributes can be assigned constant values to provide a default value for attributes that are not available or cannot be otherwise derived (for example, `password`).

An *Identity attribute target* can be a resource or extended attribute, or operational attribute. An *operational attribute* is an attribute on the Identity user that controls the behavior of that user (for example, rules or assigned resources). When the output is a resource, Identity Manager writes the attribute to the resource when the attribute is changed.

You can also assign conditions under which the attribute should be pushed to a target resource. These conditions are rules that should return a value of true or false, and can also be restricted by event type (for example, create and update).

Meta View

The Meta view supports the creation and manipulation of Identity attributes. It provides a unified view of all resources in a deployment and their relationships.

Whenever you define an Identity attribute, Identity Manager creates an attribute in the `metaView` namespace in the User view that contains the value of this attribute, as illustrated below.

Meta View Attributes	User View Attributes
`firstname`	`metaView.firstname`
`lastname`	`metaView.lastname`
`waveset.roles`	`metaView.waveset.roles`

Related Forms

Identity Manager automatically generates the User MetaView form after you have configured or modified Identity attributes. It creates this form when an enabled application first accesses Identity attributes. For example, if you have designated the Identity Manager Administrator Interface as an enable application, Identity Manager creates this form when a user first accesses the Edit/Create user page.

Do not modify this form.

How Identity Attribute Transformations are Applied

Identity Attribute transformations are applied to the User view by including the User MetaView Form in the standard view cycling process using expansion, derivation, and default expressions. This cycle occurs when the User view is created (for a new user), checked out (for an existing user), refreshed, or checked in. The User MetaView Form is processed immediately before the standard user form for the application (for

example, the Tabbed User Form). This allows the standard user form to override values set through the Identity attributes, and gives the standard user form access to values in the metaView namespace.

Note that Active Sync and SPML use a slightly different view processing cycle since they operate with multiple forms, and therefore do not allow the input forms to override values set from Identity Attributes.

Sample Scenario

The following table exemplifies the relationship between Identity attributes and the resources. In this table, the `firstname` Identity attribute is set by a PeopleSoft resource. Identity Manager uses the `firstname` and `lastname` attributes as source for the rule that generates for the `fullname` and `user ID` Identity attributes.

Identity Attribute	Attribute Source	Attribute Target
`firstname`	PeopleSoft	• Active Directory • AIX
`lastname`	PeopleSoft	• Active Directory • AIX
`fullname`	• firstname (rule-generated first space last) • lastname (rule-generated first initial last)	Active Directory
`user ID`	• firstname • lastname	• Active Directory • AIX
`description`	Active Directory	Identity Manager user (stored locally)

Storage of Attribute Values

Identity attribute values can be stored either locally or retrieved dynamically during runtime. Identity Manager configures *locally stored attributes* as User Extended attributes. When you add an attribute to the Extended User Attributes list, Identity Manager stores and maintains the attribute with the user object in the repository.

Storing attribute values in this way provides two benefits:

- Meta view transformations are processed more quickly
- Identity Manager can perform searches across resources

Typically, an Identity Manager deployment retrieves attributes on-the-fly when it loads the User view.

Using Rules with Identity Attributes

You can use a rule to generate the value of an Identity attribute. When using rules this way, keep in mind:

- Rules typically have access to the entire User view, including the MetaView namespace. Access includes the `activeSync` namespace, if the event is derived from an Active Sync resource.
- Rules used for Meta view transformations should act only on Meta view attributes. Avoid modifying User view attributes.
- Use rules only as sources for Identity attributes

2 Data Loading and Synchronization

This chapter presents an overview of the techniques that can be used to load and synchronize account information from a resource into Identity Manager. See also *Identity Manager Scenarios* for examples that illustrate how accounts can be loaded and linked to Identity system user accounts.

It is important to clearly distinguish between Identity system users and resource accounts. The following definitions make it easier to understand the topic:

- **User** — A virtual identity that is managed by Identity Manager. A Identity Manager user may refer to any number of accounts.
- **Account** — A concrete identity that is managed by a resource (or, more precisely, by an external system or application that is represented as a Resource object in Identity Manager). For example, an entry in `/etc/passwd` on a UNIX system, an entry in the SAM database on a Windows system, and a UserProfile in RACF all represent accounts.
- **Administrator** — A person with responsibility for configuring and maintaining the Identity Manager system.

Identity Manager stores information about known resource accounts and users in the **account index**. At a minimum, each entry in the account index contains an account ID and an Identity Manager resource ID. An entry might also contain additional information, such as the native GUID or the status (enabled/disabled) of an account. An entry might also record the ID of an Identity Manager user as the owner of the account, or it might record a list of possible owners.

Types of Data Loading

Data loading is the process of importing account information from resources into Identity Manager and assigning these accounts to Identity Manager users.

Identity Manager supports the following features that load account data from resources:

- Discovery — Provides basic functions that initially load resource accounts into Identity Manager.
- Reconciliation — Periodically loads resource account information into Identity Manager, taking action on each account according to configured policy.
- Active Sync—Allows information that is stored in an “authoritative” external resource (such as an application or database) to synchronize with Identity Manager user data. An Active Sync-enabled adapter “listens” or polls for changes to the authoritative resource.

Each of these concepts is discussed in detail. A table comparing the types of data loading can be found in *Summary of Data Loading Types*.

Discovery

The discovery processes are designed to be used when a resource is being deployed for the first time. They provide a means to load account information into Identity Manager quickly. As a result, they do not provide all the features found in reconciliation or Active Sync. For example, the discovery process does not add entries to the Account Index. Nor can you run workflows before or after discovery. However, the discovery processes allow you to determine more quickly whether correlation rules are working as expected.

When you begin a discovery process, Identity Manager determines whether an input account matches (or correlates with) an existing user. If it does, the discovery process merges the account into the user. The process will create a new Identity Manager user from any input account that does not match.

Identity Manager provides the following discovery functions:

- Load From File — Reads accounts listed in a file and loads them into Identity Manager.
- Load From Resource — Extracts accounts from a resource and loads them directly into Identity Manager.
- Create Bulk Action — Executes user creation commands listed in a file.

Refer to the following sections for more information about each of these discovery processes.

Load from File

The Load from File discovery process reads account information that has been written into an XML or CSV (comma-separated values) file.

Some resources, such as Active Directory, have the ability to export native account information into a comma-separated values (CSV) format. These CSV files can be used to create Identity Manager accounts. See *Identity Manager Administration* for more information about CSV formatting.

When you load from a file, you must specify which account correlation and confirmation rules to use. See *Correlation and Confirmation Rules* on page 2-8 for more information.

Load from Resource

The Load from Resource feature scans a target system and returns information on all users. Identity Manager then creates and updates users. An adapter must have been configured for the resource before you can load from the resource.

When you load from a resource, you must specify which account correlation and confirmation rules to use. See *Correlation and Confirmation Rules* on page 2-8 for more information.

Create Bulk Action

Bulk actions allow you to act on multiple accounts at the same time. You can use bulk actions to create, update, and delete Identity Manager and resource accounts, but this discussion will be limited to Identity Manager creating accounts. See *Identity Manager Administration* for a full description of bulk actions.

Bulk actions are specified using comma-separated values (CSV). The structure of these values differs from those specified in a Load from File process.

The CSV format consists of two or more input lines. Each line consists of a list of values separated by commas. The first line contains field names. The remaining lines each correspond to an action to be performed on an Identity Manager user, the user's resource accounts, or both. Each line should contain the same number of values. Empty values will leave the corresponding field value unchanged.

Two fields are required in any bulk action CSV input:

- **user** — Contains the name of the Identity Manager user.
- **command** — Contains the action taken on the Identity Manager user. For creating Identity Manager users, this value must be Create.

The third and subsequent fields are from the User view. The field names used are the path expressions for the attributes in the views. See *Understanding the User View* in *Identity Manager Workflows, Forms, and Views* for information on the attributes that are available in the User View. If you are using a customized User Form, then the field names in the form contain some of the path expressions that you can use.

Following is a list of some of the more common path expressions used in bulk actions:

- **waveset.roles** — A list of one or more role names to assign to the Identity Manager account.
- **waveset.resources** — A list of one or more resource names to assign to the Identity Manager account.
- **waveset.applications** — A list of one or more resource groups to assign to the Identity Manager account.

- **waveset.organization** — The organization name in which to place the Identity Manager account.
- **accounts**[*resource_name*].*attribute_name* — A resource account attribute. The names of the attributes are listed in the schema for the resource.

Some fields can have multiple values. For example, the `waveset.resources` field can be used to assign multiple resources to a user. You can use the vertical bar (|) character (also known as the “pipe” character), to separate multiple values in a field. The syntax for multiple values can be specified like this:

```
value0 | value1 [ | value2 ... ]
```

The following example illustrates Create bulk actions:

```
command,user,waveset.resources,password.password,password.confi
rmPassword,accounts[AD].description,accounts[Solaris].comment

Create,John Doe,AD|Solaris,changeit,changeit,John Doe,John Doe
Create,Jane Smith,AD,changeit,changeit,Jane Smith,
```

The Create bulk action is more versatile than the from Load from File process. Bulk actions can work with multiple resources, while Load from File loads information from one resource at a time.

Reconciliation

Reconciliation compares the contents of the account index to what each resource currently contains. Reconciliation can perform the following functions:

- Detect new and deleted accounts
- Detect changes in account attribute values
- Correlate accounts with Identity Manager users
- Detect accounts that are not associated with Identity Manager users
- Run a workflow in response to each account situation that it detects
- Detect when a user has been moved form one container on a resource to another container on a resource

Note An adapter must have been configured for the resource before you can reconcile. See *Identity Manager Resources Reference* for more information about adapters.

There are two types of reconciliation: full and incremental.

Full Reconciliation

Full reconciliation recalculates the existence, ownership, and situation for each account ID listed by the adapter. It examines each Identity Manager user that claims the resource to recalculate ownership.

A Identity Manager user can claim a resource by:

- Having a role that implies the resource
- Having a direct resource assignment
- Referring to an account on that resource
- Having a resource group

For each account, reconciliation process confirms that any Identity Manager owner recorded in the Account Index still exists and still claims the account. Any account that does not have an owner is correlated with Identity Manager users (as long as reconciliation policy for that resource specifies a correlation rule). If a correlation rule suggests one or more possible owners, then each of them will be double-checked in a confirmation rule (if one is specified). See *Correlation and Confirmation Rules* on page 2-8 for more information about rules.

Once a situation has been determined for the account, reconciliation will perform any response that is configured in the reconciliation policy for that resource. If the reconciliation policy specifies a workflow to be performed per-account, full reconciliation will perform this for each account that is reconciled, after the situation action is performed. See *Reconciliation Workflows* on page 2-11 for more information about workflows.

Incremental Reconciliation

Incremental reconciliation is analogous to incremental backup: it is faster than full reconciliation, and does most of what you need, but is not as complete as full reconciliation.

Incremental reconciliation trusts that the information maintained in the account index is correct. Trusting that the list of known account IDs is correct, and that ownership of the account by any Identity Manager owner is correctly recorded, allows incremental reconciliation to skip or shorten several processing phases.

Incremental reconciliation skips the step of examining Identity Manager users that claim the resource. Incremental reconciliation also calculates a situation only for accounts that have been added or deleted since the resource was last reconciled. It does this by comparing the list of account IDs in the account index for that resource to

the list of account IDs returned by the resource adapter. New accounts are recorded as existing, deleted accounts are recorded as no longer existing, and only these two sets of accounts are processed further.

Because incremental reconciliation is much faster and uses fewer processing cycles than full reconciliation, you may want to schedule incremental reconciliation more frequently and schedule full reconciliation less often.

Active Sync

Active Sync "listens" or polls for changes to a resource, detecting incremental changes in real time. Because Active Sync is designed to detect changes, it should not be used to load account information into Identity Manager for the first time. Instead, use reconciliation or a discovery process.

In general, you run reconciliation on an Active Sync resource in the following circumstances:

- To perform an initial load on the resource.
- To detect any attributes that have not been updated in Identity Manager because Active Sync has been configured to ignore or filter out the attributes.

Active Sync differs from reconciliation in the following ways:

- Active Sync allows an administrator to specify a user form that ensures attributes across multiple accounts are kept synchronized.
- A process rule can be implemented that fully controls all Active Sync processing. This is typically enabled when extraordinary actions need to be performed when an account on a resource changes, such as editing multiple objects in the repository.

Active Sync requires the use of an Active Sync-enabled adapter that has been properly configured. See *Identity Manager Administration* for more information about configuring a resource to implement Active Sync.

Summary of Data Loading Types

The following table compares the capabilities of discovery and reconciliation.

Function	Discovery	Reconciliation	Active Sync
Detect new accounts	Yes	Yes	Yes
Detect deleted accounts	No	Yes	Yes

Function	Discovery	Reconciliation	Active Sync
Detect changes in account attribute values	No	Yes	Yes
Detect accounts that are not associated with Identity Manager users	Yes	Yes	Yes
Detect when a user has been moved from one container on a resource to another container on a resource	No	Yes	Yes
Correlate accounts with Identity Manager users	Yes	Yes	Yes
Run a workflow in response to each account situation that it detects	No	Yes	Yes
Can be scheduled	No	Yes	Yes
Incremental mode	No	Yes	Not applicable
Add entries to the account index	No	Yes	Yes
Synchronize attributes on multiple resources	No	No	Yes

Managing Reconciliation

The reconciliation process is primarily managed through the Administrator Interface. However, there are some aspects of reconciliation that cannot be accomplished from this interface. For example, you might need to create new correlation and confirmation rules, reconciliation workflows, or edit the Reconcile configuration object. The following sections describe these features, and others. For general information about defining reconciliation policy, see *Identity Manager Administration*.

Reconciliation Policy

Reconciliation policies allow you to establish a set of responses, by resource, for each reconciliation task. Within a policy, you select the server to run reconciliation, determine how often and when reconciliation takes place, and set responses to each situation encountered during reconciliation. You can also configure reconciliation to detect changes made natively (not made through Identity Manager) to account attributes.

Each of these policy settings can be defined at several scopes:

- Globally (for all resource types)
- For a specific resource type
- For an individual resource instance

The value at each scope becomes the default for each sub-scope. Thus, reconciliation policy defines an *inheritance tree*:

- The global value becomes the default for every resource type.
- Each resource type can inherit the global value or specify a value.
- The resource type value is the default for every resource instance of that type.
- Each resource instance can inherit value of its parent resource type, or specify a value.

Inheritance makes it easier to manage policy for a large number of resources (especially if many of them will have the same settings).

For example, if you want to treat all resources in the same way, you need to manage only one set of policy settings, at the global level. If you want to treat all Windows resources one way and all Solaris resources another way, you need to manage policy settings at only two scopes: one for each of these two resource types. If there are exceptions to the policy defined at the resource type level for a few specific resource instances, the necessary policy settings can be overridden (specified) for those individual resources. Since each policy setting is inherited separately, only the settings that differ need to be specified; the other policy settings may still inherit their values from above.

Correlation and Confirmation Rules

Identity Manager matches resource accounts that are not linked to a user with Identity Manager users in two phases:

- **Correlation** -- Finding potential owners
- **Confirmation** -- Testing each potential owner

A *correlation rule* looks for Identity Manager users that might own an account. A *confirmation rule* tests a Identity Manager user against an account to determine whether the user actually does own the account. This two-stage approach allows Identity Manager to optimize correlation, by quickly finding possible owners (based on name or attributes), and by performing expensive checks only on the possible owners.

Reconciliation policy allows you to select a correlation rule and a confirmation rule for each resource. (You may also specify No Confirmation Rule.) The default correlation rule is to look for a user with a name that exactly matches the account ID of the input account. By default, no confirmation rule is used.

Note Correlation and confirmation rules are also used for discovery and Active Sync.

Identity Manager predefines a number of correlation and confirmation rules in `sample/reconRules.xml`. You can also write your own correlation and confirmation rules. Any rule object with a subtype of `SUBTYPE_ACCOUNT_CORRELATION_RULE` or `SUBTYPE_ACCOUNT_CONFIRMATION_RULE` automatically appears in the appropriate Reconciliation Policy selection list.

Correlation Rules

A correlation rule can generate a list of user names based on values of the attributes of the resource account. A correlation rule may also generate a list of attribute conditions (referring to queryable attributes of a user object) that will be used to select users.

A correlation rule is run once for each unclaimed account.

Note A correlation rule should be relatively “inexpensive” but as selective as possible. If possible, defer expensive processing to a confirmation rule.

Identity Manager predefines several correlation rules in `sample/reconRules.xml`:

- **User Name Matches AccountId** -- Returns the value of the `accountId` attribute. It selects as a possible owner any Identity Manager user with a name that matches the resource account ID. This is the default correlation rule.
- **User Owns Matching AccountId** -- Returns a list of attribute conditions. This will select as a possible owner any Identity Manager user that owns a resource account that matches the same `accountId` value.
- **User Email Matches Account Email** -- Returns a list of attribute conditions that will select Identity Manager users based on the account's `email` attribute.

Input for any correlation rule is a map of the account attributes. Output must be one of:

- String (containing user name or ID)
- List of String elements (each a user name or ID)
- List of WSAttribute elements
- List of AttributeCondition elements

A more complicated rule might combine or manipulate account attribute values to generate a list of names or a list of attribute conditions.

Note Attribute conditions must refer to queryable attributes, which are configured as `QueryableAttrNames` in the UserUIConfig object.

For example, `reconRules.xml` contains a fourth sample correlation rule, User FullName Matches Account FullName. XML comments disable this rule, because it will not work correctly without additional configuration. This rule looks for Identity Manager users based on `fullname`, but this attribute is not queryable by default.

Correlating on an extended attribute requires special configuration:

- The extended attribute must be specified as queryable in `UserUIConfig` (added to the list of `QueryableAttrNames`).
- The Identity Manager application (or the application server) may need to be restarted for the `UserUIConfig` change to take effect.

Confirmation Rules

A confirmation rule is run once for each matching user returned by the correlation rule.

A typical confirmation rule compares internal values from the user view to the values of account attributes. As an optional second stage in correlation processing, the confirmation rule performs checks that cannot be expressed in a correlation rule (or that are too expensive to evaluate in a correlation rule). In general, you need a confirmation rule only in the following circumstances:

- The correlation rule may return more than one matching user
- User values that must be compared are not queryable

Identity Manager predefines two confirmation rules in `sample/reconRules.xml`:

- **User Email Matches Account Email** -- Returns a value of true if the user's email matches the account's email. This illustrates the fact that many ownership decisions could be expressed with either a correlation rule or a confirmation rule. However, since the `email` attribute of an Identity Manager user is automatically queryable, it would almost always be more efficient to express this as a correlation rule.
- **User First And Last Names Match Account** -- Uses the XPRESS language to compare the user's first and last name to the same values of the account.

Inputs to any confirmation rule are:

- **userview** -- Full view of an Identity Manager user.
- **account** -- Map of resource account attributes

A confirmation rule returns a string-form Boolean value of true if the user owns the account; otherwise, it returns a value of false.

The default confirmation rule is No Confirmation Rule. This assumes that the correlation rule is selective enough to find at most one user for each account. If the correlation rule selects more than one user, the account situation will be DISPUTED.

Reconciliation Workflows

You can extend normal reconciliation processing by exposing a number of attachment points for user-defined workflows.

Pre-Resource Workflow

A pre-resource workflow can be launched before any other reconciliation processing is started. The Notify Reconcile Start workflow is an example of a pre-resource workflow.

The Notify Reconcile Start workflow e-mails an administrator with notice that a reconcile has started for a resource. You must configure the Notify Reconcile Start e-mail template before running this workflow.

The following parameters are passed to the pre-resource workflow:

- **resourceName** -- Name of the resource that will be reconciled.
- **resourceId** -- Object ID of the resource that will be reconciled.

Per-Account Workflow

The per-account workflow is launched for each account processed by reconciliation, after the response (if any) has completed. The type of response and the response result do not affect this workflow.

The Notify Reconcile Response workflow is an example of a per-account workflow. It e-mails an administrator when the reconcile process attempts to automatically respond to a discovered situation.You must configure the Notify Reconcile Response e-mail template before running this workflow.

The following parameters are passed to the per-account workflow:

- **accountId** -- Name of the resource account that was reconciled.
- **resourceId** -- Object ID of the resource being reconciled.
- **resourceName** -- Name of the resource being reconciled.
- **userID** -- Object ID of the Identity Manager user identified as the account owner (by claim or correlation, depending on the situation). If no user is associated with the account, this is null.

- **userName** -- Name of the Identity Manager user identified as the account owner (by claim or correlation, depending on the situation). If no user is associated with the account, this is null.
- **initialSituation** -- The situation that was initially discovered for the account, triggering the response. The value is a valid message key.
- **responseSuccess** -- Boolean value indicating whether the response completed successfully. If no response was performed, this is null.
- **finalSituation** -- Reconciliation situation the account was in after applying the response. If the account no longer exists and Identity Manager contains no references to it, this is null.

Post-Resource Workflow

A post-resource workflow can be launched after all other reconciliation processing is complete. The Notify Reconcile Finish workflow is an example post-resource workflow.

The Notify Reconcile Finish workflow e-mails an administrator with notice that a reconcile has finished for a resource. You must configure the Notify Reconcile Finish e-mail template before running this workflow.

The following parameters are passed into the post-resource workflow:

- **resourceName** -- Name of the resource that was reconciled.
- **resourceId** -- Object ID of the resource just reconciled.

Auditing Native Changes

The Audit Native Change To Account Attributes workflow is launched when reconciliation or the provisioner detects a change to the attributes of a resource account that was not initiated through Identity Manager. Only user-specified attributes are monitored for changes. By default, no attributes are monitored.

The following parameters are passed to the workflow:

- **resource** -- Resource object where the account was changed natively.
- **accountID** -- Name of the resource account that was changed natively.
- **prevAttributes** -- Map containing the monitored resource account attributes recorded by Identity Manager.
- **newAttributes** -- Map containing the monitored resource account attributes currently set on the resource.
- **attributeChanges** -- Map containing the List of generic objects that indicate which attributes have changed. Each object contains the previous and new values.

- **formattedChanges** -- String representing the attribute changes in compact format, suitable for an audit record.

To audit native changes, you must do the following:

- On the **Edit Reconciliation Policy** page, select the **Detect native changes to account attributes** option from the **Attribute-level reconciliation** drop-down menu. You might need to uncheck the **Inherit resource type policy** check box to display a list of attributes. Select the attributes to audit.
- Add **Changes Outside Identity Manager** to the list of audit events. To do this, select the Configure tab, then Audit Events on the left.

Resource Scheduling

Reconciliation maintains two separate schedules for each resource: one for incremental reconciliation, and another for full reconciliation.

Each resource is scheduled by a separate “requester” task. Configuring a reconciliation schedule for a resource in the reconciliation policy GUI configures the TaskSchedule for the requester task. This allows reconciliation to be controlled by an external task scheduler, if desired. It also minimizes the overhead of the reconciliation task. A reconciliation daemon that is not doing anything consumes very few resources, since it periodically polls an in-memory queue (rather than querying the database for resources that are ready to reconcile).

Reconciliation accesses each resource through a resource adapter. Reconciliation calls the adapter *directly* to list accounts, iterate accounts, or fetch an individual resource account. Reconciliation also accesses resources *indirectly* through Provisioner, and uses Provisioner to create a resource account or Identity Manager user from a resource account.

Reconciliation and Provisioner both maintain the account index. Also, reconciling a resource prunes the Account Index each time. The reconciliation task automatically removes any entry for an account that no longer exists on the resource, unless that account is owned by a Identity Manager user. Therefore, it should not be necessary to attempt to manually clear the Account Index for a resource.

Each Identity Manager server runs reconciliation as a daemon task. This means that the Identity Manager scheduler starts the reconciliation task immediately and automatically restarts the task if it dies.

Note Resource reconciliation is not automatically restarted. The ReconTask daemon itself is automatically restarted, which enables it to respond to any new request; but any request in process when the host server dies (or when the application server is shut down) is lost. The daemon does not restart resource reconciliation because it may be inappropriate to reconcile the resource at a time other than when requested.

Reconcile Configuration Object

The ReconcileConfiguration object contains several attributes that cannot be edited from the Edit Reconciliation Policy page.

The following table defines the reconciliation attributes. Use the debug pages to edit the attribute values in the ReconcileConfiguration object (#ID#Configuration:ReconcileConfiguration).

Attribute	Description
fetchTimeout	The number of milliseconds the reconciliation process should wait for a response from a resource when fetching an account. The default value is 1 minute (60000 milliseconds).
listTimeout	The number of milliseconds the reconciliation process should wait for a response from a resource when listing accounts. The default value is 10 minutes (600000 milliseconds).
maxConcurrentResources	The maximum number of resources that a server should reconcile concurrently. The default value is 3.
maxQueueSize	The maximum number of entries in a reconciliation server's work queue. The default value is 1000.

The following example shows the default values for the ReconcileConfiguration object.

```
<Configuration id='#ID#Configuration:ReconcileConfiguration'
name='Reconcile Configuration'>
 <Extension>
  <Object>
    <Attribute name='listTimeout' value='600000' />
    <Attribute name='fetchTimeout' value='60000' />
    <Attribute name='maxConcurrentResources' value='3' />
```

```
    <Attribute name='maxQueueSize' value='1000' />
  </Object>
 </Extension>
 <MemberObjectGroups>
  <ObjectRef type='ObjectGroup' id='#ID#All' name='All'/>
 </MemberObjectGroups>
</Configuration>
```

Managing Active Sync

Active Sync-enabled adapters can be managed in the Administrator Interface. This interface contains a wizard that allows an administrator to fully configure most aspects of Active Sync on a single adapter. The wizard also allows the administrator to construct a resource, or input, form, without using the Business Process Editor (BPE). For more details about the Active Sync wizard, see *Identity Manager Administration*.

How Active Sync-Enabled Adapters Work

This section describes:

- Overview of the basic steps of adapter processing
- Using rules
- Using forms
- Launching workflow processes

Basic Steps of Adapter Processing

All Active Sync-enabled adapters follow the following basic steps when listening or polling for changes to the resource defined in Identity Manager. When the adapter detects that a resource has changed, the Active Sync-enabled adapter:

1. Extracts the changed information from the resource.
2. Determines which Identity Manager object is affected.
3. Builds a map of user attributes to pass to the system, along with a reference to the adapter and a map of any additional options, which creates an Identity Application Programming Interface (`IAPI`) object.
4. Submits the `IAPI` object to the `ActiveSync` Manager.
5. `ActiveSync` Manager processes the object and returns to the adapter a `WavesetResult` object that informs the Active Sync-enabled adapter if the operation succeeds. This object can contain many results from the various steps

that the Identity Manager system uses to update the identity. Typically, a workflow also handles errors within Identity Manager, often ending up as an Approval for a managing administrator.

Using Rules

When the Active Sync-enabled adapter detects a change to an account on a resource, it either maps the incoming attributes to an Identity Manager user, or creates an Identity Manager user account if none can be matched and if the Active Sync resource has been configured to do so.

The Active Sync wizard allows you to specify rules to control what happens when various conditions occur. The following table describes each type of rule.

Parameter	Description
Process Rule	Either the name of a TaskDefinition, or a rule that returns the name of a TaskDefinition, to run for every record in the feed. The process rule gets the resource account attributes in the activeSync namespace, as well as the resource ID and name. A process rule controls all functionality that occurs when the system detects any change on the resource. It is used when full control of the account processing is required. As a result, a process rule overrides all other rules. If a process rule is specified, the process will be run for every row regardless of any other settings on this adapter. At minimum, a process rule must perform the following functions: • Query for a matching User view. • If the User exists, checkout the view. If not, create the User. • Update or populate the view. • Checkin the User view. It is possible to synchronize objects other than User, such as LDAP Roles.

Parameter	Description
Correlation Rule	If no Identity Manager user's resource info is determined to own the resource account, the Correlation Rule is invoked to determine a list of potentially matching users/accountIDs or Attribute Conditions, used to match the user, based on the resource account attributes (in the account namespace). The rule returns one of the following pieces of information that can be used to correlate the entry with an existing Identity Manager account: • Identity Manager user name • WSAttributes object (used for attribute-based search) • List of items of type AttributeCondition or WSAttribute (AND-ed attribute-based search) • List of items of type String (each item is the Identity Manager ID or the user name of an Identity Manager account) If more than one Identity Manager account can be identified by the correlation rule, a confirmation rule or resolve process rule will be required to handle the matches. For the Database Table, Flat File, and PeopleSoft Component Active Sync adapters, the default correlation rule is inherited from the reconciliation policy on the resource. The same correlation rule can be used for reconciliation and Active Sync. See *Correlation and Confirmation Rules* on page 2-8 for more information.
Confirmation Rule	Rule which is evaluated for all users returned by a correlation rule. For each user, the full user view of the correlation Identity Manager identity and the resource account information (placed under the “account.” namespace) are passed to the confirmation rule. The confirmation rule is then expected to return a value which may be expressed like a Boolean value. For example, “true” or “1” or “yes” and “false” or “0” or null. For the Database Table, Flat File, and PeopleSoft Component Active Sync adapters, the default confirmation rule is inherited from the reconciliation policy on the resource. The same confirmation rule can be used for reconciliation and Active Sync. See *Correlation and Confirmation Rules* on page 2-8 for more information.

Parameter	Description
Delete Rule	A rule that can expect a map of all values with keys of the form `activeSync.` or `account.` A LighthouseContext object (`display.session`) based on the proxy administrator's session is made available to the context of the rule. The rule is then expected to return a value which may be expressed like a Boolean value. For example, "true" or "1" or "yes" and "false" or "0" or null. If the rule returns true for an entry, the account deletion request will be processed through forms and workflow, depending on how the adapter is configured.
Resolve Process Rule	Either the name of the TaskDefinition or a rule that returns the name of a TaskDefinition to run in case of multiple matches to a record in the feed. The Resolve Process rule gets the resource account attributes as well as the resource ID and name. This rule is also needed if there were no matches and **Create Unmatched Accounts** is not selected. This workflow could be a process that prompts an administrator for manual action.
Create Unmatched Accounts	If set to true, creates an account on the resource when no matching Identity Manager user is found. If false, the account is not created unless the process rule is set and the workflow it identifies determines that a new account is warranted. The default is true.
Populate Global	If set to true, populates the global namespace in addition to the activeSync namespace. The default value is false.

If the Adapter Does Not Find the User...

If Identity Manager cannot find a match with an existing Identity Manager user, it turns an update operation into a create operation if the Create Unmatched Accounts setting is true, or the Resolve Process workflow indicates a `feedOp` of `create`.

The `feedOp` field is available to forms that contain logic to create, delete, or update users. You can use this field to disable or enable fields that are specific to one kind of event (for example, the generation of a password when the `feedOp` field is set to create).

This example `feedOp` field creates a password only when the Active Sync-enabled adapter detects a user on the resource that is not matched by a user in Identity Manager, and creates the user in Identity Manager.

```
<Field name='waveset.password'>
   <Disable>
      <neq>
         <ref>feedOp</ref>
```

```
            <s>create</s>
         </neq>
      </Disable>
      <expression>
         <cond>
            <notnull>
               <ref>activeSync.password</ref>
            </notnull>
            <ref>activeSync.password</ref>
            <s>change12345</s>
         </cond>
      </expression>
   </Field>
```

Using Forms

Active Sync-enabled adapters typically use two types of forms during processing: a *resource form* and a *user form*.

Form processing occurs in three steps:

1. Active Sync fields are filled in with attribute and resource information. Use the `activeSync` namespace to retrieve and set attributes on the resource.
2. The resource form is expanded and derived. During this expansion, all user view attributes are available.
3. The user form is expanded and derived.

The `$WSHOME/sample/forms` directory provides sample forms that end with `ActiveSyncForm.xml`. They include logic for handling the cases of new and existing users, as well as logic for disabling or deleting the Identity Manager user when a deletion is detected on the resource.

Note Sun strongly recommends that you place only resource-specific logic in the resource form and that common logic stays in the user form, possible enabled when the `feedop` field is not null. If the resource form is "none", all of the Active Sync attributes (except `accountId`) are named global and will propagate automatically.

Resource Form

The *resource form* is the form that the administrator selects from a pull-down menu when the resource is created or edited. A reference to a selected form is stored in the resource object.

Resource forms are used with Active Sync-enabled adapters in the following ways:

- Translate incoming attributes from the schema map
- Generate fields such as password, role, and organization
- Provide simple control logic for custom processing, including logic for handling the cases of new and existing users, as well as logic for disabling or deleting the Identity Manager user when a deletion has been detected
- Copy and optionally transform attributes from `activeSync` to fields that the user form takes as inputs. The required fields for a creation operation are `waveset.accountId` and `waveset.password`. Other field can be set, too, ((for example, `accounts[NT].email` or `waveset.resources`).
- Cancel the processing of the user by setting `IAPI.cancel` to `true`. This is often used to ignore updates to certain users.

  The following example shows a simple field that will ignore all users with the last name Doe:

```
<Field name='IAPI.cancel'>
   <Disable>
      <neq>
         <ref>activeSync.lastName</ref>
         <s>Doe</s>
      </neq>
   </Disable>
   <expression>
      <s>true</s>
   </expression>
</Field>
```

Resource forms include logic for handling the cases of new and existing users, as well as logic for disabling or deleting the Identity Manager user when a deletion has been detected.

User Form

The *user form* is used for editing from the Identity Manager interface. You assign it by assigning a *proxy administrator* to the adapter. If the proxy administrator has a user form associated with him, this form is applied to the user view at processing time.

Proxy Administrator and the User Form

You set a proxy administrator for an adapter through the `ProxyAdministrator` attribute, which you can set to any Identity Manager administrator. All Active Sync-enabled adapter operations are performed as though the Proxy Administrator was performing them. If no proxy administrator is assigned, the default user form is specified.

Alternative Form to Process Attributes

Best practice suggests keeping common changes, such as deriving a fullname from the first and last name, in the *user form*. The *resource form* should contain resource-specific changes, such as disabling the user when their HR status changes. However, you can alternatively place it in an included form after the desired attributes are placed in a common path, such as incoming.

Examples

```
<Form>
   <Field name='incoming.lastname'>
      <ref>activeSync.lastname</ref>
   </Field>
   <Field name='incoming.firstname'>
      <ref>activeSync.firstname</ref>
   </Field>
</Form>
```

Subsequently, in the common form, reference incoming.xxx for the common logic:

```
<Form>
   <Field name='fullname'>
      <concat>
          <ref>incoming.firstname</ref>
          <s> </s>
          <ref>incoming.lastname</ref>
      </concat>
   </Field>
</Form>
```

Process Cancel Action

To cancel the processing of a user, set `IAPI.cancel` to `true` in the resource form. You can use this to ignore updates to certain users.

Note If `IAPI.cancel` is set to a value of true in an Active Sync form, then the process associated with an IAPIUser or IAPIProcess event will not be launched.

The following example shows a simple field in the resource form that ignores all users with the last name *Doe*:

```
<Field name='IAPI.cancel'>
   <Disable>
      <eq><ref>activeSync.lastName</ref><s>Doe</s></eq>
```

```
   </Disable>
   <Expansion>
      <s>true</s>
   </Expansion>
</Field>
```

Launching Workflow Processes

The Active Sync wizard allows an administrator to specify a pre-poll and post-poll workflow. These workflows are similar in concept to the workflows discussed in *Reconciliation Workflows* on page 2-11.

Some Active Sync-enabled adapters support a resource attribute that runs a specified workflow instead of checking the pulled changes into the user view. This workflow is run with an input variable of only the Active Sync data. For adapters that do not support a separate process, or one where you want to use the standard user form and then launch a process, you can override the process by setting options.

```
<Form>
   <Field name='sourceOptions.Process'>
      <Expansion>
         <s>My workflow process name</s>
      </Expansion>
   </Field>
</Form>
```

The workflow specified through the form is called just like a standard provisioning workflow. Sun strongly recommends that you base your custom workflow on the standard create and update workflow. Consult the create and update user workflows in `workflow.xml`.

Example: Disabling Accounts through Active Sync-Enabled Adapters

In this example, the resource (an HR database) can be updated with an employee's current status at the company. Based on the input from this HR database, the Active Sync-enabled adapter can disable, delete, create, or perform other actions on the user's accounts across the enterprise by updating the Identity Manager repository.

The following code example disables all accounts for an employee if there is an incoming attribute called Status and it is not active (“A”). The following table identifies the four states of this attribute.

State	Description
A	active
T	terminated
L	laid off
S	pending change

Based on the value of the Status attribute, the account can be disabled or enabled.

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE Configuration PUBLIC 'waveset.dtd' 'waveset.dtd'>
<Configuration wstype='UserForm' name='PeopleSoft ActiveSync Form'>
   <Extension>
   <Form>

<!-- this is a sample of how to map the accountID to a different
field than the one from the schema map

Commented out because we want to use the default account
ID mapped from the resource Schema Map.

      <Field name='waveset.accountId'>
         <Disable><neq><ref>feedOp</ref><s>create</s></neq></Disable>
         <Expansion>
            <concat><s>ps</s><ref>waveset.accountId</ref></concat>
         </Expansion>
      </Field>
    -->

    <!-- this is the real one, limited to create -->

      <Field name='waveset.accountId'>
         <Disable><neq><ref>feedOp</ref><s>create</s></neq></Disable>
         <Expansion>
            <ref>activeSync.EMPLID</ref>
         </Expansion>
      </Field>

    <!-- we need to make up a password for accounts that are being
            created. This picks the last six digits of the SSN.
    -->
      <Field name='waveset.password'>
```

```
         <Disable><neq><ref>feedOp</ref><s>create</s></neq></Disable>
         <expression>
            <s>change123456</s>
         </expression>
      </Field>

      <Field name='waveset.resources'>
<!--     <Disable><neq><ref>feedOp</ref><s>create</s></neq></Disable>
-->
<!-- Don't change the resources list if it already contains peoplesoft
-->
         <Disable>
            <member>
               <ref>activeSync.resourceName</ref>
               <ref>waveset.resources</ref>
            </member>
         </Disable>

         <expression>
            <appendAll>
               <ref>waveset.resources</ref>
               <ref>activeSync.resourceName</ref>
            </appendAll>
         </expression>
      </Field>

    <!-- Status is mapped by the schema map to PS_JOB.EMPL_STATUS which
has at least four states - A for active, T terminated,
L laid off, and S which is a pending change.

The audit data tells us what the state was, and the global
data tells us what it is. Based on the change we can disable
or enable the account

Note that this can happen on a create also!
    -->
      <Field>
         <Disable><eq><ref>activeSync.Status</ref><s>A</s></eq></Disab
le>

      <Field name='waveset.disabled'>
         <Expansion>
            <s>true</s>
         </Expansion>
      </Field>

      <FieldLoop for='name' in='waveset.accounts[*].name'>
      <Field name='accounts[$(name)].disable'>
         <expression>
```

```
            <s>true</s>
         </expression>
      </Field>
      </FieldLoop>

      </Field>


<!-- Status is mapped by the schema map to PS_JOB.EMPL_STATUS which
has at least four states - A for active, T terminated,
L laid off, and S which is a pending change.

This is the enable logic. It is disabled if the account
status is <> A or is already enabled
-->
      <Field>
         <Disable>
            <neq>
               <ref>activeSync.Status</ref>
               <s>A</s>
            </neq>
         </Disable>


      <Field name='waveset.disabled'>
         <Disable><eq><ref>waveset.disabled</ref><s>false</s></eq></Di
sable>
            <Expansion>
               <s>false</s>
            </Expansion>
      </Field>

      <FieldLoop for='name' in='waveset.accounts[*].name'>
         <Field name='accounts[$(name)].disable'>
            <Expansion>
               <s>false</s>
            </Expansion>
         </Field>
      </FieldLoop>


      </Field>

   </Form>
   </Extension>
   <MemberObjectGroups>
      <ObjectRef type='ObjectGroup' id='#ID#Top' name='Top'/>
   </MemberObjectGroups>

</Configuration>
```

3 Data Loading Scenarios

This chapter provides tips to consider when preparing to load account information into Identity Manager. It also includes sample scenarios that illustrate some of the issues that you might encounter.

Assessing Your Environment

Before you can begin loading user account information into Identity Manager, you determine which know the following questions applies to your environment:

- Is there an authoritative resource for all user IDs?

  If yes, then loading user accounts into Identity Manager should be straightforward. Use that resource as your first resource, then load accounts from other resources, using correlation rules to link the accounts together.
- Can a complete list of users be obtained from resources that overlap, but for which there is a correlation key?

  If yes, then the process of loading user accounts will be similar. Be sure that the user accounts are loaded from the overlapping resources into Identity Manager before loading accounts from other resources.
- Can a list of users be obtained from overlapping resources that do not have a correlation key?

  If yes, then determine how a unique set of users can be discovered from those resources most easily. You will probably need to manually correlate and delete users for each resource.

If the answer is no to all these questions, then the process of loading accounts is problematic. Load user accounts as best you can, and plan to delegate creation of other Identity Manager users to departmental administrators or end-users.

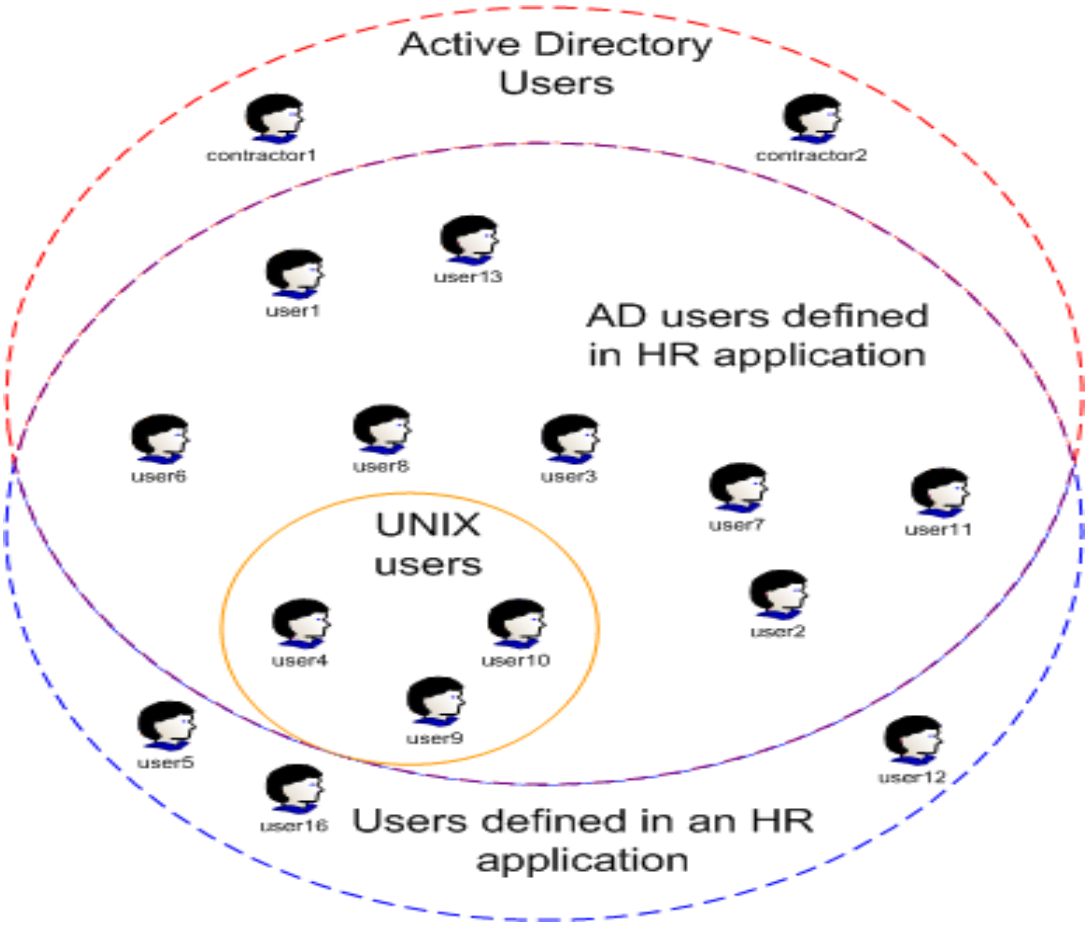
Choosing the First Resource

Ideally, the first resource you use to load accounts into Identity Manager has the following characteristics:

- **References a comprehensive set of users.** The goal of an initial load is get as many accounts into Identity Manager as possible. Thus, the following applications might be a good choice:

  - A Human Resources application, such as PeopleSoft or SAP. (If the application does not contain contractors and other temporary workers, be sure to load those accounts separately.)
  - A directory-based application, such as LDAP or Active Directory. A majority of users are often defined in a central organization or organization unit.
- **Contains enough information to construct an Identity Manager account ID**. Each Identity Manager account ID must be unique. Ideally, your resource will have an attribute that is guaranteed to be unique and can be used as a Identity Manager account ID. Examples include an employee IDs or Active Directory sAMAccountName attributes. First and last names can also be concatenated to produce an account ID, but this technique might not guarantee a unique Identity Manager account will be generated.
- **Stores user attributes that can be used correlation keys.** To link resource accounts in Identity Manager, you must have attributes that have the same values across two or more resources. Ideally, the values on the secondary resources will always perfectly match values on the first resource. In addition, it would be ideal if the values on the secondary resource are unique within that resource. The best attributes include employee ID and full name, but any other attribute that is present and consistent across multiple resources is acceptable.
- **Contains data that can be considered authoritative.** If users can edit their own account data, then the data might not be consistent across systems.

The following diagram illustrates a small scenario in which a company has three types of resources. Most of the company's workers are defined in a Human Resources application, such as PeopleSoft or SAP. However, the company does not enter contractors in the HR application, so the contractors cannot be loaded into Identity Manager using this application. The Active Directory also defines most, but not all, users. (These users might be factory workers with no need for computer access.) Thus, the majority of users are defined in both resources, but neither contains all the users. Some workers also have UNIX accounts.



Which resource should be selected as the first resource? The UNIX resource can be safely eliminated, because it does not contain a comprehensive set of users. Active Directory and the HR application contain about the same number of users, so neither has a clear advantage.

Factors that can help determine whether the Active Directory or HR application should be loaded first include the following:

- Urgency to manage accounts within Identity Manager. If the workers not defined in Active Directory (that is, they are defined in the HR application only) do not have any additional resource accounts, such as UNIX or other systems, then the HR application might not be as crucial as the Active Directory resource.

- Correlation keys. If one resource has attributes that are also on the UNIX accounts, then that attribute might be a better choice.
- Identity Manager login names. If one resource creates a more desirable login name of Identity Manager, then this can be a deciding factor.

Choosing the First Data Loading Process

After you have chosen which resource will be used as the starting point for loading user data into Identity Manager, you must decide which process to use. The following table provides a summary of the benefits and drawbacks of each data loading process. A discussion of each data loading process follows.

Data Loading Process	Advantages	Disadvantages
Load from File	• Quickest loading process. • Easy to control which attributes are loaded.	• Easier to configure and faster than reconciliation. • Requires customer time to generate a CSV file from a resource. • Requires a full reconciliation before production. • Cannot be used to update accounts.
Load from Resource	• Works with all resources • Easier to configure than reconciliation	• Cannot pick and choose which resource accounts will be loaded. • Requires a full reconciliation before production.
Create bulk action	• Allows you to add multiple accounts simultaneously to an Identity Manager user.	• Slower than loading from resource or reconciliation. • Cannot easily generate the CSV file from resources. • Requires detailed knowledge of Identity Manager to make full use of this feature. • Requires a full reconciliation before production.

Data Loading Process	Advantages	Disadvantages
Reconciliation	• Can implement all aspects of reconciliation policy • Using reconciliation up-front prevents last-minute surprises	• Cannot pick and choose which resource accounts will be loaded. • Can take a large amount of time to load all accounts in large environments (over 50,000 employees)
ActiveSync	If at all possible, avoid choosing ActiveSync as the means to load account information. ActiveSync is designed to detect changes, and as a result, initial loads are slow.	

Load from File

The Load from File process seeds Identity Manager accounts with basic values, such as account ID, first and last name, and e-mail address. The account ID is the only required attribute.

The Load from File process imports the contents of a comma-separated values (CSV) file into Identity Manager. The top line of this file contains a list of attribute names, separated by commas. Each subsequent line contains a series of corresponding attribute values. All attributes must also be separated by commas.

Note Load from File also accepts XML files, but the syntax of an XML file must match the syntax generated by the Extract to File feature. This format is beyond the scope of this discussion.

The data in a CSV file is often exported from a resource. For example, the Active Directory Users and Computers MMC (Microsoft Management Console) allows you to export the contents of an organization unit directly into a CSV file. The console exports all users defined in the organization unit as well as the displayed attributes. Therefore, you should verify only attributes that will be managed by Identity Manager are displayed in the Active Directory Users and Computers MMC. Including extraneous attributes will cause loading times to increase.

Some resources are not capable of directly exporting user account information to CSV format. If you wish to use to this method of data loading, you might need to extract the information programmatically or add data manually.

For example, the first three lines of a CSV file might look like this:

```
accountId,firstname,lastname,EmployeeID
Josie.Smith,Josie,Smith,1436
AJ.Harris,Anthony,Harris,c310
```

The attributes listed in the CSV file must be pre-defined as user view attributes. Basic attributes such as `accountId`, `email`, `password`, and `confirmpassword` are pre-defined. Others are defined in the extended user attributes configuration object. By default, this object adds `firstname`, `lastname`, and `fullname` to the list of available attributes.

If you want to retain values for attributes that are not pre-defined in Identity Manager, such as an employee ID, you must add them to the extended user attributes configuration object.

The Load from File process configuration page prompts for a correlation and confirmation rules. Since this is the first attempt to load data, select the **User Name Matches AccountId** correlation rule. You do not need a confirmation rule.

It is important to remember that the data contained in the CSV file is used for Identity Manager accounts only. Even if the data is exported directly from Active Directory, for example, the data is not linked to any Active Directory accounts or resources unless you have created a custom user form to do this. Without a custom user form, a different data loading mechanism must be used to link resource account data to a Identity Manager user. User forms are discussed briefly in *Assigning User Forms* on page 3-10.

Note The Load from File process does not add entries into the Identity Manager account index. Therefore, you must perform a full reconciliation, or update the users, before your Identity Manager deployment is complete. In addition, the Load from File does not run any workflows when creating users in Identity Manager.

Load from Resource

Although the configuration pages for the Load from Resource and Load from File processes are almost identical, the Load from Resource is functionally closer to reconciliation. The Load from Resource and reconciliation processes pull data from the resource, and then adds the accounts it finds to Identity Manager. Therefore, the adapter must be configured before you perform either of these operations.

Load from Resource is faster on the initial run than reconcile but does not populate the Account Index. Reconcile on the second execution running in Incremental mode should be faster than Load from Resource. If reconciliation is the desired tool as the long-term solution, then use reconciliation for the initial load of users.

A user form can be used to set the account ID, place users in an Organizations, and perform other related tasks related to creating users. See *Assigning User Forms* on page 3-10 for more information.

When you seed Identity Manager accounts for the first resource using Load from Resource, the correlation and confirmation rules are not very meaningful. Select the **User Name Matches AccountId** correlations rule. You do not need a confirmation rule.

Note The Load from Resource process does not add entries into the Identity Manager account index. Therefore, you must perform a full reconciliation, or update the users, before your Identity Manager deployment is complete. In addition, the Load from File does not run any workflows when creating users in Identity Manager.

Create Bulk Actions

The Create bulk action loads data from a CSV file. Unlike the Load from File process, the create bulk action allows you to define any writable attribute in the user view, including Identity Manager-specific attributes, global attributes, and resource account attributes. This flexibility means that it will probably be more difficult to assemble a bulk actions CSV file. If your bulk actions affect multiple resources, you will need to find a way to merge resource data into a single CSV file. However, you could also define a simpler bulk action file and use subsequent update actions to load data into Identity Manager user accounts.

Bulk actions runs the default workflows for create, update, and delete actions. This slows down the process of loading user accounts, but add greater flexibility.

The following example illustrates the use of Create bulk actions only. The command and user attributes are required.

```
command,user,waveset.resources,password.password,password.confirmPassw
ord,accounts[MyAD].description,accounts[MySolaris].comment

Create,John Doe,MyAD|MySolaris,changeit,changeit,John Doe,John Doe
Create,Jane Smith,MyAD,changeit,changeit,Jane Smith
```

The following example illustrates how the Create and Update bulk actions can be used in two separate files.

```
command,user,waveset.resources,password.password,password.confirmPassw
ord,accounts[MyAD].description,
Create,John Doe,MyAD,changeit,changeit,John Doe,John Doe
Create,Jane Smith,MyAD,changeit,changeit,
Jane Smith

command,user,waveset.resources,password.password,password.confirmPassw
ord,accounts[MySolaris].comment
Update,John Doe,MySolaris,changeit,changeit,John Doe
Update,Jane Smith,MySolaris,changeit,changeit,Jane Smith
```

Note Creating accounts using bulk actions does not add entries into the Identity Manager account index. Therefore, you must perform a full reconciliation before your Identity Manager deployment is complete.

Reconciliation

The first reconciliation of a resource will probably take longer than any subsequent reconciliation. You can expect the first reconciliation of a resource to add a large number of Account Index entries.

Preparing for Data Loading

Review the following sections before you begin the process of loading account information into Identity Manager:

- Configuring an Adapter
- Setting Account ID and Password Policies
- Creating a Data Loading Account
- Assigning Forms

Configuring an Adapter

To manage accounts on resources, you must configure an adapter for each source of account information. If you are using the Load from File process or bulk actions, then the adapter configuration can wait until you are ready to reconcile. Otherwise, the adapter must be configured before you can load data into Identity Manager.

For general information about configuring an adapter, see the *Identity Manager Administration* Guide. For detailed information about a specific adapter, refer to the *Identity Manager Resources Reference* or the online helps.

Setting Account ID and Password Policies

When you load account data from a resource via Load from Resource, reconciliation, or ActiveSync, Identity Manager does not obtain the password from the resource. (It would be a security breach on the part of the resource if it yielded the password.) Therefore, the Identity Manager account passwords will not be the same as the those on the resource. By default, Identity Manager generates a random password that must be reset. However, you can also use the password view in the user form to specify a temporary password, such as a literal string that is the same for everyone, or is the same as the Identity Manager account ID. See *Assigning User Forms* on page 3-10 and the chapter titled *Identity Manager Views* for more information.

For bulk actions, and Load from File, you can specify password values in the CSV file. These should be considered temporary passwords that users must change.

Policies establish limitations for Identity Manager accounts, and are categorized as:

- Identity Manager account policies -- Use these to establish user, password, and authentication policy options. Identity Manager account policies are assigned to organizations or users.
- Resource password and account ID policies -- Use these to set or select length rules, character type rules, and allowed words and attribute values.

Make sure you make any updates to the default policies before you begin loading account information into Identity Manager.

The following table lists the policies provided with Identity Manager as well as the default settings.

Policy Name	Default Characteristics
AccountId Policy	Account IDs must have a minimum length of 4 characters and a maximum length of 16 characters.
Default Lighthouse Account Policy	Sets the account ID and password policies to **AccountId Policy**, and **Password Policy**. Passwords are generated by Identity Manager, rather than by users.
Password Policy	Passwords must have a minimum length of 4 characters and a maximum length of 16 characters.The password cannot contain the user's e-mail, first name, last name, or full name.
Windows 2000 Password Policy	Passwords must have a minimum length of 6 characters. Passwords must have 3 of the following characteristics: • 1 numeric character • 1 uppercase letter • 1 lowercase letter • 1 special character In addition, the password cannot contain the account ID.

See the "Identity Manager Users" chapter in *Identity Manager Administration* for more information about account and password policies.

Creating a Data Loading Account

It is recommended that you create a separate administrator account to perform data loading for the following reasons:

- Data loading actions can be tracked more easily when auditing is enabled.
- Every Identity Manager administrator is assigned a user form that creates and edits Identity Manager users. When you create an account for data loading, you can specify a streamlined form that runs quicker than the default forms. See the next section for information.

See the *Identity Manager Administration* for more information about creating accounts.

Assigning User Forms

In the context of data loading, user forms are used to perform background processing. For example, forms can work in conjunction with resource adapters to process information from an external resource before storing it in the Identity Manager repository. They can also be used to place users in the correct Organization based on input user data.

The user view is a data structure that contains all available information about an Identity Manager user. It includes:

- Attributes stored in the Identity Manager repository
- Attributes fetched from resource accounts
- Information derived from other sources such as resources, roles, and organizations

Views contain many attributes, and a view attribute is a named value within the view (for example, `waveset.accountId` is the attribute in the user view whose value is the Identity Manager account name).

Most form field names are associated with a view attribute. You associate a field with a view attribute by specifying the name of the view attribute as the name of the form field. For more information on the user view, including a reference for all attributes in the user view, see the chapter titled *Views*.

The following fields are often in a user form that loads users.

- firstname
- lastname
- fullname
- email
- waveset.accountId
- waveset.organization
- EmployeeId

The waveset.accountId and waveset.organization are values specific to Identity Manager. The EmployeeId attribute is a customized attribute. Its use is illustrated in *Defining Custom Correlation Keys* on page 3-13.

Identity Manager provides numerous forms that are pre-loaded into the system. Additional forms are also available in the `$WSHOME/sample/forms` directory. Many of the forms in this directory are resource-specific. You might wish to review these forms with the Business Process Editor (BPE) to determine whether they should be used in production.

To increase performance during bulk operations, the user form assigned to an administrator should be as simple as possible. If you want to create a form for data loading, then you can remove code that is designed to display data. Another example of simplifying the form would be if you use bulk add actions. Your CSV file could define basic attributes such as firstname and lastname. These attributes could then be removed from the administrator's user form. See the chapter titled *Forms* for more information about creating and editing forms.

Note Do not directly modify a form provided with Identity Manager. Instead, you should make a copy of the form, give it a unique name, and edit the renamed copy. This will prevent your customized copy from being overwritten during upgrades and service pack updates.

Linking to Accounts on Other Resources

Identity Manager uses correlation and confirmation rules to link accounts. A correlation rule looks for Identity Manager users that might own an account. It returns a list of users that match the criteria defined in the correlation rule. A confirmation rule tests a Identity Manager user against an account to determine whether the user actually does own the account. It returns true or false values. This two-stage approach allows Identity Manager to optimize correlation, by quickly finding possible owners (based on name or other attributes), and by performing expensive checks only on the possible owners.

Before you begin using correlation and confirmation rules, you must be familiar with the data that is present from the first data load. The Identity Manager accountId will always be present. If you performed a Load from File or a Create bulk action, then the values in the heading row of the CSV file are also present. If you performed a Load from Resource or reconciliation, some key attributes found on the resource will be present, but others will be present only if they are explicitly saved.

In addition, you must be familiar with the account data stored on the secondary resources as well. Ideally, a secondary resource contains data that overlaps with data that has already present in Identity Manager.

This can be more difficult than it sounds. Different resources often have varying requirements for user accounts. As an example, the following table compares the requirements and restrictions for a Windows account name and a Solaris account name.

Characteristic	Windows	Solaris
Maximum length	20 characters	8 characters
Special characters permitted	All but " / \ [ ] : ; | = , + * ? < >	period (.), underscore (_), and hyphen (-) only
Able to specify a full name	Yes	There is one comment field. Traditionally, the comment field lists the user's full name, but other information could be included.
Able to specify additional comments, such as employee ID?	Yes	

The differences between Windows and Solaris account names highlight some of the difficulties in linking accounts:

- Because of the differences in maximum length, the account names will usually be different on these two systems. On Windows, users often have an account name that matches their first and last name. On Solaris, account names might be the concatenation of the first letter of the first name plus the first seven digits of the last name. Therefore, a correlation rule that compares account IDs will probably not be enough to link a Solaris account to a Windows account.
- To create a user account on Windows, the administrator must supply a display name. Solaris user accounts do not require display names, although the optional GECOS field can be used to specify a user's name. There are no guidelines about the contents or format of this field. A user's GECOS field might be blank or contain text unrelated to the user's Windows display name. Therefore, a correlation rule that compares full names might not trigger as often as you would expect.

Consider the following questions as you prepare to link accounts:

- Do you have users with similar names, such as Mary A. Jones vs. Mary B. Jones, or John Doe Jr. vs. John Doe III? If so, how will you distinguish between them?
- Were account names on each resource defined in a consistent manner? If yes, then it will be easier to define a rule that compares an account ID on a resource with a value stored in Identity Manager.
- Did users have the ability to edit resource account data? If yes, then the data might not match values that are stored on a system that users cannot change.

Defining Custom Correlation Keys

A rule cannot compare an account value on a resource with an Identity Manager value unless the value is stored in the system. The `accounts[Lighthouse]` attribute stores many of these values, but additional values must be added with the Extended User Attributes Configuration object. The system does not save attributes that are not registered in the configuration object.

By default, the following attributes are included as extended user attributes:

- firstname
- lastname
- fullname

Note The fullname extended user attribute must be added to the list of QueryableAttrNames.

If you want to use a different attribute, such as an employee ID as part of a correlation rule, then you must add it to the User Extended Attributes configuration object. Use the following steps to do this:

1. Access the Identity Manager debug page at `http://`*`PathToIDM`*`/debug` . The System Settings page is displayed.
2. Select Configuration from the **List Objects** pull-down menu. The **List Objects of type: Configuration** page is displayed.
3. Select the **edit** link for **User Extended Attributes**.
4. Add the new attributes to the List element, for example:

   ```
   <String>EmployeeId</String>
   ```

   The attribute must be defined as an Identity Manager attribute on the Account Attributes (schema map) page for the resource.
5. Save your changes. Identity Manager returns to the System Settings debug page.

The custom attribute must also be added to the QueryableAttrNames element in the UserUIConfig configuration object.

1. Select Configuration from the **List Objects** pull-down menu. The **List Objects of type: Configuration** page is displayed.
2. Select the **edit** link for **UserUIConfig**.
3. Add the new attributes to the <QueryableAttrNames><List> element, for example:

   ```
   <String>EmployeeId</String>
   ```
4. Save your changes. Identity Manager returns to the System Settings debug page.
5. Restart your application server.

Creating Custom Rules

Identity Manager predefines a number of correlation and confirmation rules in `sample/reconRules.xml`. You can use these as a basis for your own rules. Rules must be assigned a subtype of `SUBTYPE_ACCOUNT_CORRELATION_RULE` or `SUBTYPE_ACCOUNT_CONFIRMATION_RULE`.

The following rule compares the `account.EmployeeId` attribute, which is defined on the secondary resource, with the `EmployeeId` attribute that was previously loaded into Identity Manager. If the secondary resource has an `account.EmployeeId` value, then the correlation rule returns a list of users that match the `EmployeeId`.

```
<Rule subtype='SUBTYPE_ACCOUNT_CORRELATION_RULE' name='Correlate
Employee IDs'
   <cond>
      <ref>account.EmployeeId</ref>
      <list>
         <new class='com.waveset.object.AttributeCondition'>
            <s>EmployeeId</s>
            <s>equals</s>
            <ref>account.EmployeeId</ref>
         </new>
      </list>
   </cond>
</Rule>
```

In this example, the EmployeeId attribute has been previously added to the User Extended Attributes and UserUIConfig configuration objects. If this attribute has was not included as a default Identity Manager attribute name for the resource, it must also be added or edited on the schema map for the resource.

Correlation rules return a list of possible matches. If the results are expected to return only one match, such as an employee ID, then no confirmation rule would be needed. However, if there could be multiple matches, which could be the case if correlation found accounts that matched by first and last name, then a confirmation rule would be needed to further identify the match.

Rules can be added to Identity Manager by using the Business Process Editor, importing an XML file, or editing and renaming an existing rule via the debug page.

Manually Linking Accounts

Identity Manager provides several mechanisms that can be used to assign accounts when correlation and confirmation rules do not find a match.

Using the Account Index

The Account Index records the last known state of each resource account known to Identity Manager. It is primarily maintained by reconciliation, but other Identity Manager functions will also update the Account Index, as needed.

Note Load from resource, load from file, and bulk actions do not update the Account Index.

To view the account index, click the Resources tab, then click the Account Index link on the left. Then navigate to a resource to display the status of all accounts on that resource.

When you right-click on an uncorrelated account (represented in the Account Index table with a situation of “UNMATCHED” and an Owner of “_UNKNOWN_”), Identity Manager displays a menu that presents you with the options of creating a new Identity Manager user account, running reconciliation on a single account using the reconciliation policy in effect for the resource, specifying an owner, or deleting or disabling the resource account. If you select the “Specify Owner” option, Identity Manager displays a screen that allows you to search for owners that might criteria that you specify. Refer to the *Identity Manager Administration* for more information.

Enabling Self-Discovery

The Identity Manager User Interface can be configured to allow Identity Manager users to discover their own resource accounts. This means that a user with an Identity Manager identity can associate it with an existing, but unassociated, resource account. Self-discovery can be enabled only on resources that support pass-through authentication.

To enable self-discovery, you must edit the End User Resources configuration object, and add to it the name of each resource on which the user will be allowed to discover accounts. To do this:

1. Access the Identity Manager debug page at `http://PathToIDM/debug` . The System Settings page is displayed.
2. Select Configuration from the **List Objects** pull-down menu. The **List Objects of type: Configuration** page is displayed.
3. Select the **edit** link for **End User Resources**.
4. After the <List> element, add `<String>Resource</String>`, where *`Resource`* matches the name of a resource object in the repository. For example, to allow users to self-discover their accounts on resources `NT` and `Solaris`, edit the <List> element as follows:

```
<List>
   <String>NT</String>
   <String>Solaris</String>
</List>
```

5. Save your changes. Identity Manager returns to the System Settings debug page.

When self-discovery is enabled, the user is presented with a new menu item on the Identity Manager User Interface (**Inform Identity Manager of Other Accounts**) This area allows him to select a resource from an available list, and then enter the resource account ID and password to link the account with his Identity Manager identity.

Example Scenarios

This section provides scenarios that illustrate the process of loading accounts from one or more resources. The following scenarios discuss issues that might arise in your environment.

- Active Directory, SecurID, and Solaris
- LDAP, PeopleSoft, and Remedy
- Expedited Bulk Add

Active Directory, SecurID, and Solaris

A company wants to use Identity Manager to manage Active Directory, SecurID, and Solaris accounts. All workers have an Active Directory account, and most employees have a SecurID account. Only a fraction of employees of have a Solaris account. After examining the account data on each resource, the Identity Manager administrator has determined the following attributes can be used as correlation keys:

Possible Correlation Keys	Active Directory	SecurID	Solaris
Account ID matches AD	N/A	Yes	No
Employee ID	Yes	No	No
Full name	Yes	Yes	Yes (Description attribute)

Because all employees have an Active Directory account, it will be used as the first data loading resource. SecurID will be loaded second, because the account IDs on this resource match those on Active Directory. Account IDs are always unique, therefore this is a better correlation key than full name. The Active Directory and SecurID accounts are expected to correlate without problems.

Correlating the Solaris accounts will be difficult. The only correlation attribute that exists on Solaris accounts is the user's full name. Solaris does not have individual attributes for defining first name and last name. As a result, the correlation rule will be a comparison of the string defined in the Solaris useradd -c command with the fullname value in Active Directory. The comparison will often fail, due to factors such as use of nicknames or extraneous spaces and punctuation.

Example Users

In this scenario, the following users demonstrate some of the possible problems you might encounter when loading accounts:

Worker name	AD and SecurID Logon Name	AD Full Name	Solaris Account Name	Solaris Description
Anthony Harris	AJ Harris	Anthony J Harris	ajharris	A.J. Harris
Isabelle Moreno	Isabelle Moreno	Isabelle Moreno	imoreno	Isabelle Moreno
John Thomas (Sr.)	John Thomas	John Thomas	jthomas	John Thomas
John Thomas (Jr.)	John P. Thomas	John P. Thomas	jthomas2	John Thomas
Robert Blinn	Robert Blinn	Bob Blinn	rblinn	Bob Blinn
Theodore Benjamin	Theodore Benjamin	Theodore Benjamin	tbenjami	Ted Benjamin

Loading Active Directory Accounts

Use the following steps as a guideline for using reconciliation to load Active Directory accounts into Identity Manager.

1. From the Resources page in the Administrative interface, select the Windows 2000/ Active Directory resource from the New Resource pull-down menu. Then configure the adapter.

   Make sure you do not delete the accountId or fullname Identity Manager user attribute from schema map. Also make sure the identity template is correct. See the online helps and the *Identity Manager Resources Reference* for more information about configuring the adapter.
2. (Optional) Edit the account and password policies as desired. See *Setting Account ID and Password Policies* on page 3-8 for more information.
3. (Optional) Create a user form that will be used for reconciliation. See *Assigning User Forms* on page 3-10 for more information.

4. (Optional) Create an Identity Manager user for performing data loading. Assign the user form created in the previous step to the user.
5. Configure the reconciliation policy for the resource. On the first resource, the correlation rule is not important, and the confirmation rule is not used when creating Identity Manager users. Since this is the first resource, you probably want to assign the UNMATCHED situation to the value “Create new Identity Manager user based on resource account.”
6. If you created a user to perform data loading, log in as that user. This step is not necessary for reconciliation, but would be for Load from File, Load from Resource, or Bulk actions.
7. Reconcile the Active Directory resource.

Results

If you used the default Identity Manager account policy and default Active Directory identity template, Identity Manager will not create an Identity Manager user that links to Theodore Benjamin’s Active Directory account, because his name contains more than 16 characters. For this example, the account ID policy was set to 25 characters.

Identity Manager creates user accounts for all resource accounts with a situation status of CONFIRMED. This should include all users that passed the password and account ID policies. Unless your user form specified otherwise, the Identity Manager account name will be the same as Active Directory login name.

Loading SecurID Accounts

When SecurID is implemented, SecurID user records are usually imported from a Microsoft Security Accounts Manager (SAM) database or from an LDAP server. As a result, the SecurID account IDs match those from the source. This makes correlating users a relatively simple task, because there is a one-to-one correlation between SecurID and Active Directory accounts. The User Name Matches Account ID correlation rule can be used to quickly link these accounts.

To load SecurID accounts, perform the procedure described in *Loading Active Directory Accounts*, with the following modifications:

- When you are configuring the SecurID adapter, ensure that you do not delete the accountId Identity Manager user attribute.
- Configure the reconciliation policy as follows:
  - Set the correlation rule to “User Name Matches Account ID.”

- Since Active Directory is considered to be an authoritative source, and SecurID relies on Active Directory account information, you might want to set the UNMATCHED situation option to “Delete Resource Account” or “Disable Resource Account.” The UNASSIGNED situation should be set to “Link resource account to Identity Manager user.”

Results

All SecurID accounts should correlate with the Active Directory account. Perform any additional steps to resolve UNMATCHED or DISPUTED situations.

Loading Solaris Accounts

In this scenario, the fullname attribute is the only correlation key. This is a weak correlation key, because differences in spacing and punctuation guarantee matches will fail. In addition, users can change their display names with the Solaris **chfn** command. Even if full names once matched, they might not agree if any users have run the **chfn** command.

By default, the fullname attribute is not queryable. To enable this feature, you must edit the UserUIConfig configuration object, and add the fullname attribute to the <QueryableAttrNames><List> element. See *Defining Custom Correlation Keys* on page 3-13 for more information.

You will also need to create a custom rule to correlate fullname attributes. The following example, which is named “Correlate Full Names” performs the correlation. It compares the value of the `account.Description` attribute from the Solaris resource to the `fullname` attribute, a system attribute that was populated from Active Directory.

```
<Rule subtype='SUBTYPE_ACCOUNT_CORRELATION_RULE' name='Correlate Full
Names'
   <cond>
      <ref>account.Description</ref>
      <list>
         <new class='com.waveset.object.AttributeCondition'>
            <s>fullname</s>
            <s>equals</s>
            <ref>account.Description</ref>
         </new>
      </list>
   </cond>
</Rule>
```

This rule compares the Description attribute from the Solaris resource with the Identity Manager fullname attribute. If the two attributes match, the accounts are correlated, with a situation of CONFIRMED.

To load Solaris accounts, perform the procedure described in *Loading Active Directory Accounts*, with the following modifications:

- When you are configuring the Solaris adapter, ensure that you do not delete the accountId or Description Identity Manager user attribute.
- Configure the reconciliation policy as follows:
  - Set the correlation rule to “Correlate Full Names” (the example rule).
  - There could be numerous Solaris accounts that do not correlate with the accounts already loaded into Identity Manager. Set the UNASSIGNED situation to “Link resource account to Identity Manager user”. In most cases, you should set the UNMATCHED situation to “Do nothing”. Deleting or disabling unmatched users could result with a loss of data or productivity.

Results

The following table lists the users in this scenario.

Worker name	AD Full Name	Solaris Account Name	Solaris Description
Anthony Harris	Anthony J Harris	ajharris	A.J. Harris
Isabelle Moreno	Isabelle Moreno	imoreno	Isabelle Moreno
John Thomas (Sr.)	John Thomas	jthomas	John Thomas
John Thomas (Jr.)	John P. Thomas	jthomas2	John Thomas
Robert Blinn	Bob Blinn	rblinn	Bob Blinn
Theodore Benjamin	Theodore Benjamin	tbenjami	Ted Benjamin

In this example, we can expect that only accounts for Isabelle Moreno will correlate.

- The accounts for Anthony Harris, John Thomas (Jr.), Robert Blinn, and Theodore Benjamin will not correlate because the Active Directory fullname attributes do not exactly match the Solaris Description attributes. These accounts will have a situation of UNMATCHED. In this scenario, the Solaris account names are based on first initial plus last name. With the exception of the John Thomas account, assigning these unmatched Solaris accounts will be easy.
- The Solaris accounts jthomas and jthomas2 will have a situation of DISPUTED. Both of these accounts have a Description value of John Thomas. You must find another means to determine which user Solaris accounts jthomas and

jthomas2 belong to. Ideally, you could use a confirmation rule to distinguish between the two accounts. However, Solaris and Active Directory accounts do not contain enough intersecting attributes to create a confirmation rule.

LDAP, PeopleSoft, and Remedy

In this scenario, the LDAP or PeopleSoft resource could theoretically be the primary resource.

- If all employees and contractors are tracked in PeopleSoft, then this application can be considered an authoritative resource.
- If all employees have LDAP accounts, then LDAP could be considered an authoritative resource.

Remedy is not a candidate to be the primary resource, because only a small percentage of works have a Remedy account.

In many cases, if you have multiple authoritative resources, then any of those resources can be loaded first. However, the PeopleSoft Component adapter performs ActiveSync functions only. (There is another PeopleSoft adapter available, but it is limited in scope.) The PeopleSoft Component adapter does not perform reconciliation, and as a result, reconciliation policy cannot be set for the resource. There are no correlation rules available, so the PeopleSoft accounts must be loaded first. The Identity Manager account names will match PeopleSoft EMPLID (employee ID) values.

The PeopleSoft employee ID is ideal as a correlation key, because it is unique for all users defined in the system. The LDAP inetOrgPerson object contains an employeeNumber attribute. An employee ID could also be stored in an attribute with a label such as Description, or in a custom attribute. This scenario assumes the employeeNumber LDAP attribute is in use.

The Remedy adapter does not provide default attributes. You must customize the adapter to fit your environment. Because the Remedy application is often configured to send email when a request enters the system, we'll assume the e-mail attribute is available. The LDAP inetOrgPerson object also contains the mail attribute. Therefore, the e-mail address will be the correlation key.

The following table lists the correlation keys for each resource in this scenario.

Possible Correlation Keys	PeopleSoft	LDAP	Remedy
Employee ID	Yes	Yes	No
E-mail address	No	Yes	Yes

Example Users

In this scenario, the following users demonstrate some of the possible problems you might encounter when loading accounts:

Worker name	PeopleSoft EMPLID	LDAP Employee Number	LDAP Email (@example.com)	Remedy Email (@example.com)
Robert Blinn	945	945	Bob.Blinn	bblinn
William Cady	None	None	William.Cady	William.Cady
Eric D'Angelo	1096	1096	Eric.D'Angelo	Eric.D'Angelo
Renée LeBec	891	None	None	None
Josie Smith	1436	1463	Josie.Smith	None
John Thomas	509	509	John.Thomas	None
John P. Thomas	None	None	John.P.Thomas	John.P.Thomas

Loading PeopleSoft Users

Use the following steps as a guideline for using reconciliation to load PeopleSoft accounts using ActiveSync into Identity Manager.

1. From the Resources page in the Administrative interface, select the PeopleSoft Component resource from the New Resource pull-down menu. If this resource is not displayed, click the Configure Managed Resources button and add `com.waveset.adapter.PeopleSoftComponentActiveSyncAdapter` as a custom resource. This adapter requires the installation of a JAR file provided by PeopleSoft. See the *Identity Manager Resources Reference* for more information.

   Then configure the adapter. Make sure you do not delete the accountId or fullname Identity Manager user attribute from schema map. Also make sure the identity template is correct. See the online helps and the *Identity Manager Technical Reference* for more information about configuring the adapter.
2. (Optional) Edit the account and password policies as desired. See *Setting Account ID and Password Policies* on page 3-8 for more information.
3. (Optional) Create a user form that will be used for data loading. The `$WSHOME/sample/forms/PeopleSoftForm.xml` file can be used as a foundation. See *Assigning User Forms* on page 3-10 for more information.
4. Start ActiveSync on the PeopleSoft adapter.

Results

Identity Manager loads all users unless the user form indicates that an account should not be loaded. In this scenario, Renée LeBec does not have an LDAP or Remedy account. Presumably, she no longer works for the company. If you used the default PeopleSoft form, then Identity Manager disables PeopleSoft accounts for terminated employees.

The accounts for William Cady and John P. Thomas are not created because they are not defined within PeopleSoft.

Loading LDAP Users

In this scenario, the `employeeNumber` attribute in the LDAP inetOrgPerson object is the correlation key. This attribute is not listed by default in the schema map for the LDAP adapter, so you must add it manually.For this example, add the attribute `EmployeeId` to the Identity Manager User Attribute side of the schema map, and `employeeNumber` to the Resource User Attribute side.

Note The PeopleSoft adapter uses the Identity Manager attribute name `EmployeeId` by default. This value was chosen to maintain consistency between LDAP and PeopleSoft, although this is not required.

The e-mail address will be the correlation key for the Remedy resource, but it must be set-up and configured before you load LDAP accounts. The inetOrgPerson object contains the `mail` attribute, which will be the correlation key for loading Remedy accounts. The `mail` attribute also must be added to the schema map. Add the `email` attribute to the Identity Manager User Attribute side of the schema map, and `mail` to the Resource User Attribute side. `email` is a predefined Identity Manager attribute, so it is easier to user this attribute, rather than editing the User Extended Attributes or UserUIConfig configuration objects to include a `mail` attribute.

Identity Manager stores account IDs in the User object in the attribute `resourceAccountIds`. This is a multi-valued attribute, with each value taking the form *`accountId`*`@`*`objectId`*. You can create a rule that will compare the EmployeeId value from LDAP to the PeopleSoft accountId using the following rule:

```
<Rule subtype='SUBTYPE_ACCOUNT_CORRELATION_RULE' name='Correlate
EmployeeId with accountId'>
   <cond>
      <ref>account.EmployeeId</ref>
      <list>
         <new class='com.waveset.object.AttributeCondition'>
            <s>resourceAccountIds</s>
            <s>startsWith</s>
            <concat>
               <ref>account.EmployeeId</ref>
```

```
                <s>@</s>
              </concat>
            </new>
          </list>
        </cond>
</Rule>
```

Note In this scenario, it is not necessary to add attributes to the User Extended Attributes or UserUIConfig configuration objects, because the accountId and email attributes are always available to the system.

To load LDAP accounts, perform the following procedure:

1. From the Resources page in the Administrative interface, select the LDAP resource from the New Resource pull-down menu. Then configure the adapter as follows:
   - Add the EmployeeId and email Identity Manager User attributes.
   - Make sure you do not delete the accountId Identity Manager user attribute from schema map.
   - Also make sure the identity template is correct.

     See the online helps and the *Identity Manager Resources Reference* for more information about configuring the adapter.
2. Configure the reconciliation policy for the resource as follows.
   - Set the Correlation Rule to Correlate EmployeeId with accountId.
   - Set the following situation values:
     - Set the UNASSIGNED situation to “Link resource account to Identity Manager user”.
     - Set the UNMATCHED situation to an appropriate action. You might need to discuss with the PeopleSoft administrator about the possibility of adding users who are discovered on other resources. If you select the “Create new Identity Manager user based on resource account” option, the Identity Manager user will have, by default, an account name based on the LDAP `cn` attribute.
3. Reconcile the LDAP resource.

Results

In this scenario, accounts for William Cady, Josie Smith, and John P. Thomas will be in the UNMATCHED state. For Josie Smith, the employee ID values on the PeopleSoft and LDAP resources do not match. Because employee IDs are generated by PeopleSoft, then LDAP value is incorrect. Correct the mistake and reconcile again.

William Cady and John P. Thomas are not defined in PeopleSoft. As mentioned in step 2 of loading LDAP account procedure, you should consider whether the accounts need to be added to PeopleSoft.

Loading Remedy Users

The Remedy adapter does not have predefined account attributes. You must add these attributes to the schema map. Remedy uses integers to uniquely identify each attribute that it tracks. For example, the Remedy account ID might be assigned a value such as 1002000100. These Remedy attribute numbers must be added as Resource User Attributes on the schema map. At minimum, you must add the following Identity Manager User attributes:

- accountId
- email (the correlation key)

The USER_EMAIL_MATCHES_ACCOUNT_EMAIL_CORR correlation rule will link the Remedy accounts to the Identity Manager accounts.

To load Remedy accounts, perform the following procedure:

1. From the Resources page in the Administrative interface, select the Remedy resource from the New Resource pull-down menu. Then configure the adapter as follows:
   - At minimum, add the accountId and email Identity Manager User attributes. Other attributes can also be added.

     See the online helps and the *Identity Manager Resources Reference* for more information about configuring the adapter.
2. Configure the reconciliation policy for the resource as follows.
   - Set the Correlation Rule to USER_EMAIL_MATCHES_ACCOUNT_EMAIL_CORR.
   - Set the following situation values:
     - Set the UNASSIGNED situation to “Link resource account to Identity Manager user”.
     - Set the UNMATCHED situation to an appropriate action.
3. Reconcile the Remedy resource.

Results

The Remedy accounts for William Cady, Eric D'Angelo, and John P. Thomas correlated successfully because the email addresses defined in LDAP and Remedy matched. The Remedy account for Robert Blinn did not correlate. The e-mail address on Remedy was an alias. The other users in this scenario do not have Remedy accounts.

Expedited Bulk Add Scenario

The following procedure illustrates how a few users can be quickly added to Identity Manager using Create actions. Any resources referenced in the CSV file must be defined within Identity Manager before the CSV file is loaded into the system.

1. Generate a CSV file that contains information needed to perform a Create bulk action. See *Create Bulk Actions* on page 3-7 for more information about the format of the CSV file.
2. If your CSV file does not contain passwords, set the default Password Policy Options so that passwords are generated by the system. To do this, click the **Configure** tab, then **Policies** on the left. Click the **Default Lighthouse Account** Policy link. Then select the Generated option from **Password Provided by** drop-menu.
3. Create a new provisioning task based on the default Create User provisioning task.

   From the XML view, in the `TaskDefinition` tag, edit the value of the `resultLimit` parameter to 0. This value determines the number of seconds the results of the task is to be retained.

   Then delete the following sections from the task:

   ```
   <Activity id='1' name='Approve'>
   ...
   </Activity>

   <Activity id='3' name='Notify'>
   ...
   </Activity>
   ```

   Note The activity calling to the provisioner must remain, or users will not be created properly.

   Be sure to rename the task, to a value such as Fast Create User.
4. Create a new user form based on the default user form.

   From the XML view, replace the entire <Form>... </Form> structure with the following:

```
<Form help='account/modify-help.xml'>
   <Field name="viewOptions.Process">
      <Expansion>
         <s>Fast Create User</s>
      </Expansion>
   </Field>
</Form>
```

Be sure to rename the user form, to a value such as Fast User Form.

5. Create a Identity Manager account that will be used to load accounts into Identity Manager. Assign the user form created in Step 4 to this user.
6. Log in to Identity Manager using the account created in the previous step.
7. Run the Create bulk action.

4 Configuring User Actions

This appendix details how to add custom tasks to the Identity Manager Administrator Interface and configure user actions that you can execute from two areas of the interface:

- User Account Search Results page
- User applet on the Accounts page

Adding Custom Tasks

Follow these general steps to add custom tasks:

- Set up authorization for the task
- Add the task to the repository

Setting Up Custom Task Authorization

Typically, you set authorization for custom tasks to restrict access to the task to a certain set of administrators. To set up authorization:

1. Add a new authorization type (AuthType) to the repository for the task
2. Create a new AdminGroup (capability) for the task
3. Grant the new capability to one or more administrators

Step 1: Create an AuthType

The new authorization type you create should extend the existing `TaskDefinition`, `TaskInstance`, and `TaskTemplate` AuthTypes. To add the authorization type, edit the Authorization Types Configuration object in the repository and add a new authorization type element for your task.

Use the `<AuthType>` element to create a new authorization type. This element has one required property: `name`. The example below displays the correct syntax for an `<AuthType>` element.

After creating the authorization type, you must edit the Authorization Types Configuration object in the repository, and add the new `<AuthType>` element.

The following example shows how to add a custom task to move multiple users into a new organization.

Example

```
<Configuration name='AuthorizationTypes'>
      <Extension>
        <AuthTypes>
          <AuthType name='Move User'
           extends='TaskDefinition,TaskInstance,TaskTemplate'/>
        </AuthTypes>
      </Extension>
</Configuration>
```

Step 2: Create an AdminGroup

Next, create an AdminGroup that grants `Right.VIEW` for the newly created AuthType. To do this, you must create an XML file with the new administrator group, and then import it into the Identity Manager repository.

```
<?xml version='1.0' encoding='UTF-8'?>
    <!DOCTYPE Waveset PUBLIC 'waveset.dtd' 'waveset.dtd'>
    <Waveset>
    <AdminGroup name='Move User'
                protected='true'
                displayName='UI_ADMINGROUP_MOVE_USER'
                description='UI_ADMINGROUP_MOVE_USER_DESCRIPTION'>
      <Permissions>
        <Permission type='Move User' rights='View'/>
      </Permissions>
      <MemberObjectGroups>
        <ObjectRef type='ObjectGroup' id='#ID#All' name='All'/>
      </MemberObjectGroups>
    </AdminGroup>
    </Waveset>
```

Note The `displayName` and `description` attributes are message catalog keys. If these are not found in a message catalog, they are displayed as they are found in the attributes. If message catalog keys are used, you must add the messages either into `WPMessages.properties` or a custom message catalog.

Step 3: Grant Capabilities to Administrators

Finally, you must grant administrators access to execute the newly defined task. You can accomplish this in one of two ways:

- Directly assign the new capability, or
- Add the new capability to an Admin Role (either directly or by using a capabilities rule), and then assign it.

Adding a Task to the Repository

After you set up task authorization, you can add the task to the repository. The task is a typical TaskDefinition that can be defined through the Business Process Editor (BPE) or imported as XML. For example, a task to change the organization for multiple users would resemble the following example (which is included in the `samples` directory).

Example

```
<?xml version='1.0' encoding='UTF-8'?>
  <!DOCTYPE TaskDefinition PUBLIC 'waveset.dtd' 'waveset.dtd'>
  <!--  MemberObjectGroups="#ID#Top" authType="Move User" name="Change
Organizations"
        taskType="Workflow" visibility="runschedule"-->
  <TaskDefinition authType='MoveUser'
                 name='Change Organizations' taskType='Workflow'
                 executor='com.waveset.workflow.WorkflowExecutor'
                          suspendable='true'
                 syncControlAllowed='true' execMode='sync'
                execLimit='0' resultLimit='0'
                 resultOption='delete' visibility='runschedule'
                progressInterval='0'>
    <Form name='Change Organization Form'
          title='Change Organization Form'>
      <Display class='EditForm'/>
      <Include>
        <ObjectRef type='UserForm' name='User Library'/>
        <ObjectRef type='UserForm' name='Organization Library'/>
      </Include>

      <FieldRef name='namesList'/>
      <FieldRef name='orgsList'/>
      <FieldRef name='waveset.organization'/>
    </Form>
    <Extension>
      <WFProcess name='Change Organizations'
                 title='Change Organizations'>
```

```
        <Variable name='waveset.organization'/>
        <Variable name='userObjectIds' input='true'>
          <Comments>The names of the accounts to change the
                    organization on.</Comments>
        </Variable>

        <Activity id='0' name='start'>
          <ReportTitle>
            <s>start</s>
          </ReportTitle>
          <Transition to='Process Org Moves'/>
        </Activity>
<Activity id='1' name='Process Org Moves'>
          <Action id='0' process='Move User'>
            <Iterate for='currentAccount' in='userObjectIds'/>
            <Argument name='userId' value='$(currentAccount)'/>
            <Argument name='organizationId'
                      value='$(waveset.organization)'/>
          </Action>
          <Transition to='end'/>
        </Activity>

        <Activity id='2' name='end'/>
      </WFProcess>
    </Extension>
    <MemberObjectGroups>
      <ObjectRef type='ObjectGroup' id='#ID#Top' name='Top'/>
    </MemberObjectGroups>
  </TaskDefinition>
```

About the Example

Note these features of the preceding example:

- The task's `authType` attribute is set to Move User. This will restrict access to this task to users that are assigned the capability to execute this AuthType.
- The form contains FieldRefs to `namesList` and `orgsList`. These fields are defined in the User Library and Organization Library, respectively. Including these fields will display lists of the names of all selected users and all selected organizations. *For potentially dangerous tasks, you should include one or both of these fields so the user is aware of the potential effects of running the task*.
- The task has an input variable named `userObjectIds`. This variable contains a list of the names or IDs of the users selected in the User Account Search Results page or in the user applet on the Accounts page. Iterate over this variable to perform the desired action on all selected users.

The following table lists the variables that are available for input to the task.

Variable	Description
userObjectIds	List of IDs of the selected users. Available from the User Account Search Results and Accounts pages. When invoked from the User Account Search Results page, this list contains the names of the selected users.
userNames	List of names of the selected users. Available from the User Account Search Results and Accounts pages.
orgObjectIds	A List of IDs of the selected organizations. Available only from the Accounts page.
orgNames	A List of names of the selected organizations. Available only from the Accounts page.

To enable this workflow, you must also add to the repository a sub-process to change a user's organization, as shown in the following example.

Example

```
<?xml version='1.0' encoding='UTF-8'?>
  <!DOCTYPE Configuration PUBLIC 'waveset.dtd' 'waveset.dtd'>
  <!--  MemberObjectGroups="#ID#Top" configType="WFProcess" name="Move
User"-->
  <Configuration name='Move User' createDate='1083353996807'>
    <Extension>
      <WFProcess name='Move User' title='Move User'>
        <Variable name='userId' input='true'>
          <Comments>The accountId of the user to move.</Comments>
        </Variable>
        <Variable name='organizationId' input='true'>
          <Comments>The ID of the organization to move the user
                    into.</Comments>
        </Variable>

        <Activity id='0' name='Start'>
          <Transition to='Update Organization'/>
        </Activity>

        <Activity id='1' name='Update Organization'>
          <Action id='0' process='Update User View'>
            <Argument name='accountId' value='$(userId)'/>
            <Argument name='updates'>
              <map>
                <s>waveset.organization</s>
                <ref>organizationId</ref>
```

```
            </map>
          </Argument>
        </Action>
        <Transition to='End'/>
      </Activity>

      <Activity id='2' name='End'/>
    </WFProcess>
  </Extension>
  <MemberObjectGroups>
    <ObjectRef type='ObjectGroup' id='#ID#Top' name='Top'/>
  </MemberObjectGroups>
</Configuration>
```

Configuring User Actions

You must configure definitions for the buttons and actions menu selections that initiate custom actions. Definitions for the buttons and actions menu items that appear on the User Account Search Results and Accounts pages are contained in the User Actions Configuration configuration object.

You should not directly edit the User Actions Configuration object. Rather, best practice for configuring user actions is to:

- Copy the User Actions Configuration configuration object into a new configuration object.
- Modify the SystemConfiguration to point to the new configuration object.

Configure User Actions

In general, to configure user actions, you should:

- Copy the User Actions Configuration configuration object into a new XML file.
- Change the name of the new object to My User Actions Configuration.
- Make any desired modifications to My User Actions Configuration.
- Import the XML file into Identity Manager from the Import Exchange File page
- Modify SystemConfiguration to change the `userActionsConfigMapping` attribute's value to My User Actions Configuration

The configuration object consists of these configuration sections.

Attribute	Description
findUsersButtons	Contains a list of button definitions for the Administrator Interface User Account Search Results page.
userApplet.userMenu	Contains a list of menu item definitions for the user actions menu. This menu displays when you right-click a user in the applet on the Administrator Interface Accounts page.
userApplet.organizationMenu	Contains a list of menu item definitions for the organization actions menu. This menu displays when you right-click an organization in the applet on the Accounts page.

Each section contains a list of user actions to display in the interface. The button and menu configuration items have the same basic properties. Both include several extensions unique to the interface.

Example: Adding Change Organization Task to Each List

The following excerpt is an example of the user action configuration customized to add the Change Organization task to each list.

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE Waveset PUBLIC 'waveset.dtd' 'waveset.dtd'>
<Waveset>

<Configuration name='My User Actions Configuration'>
  <Extension>
    <Object>
      <!-- Buttons for the find users results page. -->
      <Attribute name='findUsersButtons'>
        <List>
          <Object>
            <Attribute name='textKey'        value='UI_NEW_LABEL' />
            <Attribute name='commandName'    value='New' />
            <Attribute name='requiredPermission'>
              <Object>
                <Attribute name='objectType' value='User' />
                <Attribute name='rights'     value='Create' />
              </Object>
            </Attribute>
            <Attribute name='alwaysDisplay' value='true' />
          </Object>
          ...
          <Object>
```

```
            <Attribute name='textKey'
value='UI_CHANGE_ORGANIZATIONS_LABEL' />
            <Attribute name='commandName'
                                value='Change Organizations' />
          </Object>
        </List>
      </Attribute>
      <Attribute name='userApplet'>
        <Object>
          <!-- The menu to display when a user is selected. -->
          <Attribute name='userMenu'>
            <List>
              <Object>
                <Attribute name='textKey'
                          value='UI_ACCT_JAVA_MENU_NEW_ORG' />
                <Attribute name='commandName'
                          value='New Organization' />
                <Attribute name='requiredPermission'>
                  <Object>
                   <Attribute name='objectType' value='ObjectGroup' />
                    <Attribute name='rights'     value='Create' />
                  </Object>
                </Attribute>
              </Object>
              ...
              <Object>
                <Attribute name='separator' value='separator' />
              </Object>
              <Object>
                <Attribute name='textKey'
                         value='UI_CHANGE_ORGANIZATIONS_MENU_LABEL' />
                <Attribute name='commandName'
                          value='Change Organizations' />
              </Object>
            </List>
          </Attribute>
         <!-- The menu to display when an organization is selected. -->
          <Attribute name='organizationMenu'>
            <List>
              <Object>
                <Attribute name='textKey'
                          value='UI_ACCT_JAVA_MENU_NEW_JUNCTION' />
                <Attribute name='commandName'
                          value='New Directory Junction' />
                <Attribute name='requiredPermission'>
                  <Object>
                   <Attribute name='objectType' value='ObjectGroup' />
                    <Attribute name='rights'     value='Create' />
                  </Object>
```

```
                </Attribute>
                <Attribute name='orgTypes' value='normal,dynamic' />
              </Object>
              ...
              <Object>
                <Attribute name='separator' value='separator' />
              </Object>
              <Object>
                <Attribute name='textKey'
                    value='UI_CHANGE_ORGANIZATIONS_MENU_LABEL' />
                <Attribute name='commandName'
                    value='Change Organizations' />
              </Object>
            </List>
          </Attribute>
        </Object>
      </Attribute>
    </Object>
  </Extension>
  <MemberObjectGroups>
    <ObjectRef type='ObjectGroup' name='All'/>
  </MemberObjectGroups>
</Configuration>
</Waveset>
```

User action definitions support these core attributes.

Attribute	Description
`textKey`	Message catalog key for the text of the button or menu item.
`commandName`	Name of the command to execute. This can be a command that is natively supported (such as New or Delete User), or the name of a TaskDefinition to execute.

`requiredPermission.objectType`	Type of object that the rights are required on in order to display this item. This is applicable only for natively supported commands. TaskDefinitions should use AuthTypes for controlling access.
`requiredPermission.rights`	Comma-separated list of `Right` names required on the specified objectType to display this item. This is applicable only for natively supported commands. TaskDefinitions should use AuthTypes for controlling access.
`alwaysDisplay`	Optional. Specifies whether to always display this button. If set to a value of `true`, the button is displayed even if user search returns no results. The default value for this attribute is `false`. Applies to `findUsersButtons` section only.

User actions definitions in the userApplet section also support these attributes:

Attribute	Description
`orgTypes`	Comma-separated list of organization types for which to display the item in the organization menu. Possible values are normal, dynamic, and virtual for normal organizations, dynamic organizations, and virtual organizations, respectively. If this attribute is not specified, the menu item is displayed for all organization types.
`separator`	Special item in the format `<Object><Attribute name='separator' value='separator'/></Object>`. Separators are displayed as horizontal bars in the Administrator Interface menus, and cannot be selected.

5 Private Labeling of Identity Manager

This chapter identifies the basic components you will need to rebrand the Identity Manager interface to match your company's intranet or corporate style guidelines. *Private labeling* is the customization of the interface to meet these corporate guidelines.

Private Labeling Tasks

There are three general categories of private labeling tasks:

- **Changing default header and footer content by incorporating** your corporate logo, changing default text, and altering colors in both the User and Administrator interfaces.
- **Changing display fonts and component colors** throughout the application through the use of a style sheet located in `styles\customStyle.css`.
- **Changing Identity Manager behavior on commonly used pages** by editing the System Configuration object. These tasks, which include disabling the **Forgot Your Password?** button, are frequently performed by users while rebranding the product interface.

Architectural Features

Private labeling includes editing the components listed in the following table.

Component	Interface
`$WSHOME/styles/`**customStyle.css**	Administrator and User
`$WSHOME/WEB-INF/lib/idmcommon.jar`	Administrator and User
`$WSHOME/user/`**userFooter_beforeFirstTableRowTag.jsp**	User Interface
`$WSHOME/user/`**userFooter_beforeEndBodyTag.jsp**	User Interface
`$WSHOME/user/`**userFooter_beforeLastEndTableRowTag.jsp**	User Interface

Component	Interface
`$WSHOME/includes/`**bodyEnd_beforeFirstTableRowTag.jsp**	Administrator Interface
`$WSHOME/includes/`**bodyEnd_beforeEndBodyTag.jsp**	Administrator Interface
`$WSHOME/includes/`**bodyEnd_beforeLastEndTableRowTag.jsp**	Administrator Interface
`$WSHOME/`**index_quickLinks.jsp**	Administrator and User

Style Sheets

Two *style sheets* affect the display characteristics of text in the product interface:

- `style.css` — Defines the display attributes of pages throughout both interfaces. This file also controls the images contained in the headers and footers
- `customStyle.css` — Contains any changes to the default settings contained in `style.css`. Settings in this file override the settings in `style.css`.

Default Text

Default text occurs throughout the product interface in the following:

- form titles, subtitles, buttons, odd and even rows, section heads
- general text
- warning messages
- navigation button text, including both available and selected navigation buttons
- table header/body text

Text Attributes

Display attributes include

- title — font-family, font-size, font-weight, color
- button — text-alignment, background-color
- text — same as title, text-decoration, white-space
- flyover help — vertical-align and text-align

Default Style Settings

The `$WSHOME/styles/style.css` file contains default style settings. Do not edit this file.

Customized File

The `customStyle.css` file contains customizations and is not overwritten during product upgrades. Settings defined there will override the default settings in style.css. *Include all your customizations in customStyle.css*.

JSP Files

Several JSP files contain the default settings for headers and footers: `userFooter.jsp`, `userHeader.jsp`, `bodyEnd.jsp`, and `bodyStart.jsp`. Do not edit these files. Instead, to preserve your customizations during product upgrade, edit only the JSPs listed in *Architectural Features*.

WPMessages_en.properties File

The `$WSHOME/WEB-INF/lib/idmcommon.jar` file contains the message catalog entries that you can extract into a `WPMessages_en.properties` file for editing.

Customizing Headers and Footers

Customization tasks are identical for both interfaces, although you must edit different files.

Note Avoid editing any .jsp file other than the specified files. If you must edit one, first back up the .jsp to a safe location before copying, editing, and renaming it.

Changing Header Appearance

The most typical labeling tasks involve

- Changing the image referenced in the header section of the page from the default Sun logo to corporate standards. Replace or remove images by editing `customStyle.css`.
- Suppressing the Identity Manager logo
- Using corporate internal look and feel guidelines, specifically borders, header, and background colors

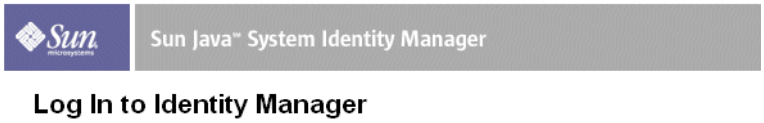


Figure 1. Default Identity Manager bodyStart.jsp/userHeader.jsp as displayed in the browser

Changing Footer Appearance

Most typical changes can be done by editing the various files identified below.

Typical labeling tasks are described in the following table.

Change	Edit this file...
Add space above the existing footer to incorporate space for comments	`.../user/userFooter_beforeFirstTableRowTag.jsp`
Add Row under the footer for a copyright or legal disclaimer	`.../user/userFooter_beforeEndBodyTag.jsp`
Add Table cell to the end of the footer to link to a security policy	`.../user/userFooter_beforeLastEndTableRowTag.jsp`
Change “Logged in as ..” text	You cannot change this through .jsps. You can edit the custom message catalog. See *Extracting the WPMessages_en.properties File* later in this chapter.

Figure 2. Default Identity Manager bodyEnd.jsp as displayed in the browser.

Customizing the Login Page

Typical customizations to the Login page include:

- Adding a list of quick links
- Changing the default login text
- Changing page title and subtitle

Adding a List of Quick Links

A typical customization to the user home page involves adding a custom list of links to tasks or resources that users frequently access in your environment. These quick links offer a shortcut through the product interface to frequent destinations.

To add a list of quick links

1. Add links to the list in `index_quickLinks.jsp`.
2. Uncomment the list section by removing the surrounding <%-- and --%> tags.
3. Save the file. You do not need to restart your application server.

Changing the Default "Logged in as .." Text

1. Extract the file `WPMessages_en.properties` from `WEB-INF/lib/idmcommon.jar` to the config directory (for English).
2. Edit the file by changing the value for UI_NAV_FOOT_LOG_AS to the string you want displayed. The user name will appear where the {0} notation is.
3. Restart your application server.

Extracting the WPMessages_en.properties File

To extract the `WPMessages_en.properties` from the `$WSHOME/WEB-INF/lib/idmcommon.jar`, follow one of these two procedures:

On a machine running a UNIX operating system:

1. **`cd $WSHOME/WEB-INF/lib`**
2. Enter **`jar -xf idmcommon.jar com/waveset/msgcat/WPMessages_en.properties`**
3. Enter **`mv com/waveset/msgcat/WPMessages_en.properties $WSHOME/config`**
4. Enter **`rm -r com`**
5. **`cd $WSHOME/config`**

On a machine running a Windows operating system:

1. **`cd %WSHOME%/WEB-INF/lib`**
2. Enter **`jar -xf idmcommon.jar com/waveset/msgcat/WPMessages_en.properties`**
3. Enter **`move com\waveset\msgcat\WPMessages_en.properties %WSHOME%\config`**
4. Enter **`rmdir -r com`**

5. **cd %WSHOME%\config**

Changing Page Title and Subtitle

To change the default Login page title and subtitle and welcome message, edit the following entries in the config/`WPMessages_en.properties` file:

- UI_LOGIN_TITLE
- UI_LOGIN_TITLE_TO_RESOURCE
- UI_LOGIN_WELCOME2

To change this default text, follow the procedure for extracting and editing the `WPMessages_en.properties` file detailed in *Changing the Default "Logged in as ..." Text*.

Default WPMessages_en.properties Settings

```
UI_LOGIN_TITLE=Log In
UI_LOGIN_TITLE_TO_RESOURCE=Log In to <b>{0}</b>
UI_LOGIN_IN_PROGRESS_TITLE=Log In (In Progress)
UI_LOGIN_CHALLENGE=Enter Your {0} Password
UI_LOGIN_CHALLENGE_INFO=You are required to enter the password you
logged into
[PRODUCT_NAME] with before the requested action can be completed.
UI_LOGIN_TITLE_LONG=[PRODUCT_NAME] LogIn
UI_LOGIN_WELCOME=Welcome to the Sun Java&#8482; System [PRODUCT_NAME]
system.
Enter the requested information, and then click <b>Login</b>.
UI_LOGIN_WELCOME2=Welcome to the Sun Java&#8482; System [PRODUCT_NAME]
system.
Enter your user ID and password, and then click <b>Login</b>. If you
can't remember your password, click <b>Forgot Your Password?</b>
```

Changing Font Characteristics

Display attributes typically specify the following basic font display characteristics:

family	for example, Helvetica or Arial
size	specified in point size (for example, 14 pt)
weight	unspecified indicates normal weight. When specified, typically bold
color	typically specified as black (title -- font-family, font-size, font-weight, color

Certain components can be further defined by additional characteristics. For example, buttons can be defined with a background color, and the alignment of the text and button label.

Editing Font Characteristics

To edit, copy from `styles.css` and paste into `customStyle.css`. Then, modify the selected setting in `customStyle.css`.

Example

The following entry represents the default settings for each page title:

```
.title {
    font-family: Arial, Helvetica, sans-serif;
    font-size: 16pt;
    font-weight: bold;
    color: C;
```

Sample Labeling Exercises

The following example illustrates how to suppress the Identity Manager logo and reference a custom image in the header section of the page.

- Changing the product name
- Changing header and footer bar colors
- Changing navigation tab colors
- Changing Identity Manager behavior on commonly used pages

Changing the Product Name

To change the product name displayed, edit the `config/WPMessages_en.properties` file.

1. If you have not already, extract the file `WPMessages_en.properties` from `WEB-INF/lib/idmcommon.jar` to the config directory.
2. Edit the value for the `PRODUCT_NAME` key to your project name:

   `PRODUCT_NAME=`***Flex IT***

3. (Optional) Append your company or division name as displayed on the home page by modifying the value of `UI_HOME_TITLE`

   `UI_HOME_TITLE=Acme [`***`PRODUCT_NAME`***`]`

Note You may want to change more references in `WPMessages_en.properties` from 'Sun Java™ System' to ***Acme***.

4. For these changes to take effect, stop and restart the server.

Replacing the Identity Manager Logo with a Custom Logo

To change the logo in the Administrator or User interfaces, copy the following snippets from `styles/style.css` into `customStyle.css` and replace the Identity Manager logo image with your .gif file:

```
td.admin_header_image {
   background-image:url(images/Acme.gif”);
   background-repeat; no-repeat;
   height: 54px;
   width: 299px;
}

td.user_header_image {
   background-image: url("../images/acme.gif");
   background-repeat: no-repeat;
   height: 52px;
   width: 380px;
}
```

If you want the image to repeat across the screen, change `no-repeat` to `repeat` and increase the width to the desired length.

Note For best results, logo .gifs should be between 50 and 60 pixels high.

Changing Header and Footer Bar Colors

To change the look-and-feel of Identity Manager, edit the `styles/customStyle.css` file.

Copy the .header and .footer sections and set the background-color on both to an appropriate color (in this case, yellow).

```
.header {
    background-color: #FFFF40;
}
```

```
.footer {
    background-color: #FFFF40;
}
```

Changing Navigation Tab Colors

Copy the snippet for the navigation links across the top of the page and down the sides. Change the background-color to an appropriate color (in this case, orange).

```
.availablenavbutton {
    text-align: left;
    background-color: #FFA620;
}
```

Many other options can be customized following the same procedures. Text style and size, alignment, and the colors and configurations of various other objects can be modified following the same steps.

Note To see the changes without bouncing your server or browser, perform a Ctrl-refresh on a page.

Changing Identity Manager Behavior on Commonly Used Pages

To customize Identity Manager behavior on commonly used pages, you can alter settings in the System Configuration object.

Customizing with System Configuration Object

You can customize many commonly altered properties of the User or Administrator interfaces can by editing the System Configuration object. The attribute `<Attribute name='ui'>` and its subobjects control the product interface. Modifying the attributes under this attribute can change the behavior and of Identity Manager.

Miscellaneous Modifications: Admin Section of File

The admin section of System Configuration object file contains several attributes that are related to the Administrator Interface.

- To disable the **Forgot Your Password?** button on the Administrator login page, set `disableForgotPassword` to true.
- Setting `supressHostName` to true will suppress the display of the hostname for processes on the Task Details page.

```
<Attribute name='admin'>
   <Object>
      <Attribute name='disableForgotPassword'>
         <Boolean>false</Boolean>
      </Attribute>
      <Attribute name='workflowResults'>
         <Object>
            <Attribute name='suppressHostName'>
               <Boolean>false</Boolean>
            </Attribute>
         </Object>
      </Attribute>
   </Object>
</Attribute>
```

Miscellaneous Changes: User Section of the File

The user section of the System Configuration object file includes options for the User Interface.

- Disable the **Forgot Your Password?** button by setting `disableForgotPassword` to `true`.
- Modify the user login page title and welcome message with the corresponding entries in `WPMessages_en.properties` file in /config.

The `workflowResults` attribute contains attributes for customizing the display of workflows for nonadministrative users, as indicated below:

Attribute	Description
`anonSuppressReports`	Controls whether the workflow diagram is displayed in the anonymous user workflow status pages (`anonProcessStatus.jsp`).
`suppressHostName`	Controls whether the hostname is included in workflow status pages for end-users (`processStatus.jsp`).
`suppressReports`	Controls whether workflow diagrams is displayed to all non-anonymous users (`processStatus.jsp`).

```
<Attribute name='user'>
   <Object>
     <Attribute name='disableForgotPassword'>
       <Boolean>false</Boolean>
     </Attribute>
```

```
     <Attribute name='workflowResults'>
       <Object>
          <Attribute name='anonSuppressReports'>
             <Boolean>false</Boolean>
          </Attribute>
          <Attribute name='suppressHostName'>
             <Boolean>false</Boolean>
          </Attribute>
          <Attribute name='suppressReports'>
             <Boolean>false</Boolean>
         </Attribute>
        </Object>
      </Attribute>
   </Object>
</Attribute>
```

- To block the display of password and authentication question answers, set `obfuscateAnswers` to `true`. This setting causes answers to be displayed as asterisks in both the Administrator Interface and User Interface.

```
<Attribute name="obfuscateAnswers">
   <Boolean>true</Boolean>
</Attribute>
```

A Editing Configuration Objects

This chapter introduces an Identity Manager component called *configuration objects*. Editing configuration object properties is one way of implementing persistent changes to Identity Manager behavior.

About Configuration Objects

Configuration objects are store persistent customizations to Identity Manager. They are cached object types, which means that all configuration objects are brought into memory, and the cache is subsequently flushed, whenever a configuration object is changed.

See the Identity Manager Architectural Overview chapter for a discussion of the Identity Manager object architecture and how configuration objects interact with other Identity Manager components.

Viewing and Editing Configuration Objects

Use the Business Process Editor to view configuration and generic objects. You can access these miscellaneous configuration objects from the BPE under the Configuration Object category.

1. From the Business Process Editor main window, select **File > Open Repository Object** from the menu bar.

Tip You can also use the Ctrl-O shortcut.

2. If prompted, enter the Identity Manager Configurator name and password in the login dialog, and then click **Login**. The Select objects to edit dialog displays.
3. Double-click an object type to display all the objects that you have permission to view for that type.
4. Select a process or object, and then click **OK**. The Object window for the selected object displays, providing the following object views (tabs): Main, Repository, and XML.

For more information on using the Business Process Editor (BPE), see Introduction to the Business Process Editor in *Identity Manager Deployment Tools*

UserUIConfig Object

The UserUIConfig object controls Identity Manager User and Administrator Interface displays for account searching and editing, as well as internal system functions.

Configure this object at deployment time to improve performance of Identity Manager.

Use this object to:

- Control the columns that are displayed in the Accounts applet and Find Results pages (`SummaryAttrNames`)
- Define the attributes that users can search on within your identity deployment -- that is, the *queryable* attributes (`QueryableAttrNames`)
- Defines the attributes are stored in a separate column on the `userobj` table for optimal searching (`RepoIndexAttrs`)

Note This object controls system behavior at a fundamental level. Editing this object has widespread effects on system performance. Edit cautiously.

Viewing and Editing this Object

You can view this object, along with other configuration and generic system objects, using the Business Process Editor (BPE). For information on using the BPE to access this object, see *Introduction to the Business Process Editor* in *Identity Manager Deployment Tools*.

Refreshing Users

If you add or delete attributes from the `SummaryAttrName`, `QueryableAttrName`, and `RepoIndexAttrs` sections of this object, you must update all users by subsequently refreshing them as follows:

After editing this object, you must run a `refreshType` import command on all user objects for the summary attributes to be available. If many users must be refreshed, this can be a time-consuming process.

You can importing a file as follows:

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE Waveset PUBLIC 'waveset.dtd' 'waveset.dtd'>
<Waveset>
   <ImportCommand type='refreshType' targetType='User' />
</Waveset>
```

Attribute Types

Summary attributes expose information that users can retrieve using the list command. You can configure these attributes through the `SummaryAttrTypes` and `SummaryAttrNames` sections. To include an attribute in the Find Results columns or the applet list columns display, include it as a summary attribute in this section.

Queryable attributes define the attributes that users can search on within Identity Manager. These attributes are defined in the `QueryableAttrNames` section of this object.

Inline Queryable attributes are stored in the main table (`userobj`) rather than the associated table (`userattr`). These attributes must be single-valued. Querying on inline attributes is much faster than accessing the associated attribute table. These attributes are contained in the `RepoIndexAttr` section of this object.

Object Attributes

The attributes described here comprise a subset of the default UserUIConfig object attributes. The attributes you see in your deployment may vary.

SummaryAttrNames

Attributes that are members of `SummaryAttrNames` object are designated as *summary attributes*. Identity Manager displays summary attributes in product list results. These attributes must be a superset of the `AppletColumns` and Find Results lists, but do not need to be included in the `QueryableAttrNames` list.

When editing this object, do not remove the MemberObjectGroups attribute. This attribute is used for fast authorizations.

The following attributes are the default summary attributes provided by Identity Manager. `name` and `id` are built-in summary attributes and are not described here. You can add attributes to this list.

role

Identifies the Identity Manager roles. Role IDs are separated by a vertical bar (|).

res

Lists resource names separated by a comma. If the number of elements in this list exceeds the value of `SummaryAttrResourceCountLimit`, this list is truncated, and Identity Manager appends an ellipses (...).

prov

Specifies the provisioning level. This determines how many Identity Manager-assigned resources have been provisioned on the resource. (0 = none, 1 = some, 2 = all)

dis

(Boolean) Indicates whether the user disabled.

MemberObjectGroups

Specifies the organization that this member belongs to. Do not remove this attribute.

fullname, lastname, firstname

Specifies the user's `fullname`, `lastname`, and `firstname` attribute, respectively.

QueryableAttrNames

Specifies the attributes that users can search on in Identity Manager. You can add attributes to this list.

Default queryable attributes include:

- `correlationKey`
- `role`
- `email`
- `firstname`
- `lastname`
- `prov`
- `dis`
- `name`
- `locked`
- `user_resources`

AppletColumns

Specifies the names of the columns to be displayed on the List Accounts page. Edit this list to change the contents of the columns that the List Accounts page displays. Columns named in this list must be included in `SummaryAttrNames` (or the values will show up blank in the product interface). The list consists of GenericObjects for each column. Supported attributes are:

- `width` — (Valid for applet implementation only) Specifies the initial width of the column. If omitted (or zero), the applet assigns a default initial width to the column.
- `sortBy` — (Valid for applet implementation only) If present, identifies the column the applet will sort by initially. If more than one column is designated, the left column is used.
- `label` — (Valid for both applet and treetable implementations) Specifies the message key to use for the localized column name.

ShowListCache

Indicates whether to show the **Clear List Cache** button on the List Accounts page.

TemporarySummaryAttrResourceCountLimit

Specifies the number of resources in the resource summary. An excessive number will trigger a resource schema violation. The default value is 3.

PolicyAccountAttributeNames

Defines the attributes that appear in pop ups for an accountId policy. These attributes must match attributes that can be found on the User object. You can add any attribute that is included in a user form. Default values include `email`, `firstname`, `fullname`, and `lastname`.

PolicyPasswordAttributeNames

Defines the attributes that appear in pop ups for a password policy. These attributes must match attributes that can be found on the User object.You can add any attribute that is included in a user form. Default values include `accountId`, `email`, `firstname`, `fullname`, and `lastname`.

PolicyOtherAttributeNames

Defines the attributes that appear in pop ups for other policy types. These attributes must match attributes that can be found on the User object. You can add any attribute that is included in a user form. Default values include `accountId`, `email`, `firstname`, `fullname`, and `lastname`

PolicySpecialChars

Specifies the characters to include in the string quality policy for forced exclusion or inclusion. Password and account ID policies allow specifying rules about maximum number of special characters and minimum number of special characters.

TaskBarPages

Defines the paths of the JSP pages in the Identity Manager Administrator interface for which to display the task bar at the bottom of the page. For example, when you create a new user in Identity Manager, you see Create User at the bottom of the accounts page. This is because accounts/list.jsp is included in the TaskBarPages element by default.

RepoIndexAttrs

Defines the attributes that are copied into the waveset.userobj table and indexed to facilitate searching. Any attribute named here must also be queryable.

Edit this object to enhance search performance. This list can contain only five attributes, and these map to the ATTR1-ATTR5 database columns. By default, IDM indexes `firstname`, `lastname`, and `MemberObjectGroups`. You can add an additional two attributes for fast searching, however. For example, if your deployment contains the extended attribute departmentNumber, you could add it here, ensuring that it is included in all repository searches. If you know that you will not need `firstname`, `lastname`, or `MemberObjectGroups`, you can replace these attributes with other attributes.

Note If you don't index a queryable attribute, you can still use it in a search, but the performance may be much slower depending upon how many users are in the database and how many people are running searches simultaneously.

B Enabling Internationalization

This document provides information on configuring Identity Manager to use multiple languages or display a language other than English.

Architectural Overview

Components

File	Description
`WPMessages.properties`	Default message file. Located in `$WSHOME/idm/web/WEB-INF/classes/com/waveset/msgcat`. Shipped as part of the idmcommon.jar file. Displays message text in English and loads by default unless you've customized your IDM installation to behave otherwise.
`Waveset.properties`	Located in $WSHOME/config. Edit to enable support for multiple languages. (Set `Internationalization.enabled` to `true`.)
System Configuration Object	Specify custom message catalog.
additional message file for each supported language	Additional supported languages each require its own message file.WPMessages_xx_XX.properties, where xx represents the language and XX represents the country. For example, WPMessages_en_US.properties contains messages in American English. Each international catalog has its own .jar.
`i18n.xml`	Contains multiple `<Object>` elements that define which languages an Identity Manager administrator or user can choose.

Additional notes:

- If you have loaded a new catalog in /config, the new catalog takes precedence over the default. This applies only to catalogs of the same name.
- If you have more than one message file, you can specify the catalog from which a message key is derived by specifying `catalogname:keyname`.

Typical Entry

Messages are contained in key/text pairs and contains three parts:

- A text string, or key, that is an identifier used by the code to retrieve data. This required component should not be translated. It is used in the product configuration, and acts as a placeholder for the translation.
- A equals ("=") sign separating the key and text. This is required.
- A string containing data that will be displayed when running the application. This is the translation, used in place of the key whenever the page is rendered in the browser.

Each line in the resource array contains two strings. Translate the second quoted string on each line.

Certain strings to be translated contain special codes for data that will be inserted into the string when it is displayed. For example, if you have the following string to translate:

```
UI_USER_CONNECT={0}, connected at 100 mbs
```

the rendered version could appear as `jfaux, connected at 100 mb`

Translations typically appear inside a browser, so it is appropriate to add HTML tags to format the string, as shown below:

_FM_ACCOUNT_ID_HELP=Account ID
Enter a name for this user. This field is required.

Enabling Support for Multiple Languages

To enable support for multiple language catalogs, follow these steps, which are described in created detail below:

- Download localized files.
- Edit the Waveset.properties file by setting `Internationalization.enabled` to `true`.
- Import, edit and load the .\sample\i18n.xml file
- Restart Identity Manager
- Choose a language at the login screen. Additionally, the default language should match the language selected in the user's browser.

Step One: Download and Install Localized Files

Before You Install

Perform the following tasks before you install localized files:

1. Install Identity Manager. See *Identity Manager Installation* for detailed installation procedures.
2. Make sure the following locales on the application server have been set to UTF-8.
   - Application server instance
   - Database
   - Java Virtual Machine (JVM)

   Refer to the documentation for these products for information about setting the locale.

Download Message Catalog Files

Download the appropriate msgcat jar and place it in WEB-INF/lib. The Identity Manager Image Server website provides the following ZIP files that contain localized product files and documentation.

File Name (.zip)	Language	Locale
IDM__5_0_l10n_de	German	de_DE
IDM__5_0_l10n_es	Spanish	es_ES
IDM__5_0_l10n_fr	French (France and Canada)	fr_FR
IDM__5_0_l10n_it	Italian	it_IT
IDM__5_0_l10n_ja	Japanese	ja_JP
IDM__5_0_l10n_ko	Korean	ko_KR
IDM__5_0_l10n_pt	Brazilian Portuguese	pt_BR
IDM__5_0_l10n_zh	Simplified Chinese	zh_CN
IDM__5_0_l10n_zh_TW	Traditional Chinese	zh_TW

Download the ZIP file to a temporary location. By default, the contents of the ZIP file are extracted to the *`FileName`*`\IDM_5_0_l10n` directory, where *`FileName`* matches the name of the downloaded file, minus the ZIP extension.

Zip File Contents

Every extracted ZIP file contains the following:

- A JAR file containing localized message catalogs, help files, and other essential files. The JAR file is named `IDM_5_0_l10n_Locale.jar`.
- Identity Manager Localization README

Most extracted ZIP files contain the following:

- A translated version of the Identity Manager 5.0 release notes
- A translated version of *Identity Manager Administration*

Additional translated publications might also be available.

Install Localized Files

Use the following steps to install localized files on your application server.

1. Copy the JAR file from the temporary location to the *IdentityManagerInstallation*`/WEB-INF/lib` directory.

Step Two: Edit the Waveset.Properties File

1. Open the *IdentityManagerInstallation*`/config/Waveset.properties` file with a text editor.
2. Change the `Internationalization.enabled` property to `true`.
3. Save your changes and close the file.
4. Either restart Identity Manager or select Debug --> Reload Properties for this change to take effect.

Step Three: Import, Edit and Load the <ApplicationDirectory>\sample\i18n.xml File

Configure this file to set which languages Identity Manager administrators and end-users can be displayed.

1. Open the *IdentityManagerInstallation*`/sample/i18n.xml` file with a text editor. The file contains multiple `<Object>` elements that define which languages an Identity Manager administrator or user can choose. Each `<Object>` element is similar to the following:

```
<Object name='en_US'>
   <Attribute name='lang' value='en'/>
```

```
    <Attribute name='cntry' value='US'/>
    <Attribute name='gif' value='images/f0-us.gif'/>
  </Object>
```

2. Add, edit, and/or delete `<Object>` elements until only the languages appropriate for your environment are present. If you need to add or edit `<Object>` elements, use the following table to determine the proper values for the object `name`, and `lang`, `cntry`, and `gif` values.

Language	name	lang	cntry	gif
Chinese (Simplified)	zh_CN	zh	CN	images/f0-cn.gif
Chinese (Traditional)	zh_TW	zh	TW	images/f0-cn.gif
English (U.S.)	en_US	en	US	images/f0-us.gif
French (Canada)	fr_CA	fr	CA	images/f0-ca.gif
French (France)	fr_FR	fr	FR	images/f0-fr.gif
German (Germany)	de_DE	de	DE	images/f0-de.gif
Italian (Italy)	it_IT	it	IT	images/f0-it.gif
Japanese (Japan)	ja_JP	ja	JP	images/f0-jp.gif
Korean (South Korea)	ko_KR	ko	KR	images/f0-kr.gif
Portuguese (Brazil)	pt_BR	pt	BR	images/f0-br.gif
Spanish (Spain)	es_ES	es	ES	images/f0-es.gif

   a. Save your changes and close the file.

3. Load the `i18n.xml` file
   a. Log in to the Identity Manager administrative interface.
   b. Click the **Configure** tab at the top of the page. Then click on the **Import Exchange File** subtab on the left.
   c. Specify the path or browse *`IdentityManagerInstallation`*`/sample/i18n.xml` file. Then click the **Import** button to load the file.

After completing these steps, log out of Identity Manager. Restart your application server. When you reload the Identity Manager login page, the flags and locales you specified in the `i18n.xml` file are displayed. Select the appropriate flag to view localized text.

C LDAP Deployment Scenario

The LDAP scenario illustrates how Identity Manager can be deployed in an environment where user accounts are defined on a LDAP resource. This scenario assumes that LDAP is the only resource to be managed. (Name of scenario 2) illustrates the process of deploying Identity Manager across multiple resource.

Features

The LDAP scenario contains about 50 XML files that are loaded into Identity Manager. These files create and configure the following types of objects:

- User Form
- Rule
- Account Policy
- Admin Role
- Role
- Task Definition and Schedule
- Task Template
- Login Application and Login Module Group
- LDAP Resource
- Email Template

In addition, the scenario also provides several JSP files as well as a sample LDIF file that can be loaded onto your directory server. The LDIF file defines several users, several of which are managers.

The most important features of the scenario include the following:

- Managers can edit account attribute data for their subordinates. Managers are not explicitly granted the capabilities needed to access the Administrator Interface. However, they can view a linked list of their subordinates from the User Interface, and from this list, managers can access the account attributes for their subordinates.
- When a user edits account attribute data from the User Interface, the user's manager must approve the change for it to take effect. This is optional and only works if configured.
- Users can request access to a resource. The list of available resources is generated dynamically. Any resources that are already assigned to the user, either directly or through a role, are not displayed.

- The answer to a default authentication question is derived from an LDAP attribute.

Prerequisites

The LDAP scenario illustrates the integration of Identity Manager with the following products:

- Sun Java System Directory Server 5 2005Q1
- Sun Java System Messaging Server 6 2005Q1

Other directory servers and e-mail servers will not work with this scenario unless you customize the provided scripts.

In addition, Directory Server and Messaging Server must use the same LDAP base context.

Loading and Configuring the Scenario

Perform the following steps to configure your environment:

Important! Do not load the scenario in an existing production environment. The scenario reconfigures several system attributes.

Step 1: Set Up Directory Server

1. Create a user named **IdM User** in the base context, such as `cn=IdM User,dc=example,dc=com`. This user will be used by the LDAP resource adapter to provision to the directory.
2. Change the access control on the base context (`dc=example,dc=com`) to allow **IdM User** to have full control (read, compare, search, write, delete, add).
3. Enable the changelog on the Directory Server. From the **Configuration** tab of the Directory Server administrative console, click **Data**, then click on the **Replication** tab on the right. Check the **Enable Changelog** check box. If you are using Directory Server 5.0 or higher, also enable the **Retro Changelog Plugin** in the administrative console.

   Then restart Directory Server.
4. Change the access control on `cn=changelog` to allow IdM User to read and search.
5. Import the `$WSHOME/sample/scenarios/scenario1/schema.ldif` file into Directory Server.

6. Import the `$WSHOME/sample/scenarios/scenario1/scenario1Data.ldif` file into Directory Server.

Step 2: Configure Messaging Server

Note The LDAP scenario has been tested using Schema 1 for the mail server. The scenario should work with Schema 1.5 (compatibility mode) and Schema 2.

Add or modify the following option in the *MessageServeroot*`/config/option.dat` file to ensure that the mail forwarding and autoreply features work:

```
DELIVERY_OPTIONS=*mailbox=$M%$\\$2I$_+$2S@ims-ms-daemon, \
    &members=*,                                          \
    *native=$M@native-daemon,                            \
    /hold=@hold-daemon:$A,                               \
    *unix=$M@native-daemon,                              \
    &file=+$F@native-daemon,                             \
    &@members_offline=*,                                 \
    program=$M%$P@pipe-daemon,                           \
    #forward=**,                                         \
    *^!autoreply=$M+$D@bitbucket
```

Step 3: Load the Scenario

Use the following procedure to load the LDAP scenario into Identity Manager.

1. Configure Identity Manager so that it can display and operate task templates. Edit the `$WSHOME/config/Waveset.properties` file and set `gui.enableTaskTemplateEditor=true`.
2. Login in to Identity Manager as Configurator.
3. From the Identity Manager menu bar, click **Configure**, and then click **Import Exchange File**.

Note The browser must be launched from the Identity Manager server or have access to the files contained in the `idm/sample` directory on the Identity Manager server.

4. Enter the name or browse to the `$WSHOME/sample/scenarios/scenario1/scenario1.xml` file, and then click **Import**.

   Identity Manager displays a message indicating that the file imported successfully.
5. Copy the contents of `$WSHOME/sample/scenarios/scenario1/jsp/` into `$WSHOME/user/`. These JSP files are necessary for enabling a manager to edit a subordinate's data and for users to request access to a resource.

Step 4: Configure the LDAP Resource Adapter

Loading the scenario creates a resource named LDAP Resource. This resource must be configured to match your environment.

Use the following procedure to minimally configure the LDAP Resource. Additional configuration might be required.

1. From the Identity Manager menu bar, click **Resource**.

   Resources are grouped by type, which are represented in the list by named folders. To expand the hierarchical view and see currently defined resources, double-click the resource type folder or click **+** (plus sign) next to the folder.
2. Right click on the LDAP Resource icon and select the Edit -> Resource Wizard option from the pop-up menu. The Resource Parameters page is displayed.
3. At minimum, configure the following fields so that they match your environment:
   - **Host**
   - **User DN**
   - **Password**
   - **Base Contexts**

   Note The User DN field defaults to cn=IdM User,dc=example,dc=com. This value should be changed to the user created in *Step 1: Set Up Directory Server*.

   Click **Next** to display the Account Attributes page.
4. No configuration is required at this point on the Account Attributes page. See *LDAP Resource Attributes* on page C-6 for more information about the account attributes.

   Click **Next** to display the Identity Template page.
5. The identity template defines account name syntax for users. Update the value for the identity template. For an LDAP resource, this value in the general format:

   `uid=$accountId$,ou=`*`OrganizationUnit`*`,`*`BaseContext`*

   Click **Save**.
6. For now, disable the ActiveSync feature of the adapter.To do this, right click on the LDAP Resource icon and select Edit -> Edit Active Sync Running Settings. by changing the value of the **Startup Type** parameter to Disabled.

   Also change the **Proxy Administrator** to Configurator or other administrator with the capabilities necessary for processing changes on the resource.

   Click **Save**.
7. Restart the application server in order to disable the ActiveSync functionality.

Step 5: Configure Email Templates

In the LDAP scenario, Identity Manager sends an email in the following circumstances:

- When a user or manager changes the value of an account attribute.
- When ActiveSync discovers a user on the LDAP resource.

Use the following procedure to configure the Identity Manager email templates.

1. From the menu bar, click **Configure**, and then click **Email Templates**.
2. Click the link for LDAP ActiveSync Account Creation Notification.
3. Change the values of the **SMTP Host** and **From** fields. Edit other fields as desired.
4. Click **Save**.
5. Repeat steps 2 through 4 for the Account Update Notification email template and any other template you wish to implement.

After you have loaded the LDAP scenario and made the necessary configurations, there are no specific steps that you must perform. However, it is recommended that you use the procedures listed in the remainder of this chapter to explore the capabilities provided in the scenario. Unless otherwise specified, all files mentioned in this scenario are loaded by default.

Sample Directory Overview

The `scenario1Data.ldif` file defines several default users, some of which are managers. The following table lists each user and indicates the user's manager and subordinates, if any.

User ID	Name	Manager	Subordinates
ah1340	Anthony Harris	John Thomas	Joseph Smith
js1260	Joseph Smith	Anthony Harris	Renee Lebec
tb926	Theodore Benjamin	Bill Cady	None
ed1096	Eric D 'Angelo	Bill Cady	None
rl891	Renee Lebec	Joseph Smith	Bill Cady
jt509	John Thomas	Josie Smith	Anthony Harris

User ID	Name	Manager	Subordinates
rb945	Robert Blinn	None	Josie Smith
js1436	Josie Smith	Robert Blinn	John Thomas
sp486	Stephen Perkins	Bill Cady	None
bc1165	Bill Cady	Renee Lebec	Theodore Benjamin Eric D'Angelo Stephen Perkins

To view end-user functions of this scenario, log in to the User Interface with the User ID listed in the table. The password for all users in this scenario is hello.

Creating and Configuring Identity Manager Objects

This section discusses how selected Identity Manager objects are created and configured as a result of loading the LDAP scenario. These objects are needed to support this scenario.

LDAP Resource Attributes

The users defined in the `scenario1Data.ldif` file contain attributes from standard object classes such as inetOrgPerson. Additional attributes are defined in Messaging Server object classes such as `inetmailuser`. If you are using the LDAP scenario with a mail server other than Messaging server, you must update the schema map for the LDAP resource.

The following table lists the account attributes that were added for the LDAP scenario. If you are using a directory server or e-mail server other than those supported for this scenario, you may need to make changes to the account attributes.

Identity Manager Attribute	LDAP Attribute	Description
manager	manager	Distinguished name of the user's manager.
dn	entrydn	The distinguished name for the user.
employeeNumber	employeenumber	Numerically identifies an employee within an organization
roomNumber	roomnumber	The user's office or room number.
building	extendedaddress	The user's office building name or code. This is not a standard LDAP attribute.
carLicense	carlicense	Vehicle license or registration plate
ldapGroups	ldapGroups	A string that contains a list of groups for the LDAP user.
email	mail	The user's primary email addresses.
alternateEmail	mailalternateaddress	One or more additional email addresses.
mailDeliveryOption	maildeliveryoption	One or more of `autoreply`, `hold`, `mailbox`, `native`, or `unix`.
mailHost	mailhost	The fully qualified host name of the MTA that is the final destination of messages sent to this recipient.
mailForwardingAddress	mailforwardingaddress	A forwarding address for inbound messages.
inetUserStatus	inetuserstatus	The status of a user's account with regard to global server access. Values are `active`, `inactive`, and `deleted`.
mailQuota	mailquota	The amount of disk space, in bytes, allowed for the user's mailbox.
mailAutoReplySubject	mailautoreplysubject	Subject text of auto-reply response.

Identity Manager Attribute	LDAP Attribute	Description
mailAutoReplyText	mailautoreplytext	Auto-reply text sent to all senders except users in the recipient's domain.
mailAutoReplyTextInternal	mailautoreplytextinternal	Auto-reply text sent to senders from the recipients domain.
vacationStartDate	vacationstartdate	Vacation start date and time in the format YYYYMMDDHHMMSSZ. The Z is literal.
vacationEndDate	vacationenddate	Vacation end date and time. Uses the same format as vacationStartDate.
mailAutoReplyMode	mailautoreplymode	Either `echo` or `reply`.

In addition, the following LDAP attributes are queried on Directory Server, but are not mapped to Identity Manager:

- ismanager. If this attribute is present and has the value of "Y", then the user is deemed a manager.
- manager. The value of this attribute is set to the dn of the record of the manager who manages that specific entry. Therefore, a subordinate has the manager attribute set to the dn of their manager.

Admin Roles

The `AdminRole-User.xml` file creates two admin roles:

The User admin role is defined in the `AdminRole-User.xml` file. This admin role assigns the User Account Administrator capability to the My Employees organization and allows a manager to edit the data of his or her subordinates. The User admin role also grants the Approver capability to the Top organization.

Roles

The Mail Users role assigns the LDAP resource to the user. The role is defined in the `Role-MailUser.xml` file.

If a user exists in the directory and it contains the `inetuser`, `ipuser`, `inetmailuser`, `inetlocalmailrecipient`, and `userpresenceprofile` object classes, the user will be assigned the Mail Users role when it is created in Identity Manager. If one or more of these object classes are missing, the role is not applied to the user.

When a user is created from Identity Manager and the Mail Users role is enabled, an email account will be created on the Messaging Server upon reconciliation. Since the user has a mail account, it will pass the rule Is Mail User (`Rule-IsMailUser.xml`).

The LDAP scenario assumes that there might be users defined in the directory that should not have mail accounts. If all users in the directory should have an email account, then do the following:

- Add the `inetuser`, `ipuser`, `inetmailuser`, `inetlocalmailrecipient`, and `userpresenceprofile` object classes to the list of managed object classes on the adapter.
- Change the properties of the Mail User role so that the `objectClass` resource account attribute is set to None.

Object Groups

The scenario creates the following object groups:

Users — The virtual organization in which reconciled users are placed. This organization is defined in the `ObjectGroup-Users.xml` file.

My Employees — This virtual organization is assigned in the User admin role. It invokes the Get My Subordinates rule. This organization is defined in the `ObjectGroup-MyEmployees.xml` file.

Login Application and Login Module Group

Identity Manager uses pass-through authentication to resolve the validity of the LDAP password for the end-user pages. The Administrative GUI uses the Identity Manager user's password. To enable this feature, the scenario uses the following files to create the LDAP Login Module Group for the User Interface.

- LoginApp-UserInterface.xml
- LoginModGroup-LDAPLoginModuleGroup.xml

General Configuration

The following files configure miscellaneous aspects of the LDAP scenario.

`Configuration-UserUIConfig.xml` — Adds the manager attribute to the SummaryAttrNames and QueryableAttrNames attributes so that administrators can search for users by specifying their manager's distinguished name.

`Configuration-UserExtendedAttributes.xml` — Adds manager to the view of the User object.

`Configuration-CustomMessageCatalog.xml` — Defines strings that are displayed in the on the user interfaces.

The following table lists the form and process map changes.

Process or Form?	Type	New Value	Original Value
Process	createUser	Create User Template	Create User
Process	updateUser	Update User Template	Update User
Form	endUserForm	LDAP End User Form	End User Form
Form	endUserMenu	Custom End User Menu	End User Menu
Form	userForm	Custom Tabbed User Form	Tabbed User Form

Creating User Accounts

Creating user accounts is one of the most important activities that an administrator performs. In the LDAP scenario, the majority of the user accounts are created when Identity Manager initially loads accounts from the directory server. However, other accounts will be loaded by ActiveSync, or created by an administrator in Identity Manager. This section describes the process of creating users with each of these mechanisms.

Using Reconciliation to Create Users

Before performing a reconciliation, the reconciliation policy must be configured, and a user form must be assigned to the user performing the reconciliation.

Reconciliation Policy

Reconciliation policy allows the administrator to select the server to run reconciliation, determine how often and when reconciliation takes place, and set responses to each situation encountered during reconciliation. You can also configure reconciliation to detect changes made natively (not made through Identity Manager) to account attributes.

Identity Manager provides a default global reconciliation policy, which can be inherited for any resource. However, the LDAP scenario requires some changes to the default policy, and these changes are defined in the `Configuration-ReconcileConfiguration.xml` file.

There are two significant changes to the default policy:

- The proxy administrator is set to Reconciliation Admin. When a reconciliation response is performed, it is performed as this administrator. The Reconciliation Admin has been configured to create users with the User Form with Questions user form. If you need to login as Reconciliation Admin, the password is reconadmin.

  If you want to use a different user to perform this function, assign the default user form to User Form with Questions. In addition, ensure the user has the Bulk Account Administrator capability.
- The UNMATCHED situation is configured to create a new user based on resource account. When Identity Manager detects an account on the resource that does not have a corresponding account on Identity Manager, it creates a new Identity Manager user.

User Form

The User Form used for reconciliation (User Form with Questions) is defined in the `UserForm-UserFormWithQuestions.xml` file. This user form is meant to be used for reconciliations and load-from-resource operations only.

The default user form contains overhead that is not necessary for loading account data. The User Form with Questions form has been trimmed down so that it defines the following:

- The user's global full name, which is determined by concatenating the user's first and last names
- The user's organization, which is set to Top:User
- The answer to the user's authentication question. (See *Setting Answers to Authentication Questions* on page C-21 for more information.)

Account information loaded from a resource is stored in a User object and can be accessed from the User view. As a result, detailed information about each user does not need to be referenced in the user form. The User Form with Question form contains minimal data to help speed the process of reconciliation. In a production environment with thousands of users, the reconciliation process can take hours. Therefore, eliminating all unnecessary fields and attributes speeds up the reconciliation process.

The full name and authentication answers remain in the user form because they are derived attributes. The organization specifies the Identity Manager virtual organization into which the user object should be placed.

Workflow

The LDAP scenario uses the standard Create User workflow when creating user accounts via reconciliation.

Testing Procedure

After the LDAP adapter has been configured, you can perform a reconciliation so that user data can be populated into Identity Manager.

Log in to Identity Manager as Configurator and initiate a full reconciliation by right-clicking on the LDAP resource and selecting the **Full Reconcile Now** option. When reconciliation is complete, the status of all reconciliation attempts is displayed.

Using ActiveSync to Create Users

ActiveSync is another mechanism for loading resource account data into Identity Manager. ActiveSync "listens" or polls for changes to a resource, detecting incremental changes in real time. Because ActiveSync is designed to detect changes, it should not be used to load account information into Identity Manager for the first time.

The LDAP scenario enables changes from LDAP to be propagated into Identity Manager. Once a new users are detected with the ActiveSync functionality, the new users can immediately log in to the end user pages to manage their accounts.

The scenario also creates a temporary password and emails it to the new user. This is necessary for two reasons. First, it is possible to create an account in LDAP without a password. Therefore, the user would not be able to log in to Identity Manager. Secondly, passwords are usually not readable from LDAP. This means that if the LDAP ActiveSync is to be the authoritative source of the user's password, Identity Manager will not be able to synchronize the passwords on other resources with the LDAP password. In order for the passwords to be consistent among all resources, a temporary password should be used, which the user will use to login and change all resource passwords.

User Form

The LDAP ActiveSync Form translates ActiveSync attributes into Identity Manager user attributes. A temporary password (change12345) is created in this form.

The viewOptions.Process field (listed below), causes the LDAP ActiveSync Create User and LDAP ActiveSync Update User workflows to run for creates and updates respectively. For other operations (such as delete), the normal user workflows will be run.

```
<Field name='viewOptions.Process'>
   <Comments>Use the optimized create and update workflows.</Comments>
   <Expansion>
      <switch>
         <ref>feedOp</ref>
         <case>
            <s>create</s>
            <s>LDAP ActiveSync Create User</s>
         </case>
         <case>
            <s>update</s>
            <s>LDAP ActiveSync Update User</s>
         </case>
         <case default='true'>
            <null/>
         </case>
      </switch>
   </Expansion>
</Field>
```

Workflow

The `TaskDefinition-LDAPActiveSyncCreateUser.xml` file defines the workflow to run when a user is created through ActiveSync. This workflow is run whenever a user is discovered on the resource.

This workflow has been reduced to two actions. The first action is to provision the account in Identity Manager. The workflow then sends an email notification that contains the temporary password (chagne12345) to the newly-created user.

Also, the resultLimit workflow attribute is set to 0. This also helps speed up the workflow in that the results in the repository.

Email Template

The `EmailTemplate-LDAPActiveSyncAccountCreationNotification.xml` file defines an email template that sends a temporary password to the newly created user.

Testing Procedure

Prerequisites

The LDAP resource must be configured for ActiveSync. In *Step 4: Configure the LDAP Resource Adapter* on page C-4, ActiveSync was disabled. Run the Active Sync Wizard by right clicking on the LDAP resource and selecting Edit > Active Sync Wizard.

1. On the Synchronization Mode page set the **Input Form** field to LDAP ActiveSync Form. Then click **Next**.
2. On the Active Sync Running Settings page:
   - Change the **Startup Type** to Automatic or Manual.
   - Set the **Proxy Administrator** to Configurator or other administrator with the capabilities necessary for processing changes on the resource.
   - Specify a directory to which a log file will be written in **Log File Path**.
   - Assign a value such as 5 minutes in the **Poll Every** field.

   Then click Next.
3. On the General Active Sync Settings page, set the Filter Changes By field to `cn=IdM User`. This will prevent changes that were provisioned into the resource from Identity Manager from being reprocessed as Active Sync events.

   Then click Save.

Restart Identity Manager to allow the changes to take effect.

Procedure

Use the following steps to test that Identity Manager creates a new user when an account is detected on an ActiveSync resource.

1. Create a new inetOrgPerson in the base context directory using the directory server's native tools. Do not use Identity Manager to create the user.
2. Wait for the amount of time specified in the Poll Every field until the ActiveSync changes are loaded into Identity Manager. In some circumstances, you might need to start ActiveSync.
3. In Identity Manager, click the Accounts tab. The user has been added to the Users organization.
4. Log in to the User Interface using the new user's account ID and temporary password.
5. Use the "Change Password" link to change the user's password.

The end result is that users created externally in LDAP are given accounts in Identity Manager. A temporary password is set on the LDAP resource, and the user is notified of the temporary password.

Using Identity Manager to Create Users

The LDAP scenario requires customized user forms, workflows, task templates, and notifications. These aspects are discussed in detail.

User Forms

The following user forms help create new Users

- Custom Tabbed User Form
- Custom Dynamic LDAP User Form

Custom Tabbed User Form

The LDAP scenario replaces the Tabbed User Form with the Custom Tabbed User Form. The standard Tabbed User Form controls the layout of the Create and Edit User pages. These pages have several navigation tabs that allow the administrator to set or view a user's identity, assignments, capabilities, and resource attributes. The Custom Tabbed User Form is essentially the same, except that it uses a field reference to "Resource Tabs" in the `UserForm-CustomDynamicUserForms.xml` file (Custom Dynamic User Forms) instead of MissingFields. This will generate an LDAP tab if the

user has been assigned an LDAP resource, either directly, or through a role. The Custom Dynamic User Forms in turn references the Custom Dynamic LDAP User Form.

To register a replacement Tabbed User Form, the userForm form mapping must be changed. The scenario automatically changes the userForm value to Custom Tabbed User Form.

Custom Dynamic LDAP User Form

The Custom Dynamic LDAP User Form controls the LDAP-specific attributes on the Create or Edit User page. It divides the LDAP-specific attributes into two sections: User Attributes and Mail Attributes.

In the User Attributes section, the Manager field contains a drop-down menu containing a list of managers. This list is populated by calling the Get Managers rule, which checks for accounts on the LDAP resource where the `ismanager` attribute is set to Y. The rule returns the `entrydn` attribute for each account that meets that criterion.

The Groups multi-select box is populated by the Get LDAP Groups rule. This rule queries the resource and returns a list of LDAP groups. If the account is already a member of an LDAP group, the group is placed in the Current Groups box.

On the Assignments navigation tab, if the Mail Users role is assigned to a user, then the **Mail Account Created** field contains a value of Yes. If the LDAP resource is assigned directly, without the role, then the **Create Mail Account** button is displayed. When this button is clicked, then the user is assigned the Mail Users role, and the button disappears.

The user form performs a validation on any value entered in the **Primary Email** and **Other Email** fields. The system checks for matching email addresses on the directory server. If a match is found, an error message is displayed.

Workflow

In the LDAP scenario, a task template is launched when a user is created from the Administrative interface. A *task template* is a set of components and product enhancements that enable administrators to configure workflow behavior through the Administrative Interface, instead of writing customized workflows to achieve the same result.

Task templates can be accessed by clicking **Tasks**, then the **Task Templates** subtab.

A task template contains the following elements:

- TaskDefinitionRef — Defines which TaskDefinition will be launched to perform the action, in this case, to create a user. Standard workflows are often defined as a TaskDefinition.
- FormRef — A reference to a form that can generate a user interface for configuring a TaskTemplate.
- Variables — Contains attributes that are used in configuring the task. Variables are not discussed here.

(Cross reference to new Task Template documentation.)

The following XML fragment shows the top of the `TaskTemplate-CreateUser.xml` file:

```
<TaskTemplate id='#ID#TaskTemplate:CreateUser' name='Create User
Template' taskType='TaskConfiguration' visibility='invisible'>
   <TaskDefinitionRef>
      <ObjectRef type='ProvisioningTask' name='Create User'/>
   </TaskDefinitionRef>
   <FormRef>
      <ObjectRef type='UserForm' name='Create User Template Form'/>
   </FormRef>
...
</TaskTemplate>
```

The TaskTemplate is named Create User Template. The TaskDefinitionRef points to the Create User ProvisioningTask, which is the standard workflow for creating users. The FormRef refers to the Create User Template Form, a pre-defined form that controls the process of creating a user through a TaskTemplate.

Approvals

The Create User Template task template defines an approval workflow that runs when a user is created in Identity Manager. This task template does not configure notifications or other template features, but they can still be implemented, if desired, in this scenario.

To enable the Create User Template, the following prerequisites must be met:

- The createUser process map must be updated to point to the Create User Template. Click **Configure**, then the **Form and Process Mappings** subtab. Under the Process Mappings section, change the value for createUser from `Create User` to `Create User Template`. This mapping has already been configured in the LDAP scenario.

- The Get Managers rule (`Rule-GetManagers.xml`) uses the `ismanager` attribute to determine whether an account belongs to a manager, as shown in the following XML fragment:

```
<map>
   <s>searchAttrsToGet</s>
   <list>
      <s>cn</s>
      <s>dn</s>
      <s>ismanager</s>
   </list>
   <s>searchScope</s>
   <s>subTree</s>
   <s>searchFilter</s>
   <s>(&amp;(objectclass=inetorgperson)(ismanager=Y))</s>
</map>
```

  If you use a different attribute to convey this status, change the rule to match this customized attribute. If you do not use this type of attribute, remove all references to the `ismanager` attribute. In this case, also change final string in the map to the following:

```
<s>(objectclass=inetorgperson)</s>
```

In this scenario, a create user request requires approval from the user's manager, who is not an Identity Manager administrator. The manager is determined by an approval query. The following fields have been pre-configured for this scenario:

- **Determine additional approvers from** — The **Query** option indicates the account IDs of approvers will be determined by querying the LDAP resource. The parameters of the query are constructed from the **Resource Attribute to Query** and **Attribute to Compare** fields.
- **Resource Attribute to Query** — The `entrydn` attribute is queried on the resource. The query is searching for the user's manager.
- **Attribute to Compare** — The value of `user.accounts[LDAP Resource].manager` is compared to the value in the `entrydn` resource attribute. A match indicates the approver of the new user has been found.
- **Approval times out after** — The default value is 1 day. If the manager does not approve the request to create the user, an approval request will be sent to the manager's manager. (The **Escalation Administrator Query** is identical to the **Approval Administrator Query**.) Note that an approval request can only be escalated once. If the second-line manager does not approve the request in the specified time, the request is not forwarded to the third-line manager.
- **Approval Attributes** — Attributes that will be displayed on the approval notice. The new user's first name, last name, email address, and LDAP groups are displayed on the notice, but these values can be changed by the approver. See (cross-reference) for more information about the approval user form.

If the LDAP resource is configured so that an approver is required on the resource, that approval must occur before the manager receives the approval request.

Notifications

The LDAP scenario uses a standard workflow process, Notification Evaluator, to handle notifications. The Notification Evaluator process has been extended by adding an activity named Process User Manager.

The Process User Manager activity calls two rules:

- Get LDAP Resource Name — Determines which resource to contact to determine the user's manager.
- Get Manager Account Id — Queries the LDAP resource and returns the value of the user's manager attribute.

Upon fetching the value of the manager's account ID, Identity Manager will attempt to send a notification email to the manager. The email template must be configured correctly

Updating User Accounts

In general, the workflow processes that occur when changes are made to account mimic those that occur when accounts are created. The processes are simpler, however, because reprovisioning an existing user is less complicated than provisioning a new user.

Updates Discovered by Reconciliation

Data changes discovered by reconciliation go through the same general process as when creating a user. The User Form with Questions form controls the process.

See *Using Reconciliation to Create Users* on page C-11 for more information.

Updates Discovered by ActiveSync

The LDAP ActiveSync Update User workflow defines how ActiveSync updates are performed. When changes are discovered via ActiveSync, Identity Manager triggers the reProvision workflow service.

See *Using ActiveSync to Create Users* on page C-12 for more information.

Updates Performed with Identity Manager

In the LDAP scenario, user data can be edited by an administrator from the Administrative Interface, or by an end-user or manager from the User Interface. Regardless of how this data is changed, Identity Manager runs the same workflow and handles approvals and notifications in the same manner. See *Using Identity Manager to Create Users* on page C-15 about the workflow, approvals, and notifications.

The Update User Template TaskTemplate defines the workflow. It is an abbreviated version of the Create User Template TaskTemplate. In most cases, it is less urgent to change attributes of an existing user than to create the user in the first place. Therefore, the Update User Template TaskTemplate does not include these features.

Notifications work the same as when creating a user.

The updateUser process map must be updated to point to the Update User Template. This mapping has already been configured in the LDAP scenario, but to replicate this, click **Configure**, then the **Form and Process Mappings** subtab. Under the Process Mappings section, change the value for updateUser from `Update User` to `Update User Template`.

Implementing Changes to the User Interface

The User Interface presents a limited view of the Identity Manager system that is specifically tailored to users without administrative capabilities. By default, the User Interface allows users to change passwords and modify account attributes.

The LDAP scenario adds the following functionality to the standard user interface:

- Provides a default authentication question. The answer to the authentication question is read from the carLicense attribute of the inetOrgPerson object class.
- A user can change a variety of account attributes, including LDAP group memberships. Changing the any value requires approval from the user's manager.
- A user can request access to a resource.
- If the user is a manager, change attributes of his or her subordinates.

These features are discussed in detail below.

Setting Answers to Authentication Questions

If a user forgets his password, or his password is reset, he can answer one or more account authentication questions to gain access to Identity Manager. These questions, along with the rules that govern them, are defined in the account policy.

The Default Lighthouse Account Policy in this scenario contains the default authentication question “Car license plate number?”. When an account is first reconciled, the User Form With Questions form loads the value of the carLicense attribute of the inetOrgPerson object class. The user form performs some additional processing so that the value of carLicense is used as an answer to the authentication question.

However, it should be noted that license plate number may not be the best attribute to use for authentication questions, as this information is often omitted from the corporate LDAP servers. Also, a license plate number is not a secure piece of information, so an unauthorized person could easily reset another person's LDAP password.

A better attribute would have ACIs in the directory that restrict read access to the owner and high-level administrators, such as IDM User and directory administrators. The availability of such an attribute will vary for each directory server.

If such an attribute is not available, you could expand the scenario so that the user must specify an employee ID as well as the license plate number. To do this, you must update the resource’s schema map, update the account policy, and update the User Form with Questions form. Again, this scenario is not secure if employee IDs are publicly available.

Resource

The schema map must define any attribute that is used in any user form, rule, or other Identity Manager object. In this scenario, the `employeenumber` LDAP attribute has already been mapped to the `employeeNumber` user attribute.

Account Policy

The account policy can modified from the Administrative Interface by clicking the **Configure** tab, then the **Policies** subtab. Select the account policy to be edited and enter a new question, such as “Enter Employee ID”, under the **Secondary Authentication Policy Options** heading.

To add the question directly to the `Policy-DefaultLighthouseAccountPolicy.xml` file, modify the questions attribute list as follows.

```
<Attribute name='questions'>
```

```
   <List>
      <Question>Car license plate number?</Question>
      <Question>Enter Employee ID</Question>
   </List>
</Attribute>
```

User Form

Replace the `FieldLoop` block in the "User Form with Questions" form with the following:

```
<FieldLoop for='questionName' in='waveset.questions[*].name'>
   <Field name='waveset.questions[$(questionName)].answer'>
      <Expansion>
         <cond>
            <eq>
               <ref>waveset.questions[$(questionName)].question</ref>
               <s>Car license plate number?</s>
            </eq>
            <ref>global.carLicense</ref>
            <cond>
               <eq>
                  <ref>waveset.questions[$(questionName)].question</ref>
                  <s>Employee number?</s>
               </eq>
               <ref>global.employeeNumber</ref>
            </cond>
         </cond>
      </Expansion>
      <Disable>
         <and>
            <neq>
               <ref>waveset.questions[$(questionName)].question</ref>
               <s>Car license plate number?</s>
            </neq>
            <neq>
               <ref>waveset.questions[$(questionName)].question</ref>
               <s>Enter Employee ID</s>
            </neq>
         </and>
      </Disable>
   </Field>
</FieldLoop>
```

This code block determines whether each question defined in the authentication policy matches text specified in a `<s>` element. If the questions do not match, the fields are disabled.

Changing Account Attributes

The LDAP End User Form determines which account attributes are displayed and whether the attributes can be edited. When any attribute is edited, the user's manager is notified of the change. The manager must then approve or reject the change. For more information about the implementation of this feature, see *Using Identity Manager to Create Users* on page C-15

The LDAP End User Form allows users to request changes to the LDAP groups they are a member of. To manage LDAP groups, the ldapGroups attribute must be added to the adapter's schema map. In this scenario, the manager and employeeNumber attributes from the inetOrgPerson object class have also been added to the schema map to track who is the manager of the user.

The User Form calls the Get LDAP Groups rule. This rule gets the list of LDAP groups available on the directory server.

The LDAP End User Form also contains fields named **Manager** and **Manager Name**. The **Manager** field contains the value returned from the `manager` LDAP attribute. The **Manager Name** field, however, is derived by the Get Manager Name rule. This rule looks up the LDAP user listed in the **Manager** field and retrieves this user's first name (givenName) and last name (sn), then concatenates these values.

If the user is a manager, the Subordinates table is populated with the user IDs, names, and email addresses of subordinate users. See *Manager Editing Subordinate Data* on page C-23.

Manager Editing Subordinate Data

In the LDAP scenario, managers do not have the ability to log in to the Administrative Interface. However, managers are able to edit attributes of their subordinates because the scenario creates a virtual organization, an admin role to control the organization, and provides several rules that determines whether a manager/subordinate relationship exists.

Virtual Organization

The My Employees organization contains users whose `manager` LDAP attribute matches the distinguished name of the logged in user. If a user is not a manager, then the organization is empty, and the user cannot act on other accounts. Note that a manager can act on direct reports only. A second-line manager can edit the data of a first-line manager, but not the first-line manager's subordinates.

The organization is assigned the Get My Subordinates rule. This rule has an authType of UserMembersRule and determines whether users can be a member of this organization. The rule in turn calls the Get LDAP Resource Name and Get Subordinate Account IDs rules. The Get Subordinate Account IDs rule uses the resource returned in the Get LDAP Resource Name rule to determine where to search for LDAP users whose manager is the current user. The Get Subordinate Account IDs rule searches the `cn`, `givenname`, and `dn` attribute values on the directory and returns the value in `dn`.

Admin Role

The `AdminRole-User.xml` file creates the Users admin role. This admin role is unique in that it is automatically given to all Identity Manager users. In addition, this role has the following characteristics:

- Is assigned the User Account Administrator.
- Controls the MyEmployees virtual organization, which is created in the `ObjectGroup-MyEmployees.xml` file
- Is available to the Top organization.

User Form

End User Subordinate Form — This form is similar to the LDAP End User Form, except that it does not have a subordinates section. It is accessed after a manager clicks on the name of a subordinate.

JSP Files

The `changesSubUser.jsp` file controls the display of the End User Subordinate form. It is referenced in the LDAP End User Form.

Requesting Access to Resources

The LDAP scenario gives end users the ability to request a resource account. When the user selects a resource, Identity Manager displays a set of attributes that the user can fill in to expedite the process. To enable this feature, the scenario contains customized forms, rules, and JSP pages.

Custom End User Menu User Form

When the scenario is first loaded, the configuration script changes the endUserMenu form mapping from End User Menu to Custom End User Menu. The `UserForm-CustomEndUserMenu.xml` file defines the Custom End User Menu form, which controls what is displayed when an end-user logs in to the User Interface.

The Custom End User form differs from the default End User menu in that it adds the link to the page that allows users to request access to resources. The following field adds the Request Resource Access link.

```
<Field>
   <Display class='Link'>
      <Property name='name' value='Request Resource Access'/>
      <Property name='URL' value='user/requestResources.jsp'/>
   </Display>
</Field>
```

Resource Request User Form

In the default Administrative Interface, the Tabbed User Form is displayed when creating or editing users. On the Assignments navigation tab, the Individual Resource Assignments multi-select box allows the administrator to assign one or more resources to the user. This multiselect box is the primary construct of the Resource Request User Form.

However, an end-user does not have the capability to view a list of available resources. To get around this limitation, the form calls the Get Unassigned Resources rule. This rule has two elements of note:

- The `RunAsUser` element specifies the user account that will be used to query for available resources. The example uses Configurator perform this task. In a production environment, you would create an administrator account with minimal capabilities to query for resources.
- The rule invokes the FormUtil method `getUnassignedResources`. This method builds a list of all accessible resources minus the names of the resources that are already assigned to the user through their role.

JSP Files

To enable the feature for allowing users to request access to resources, the following JSP files must be copied from `$WSHOME/sample/scenarios/scenario1/jsp/` into `$WSHOME/user/`.

- `requestResources.jsp` — Derived from `user/changeAll.jsp`. The only changes made to the `requestResources.jsp` file include setting the result and post URLs and changing several message catalog strings.
- `requestResourcesResults.jsp` — Copied without modification from `user/changeAllResults.jsp`.

Implementing Changes to the Administrator Interface

Modifying the Approval Page

When a manager logs in to the User Interface, a link labeled **View Inbox** is displayed if the manager has a pending approval request. The `UserForm-ApprovalForm.xml` enhances the default Approval Form by displaying attributes that have been changed.

Modifying the Find User Form

The scenario adds a new search option to the Find Users page. This option allows the administrator to locate users whose manager starts with, contains, or matches the specified text.

Enabling this feature requires modifications to the UserUIConfig configuration object and two user forms.

UserUIConfig Configuration Object

The UserUIConfig configuration object controls the account search function and how results are displayed. To search for users by manager, the `manager` attribute must be added to the `QueryableAttrNames` and `SummaryAttrnames` elements. `QueryableAttrNames` define attributes that can be searched, while `SummaryAttrNames` are the attributes that can be displayed in list results.The scenario adds these attributes automatically.

User Search Library User Form

The User Search Library User Form defines the `manager` field. This multi-part field contains the following facets:

- A checkbox that allows the administrator to conduct a search on multiple types.
- The label “Manager”.
- A reference to the manager attribute
- A drop-down menu for selecting the operator (**starts with**, **contains**, or **is**).
- A text box for entering the string to be searched for.

The following fragment defines the manager field.

```
<Field name='manager'>
   <Field name='query.clauses[clause1].conditions[manager].active'>
      <Display class='Checkbox'/>
   </Field>
   <Field>
      <Display class='Label'>
         <Property name='value' value='Manager'/>
         <Property name='rowHold' value='true'/>
      </Display>
   </Field>
   <Field name='query.clauses[clause1].conditions[manager].attribute'>
      <Expansion>
         <s>manager</s>
      </Expansion>
   </Field>
   <Field name='query.clauses[clause1].conditions[manager].operator'>
      <Display class='Select'>
         <Property name='valueMap'>
            <ref>displayHints.searchOperators</ref>
         </Property>
         <Property name='rowHold' value='true'/>
      </Display>
   </Field>
   <Field name='query.clauses[clause1].conditions[manager].value'>
      <Display class='Text'>
         <Property name='rowHold' value='true'/>
         <Property name='size' value='20'/>
         <Property name='maxLength' value='48'/>
      </Display>
   </Field>
</Field>
```

Find User Form User Form

The `UserForm-FindUserForm.xml` file modifies the default Find User Form to add a field reference to the `manager` field, which is defined in the User Search Library form.

Index

S

T

U

V

W

X