# Novell
# Designer for Identity Manager

1.2

July 14, 2006

www.novell.com

IDENTITY MANAGER USER
APPLICATION: DESIGN GUIDE

Novell®

## Legal Notices

Novell, Inc. makes no representations or warranties with respect to the contents or use of this documentation, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.

Further, Novell, Inc. makes no representations or warranties with respect to any software, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to make changes to any and all parts of Novell software, at any time, without any obligation to notify any person or entity of such changes.

Any products or technical information provided under this Agreement may be subject to U.S. export controls and the trade laws of other countries. You agree to comply with all export control regulations and to obtain any required licenses or classification to export, re-export or import deliverables. You agree not to export or re-export to entities on the current U.S. export exclusion lists or to any embargoed or terrorist countries as specified in the U.S. export laws. You agree to not use deliverables for prohibited nuclear, missile, or chemical biological weaponry end uses. Please refer to www.novell.com/info/exports/ for more information on exporting Novell software. Novell assumes no responsibility for your failure to obtain any necessary export approvals.



Novell, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed at http://www.novell.com/company/legal/patents/ and one or more additional patents or pending patent applications in the U.S. and in other countries.

Novell, Inc.
404 Wyman Street, Suite 500
Waltham, MA 02451
U.S.A.
www.novell.com

*Online Documentation:* To access the online documentation for this and other Novell products, and to get updates, see www.novell.com/documentation.

## Novell Trademarks

For Novell trademarks, see the Novell Trademark and Service Mark list (http://www.novell.com/company/legal/trademarks/tmlist.html).

## Third-Party Materials

All third-party trademarks are the property of their respective owners.

# About This Guide

This guide describes how to use the Designer to create user application components. It explains how to work with the provisioning view, the directory abstraction layer editor, and the provisioning request definition editor.

### Audience

This guide is intended for designers responsible for creating workflow-based provisioning applications that run on Identity Manager.

### Feedback

We want to hear your comments and suggestions about this manual and the other documentation included with this product. Please use the User Comments feature at the bottom of each page of the online documentation, or go to www.novell.com/documentation/feedback.html and enter your comments there.

### Additional Documentation

For documentation on other Identity Manager features, see the Identity Manager Documentation Web site (http://www.novell.com/documentation/idm).

### Documentation Conventions

In Novell documentation, a greater-than symbol (>) is used to separate actions within a step and items in a cross-reference path.

A trademark symbol (®, ™, etc.) denotes a Novell trademark. An asterisk (*) denotes a third-party trademark.

When a single pathname can be written with a backslash for some platforms or a forward slash for other platforms, the pathname is presented with a backslash. Users of platforms that require a forward slash, such as Linux or UNIX, should use forward slashes as required by your software.

# Introduction to the User Application Design Tools

1

This section provides an overview of the tools available for designing the user application. Topics include:



## 1.1 About the Provisioning View

The provisioning view provides persistent access to Designer’s provisioning features. Use the provisioning view to perform these actions on provisioning objects:

- Import object definitions from the Identity Vault or the local file system.
- Export object definitions to the local file system.
- Validate local object definitions.
- Deploy object definitions to the Identity Vault.
- Compare the objects on the local file system with those in the Identity Vault.
- Access the directory abstraction layer editor.
- Access the provisioning request defintions editor.

Double-clicking an item from the provisioning view opens the editor for that item.

## 1.2 About the Directory Abstraction Layer Editor

The directory abstraction layer editor allows you to define directory abstraction layer definitions. Use the directory abstraction layer editor to modify the user application’s behavior by:

- Adding new entities (Identity Vault objects).
- Defining the set of attributes for an entity.
- Specifying the contents of lists.
- Modeling relationships among entities.
- Defining automatic lookups between entitites.

## 1.3 About the Provisioning Request Definition Editor

The provisioning request definition editor allows you to create custom provisioning request definitions by using a rich set of Eclipse-based design tools. Use the provisioning request definition editor to:

- Define the basic characteristics of the provisioning request.
- Design the associated workflow.
- Define the request and approval forms.
- Configure the activities and flow paths.

## 1.4 About the ECMA Expression Builder

Designer incorporates an ECMAScript interpreter and expression editor, which allows you create script expressions that refer to and modify workflow data. For example, you can use scripting to:

- Create new data items needed in a workflow under the flowdata element.
- Perform basic string, date, math, relational, concatenation, and logical operations on data.
- Call standard or custom Java classes for more sophisticated data operations.
- Use expressions for runtime control to:
  - Modify or override form field labels.
  - Initialize form field data.
  - Customize e-mail addresses and content.
  - Set entitlement grant/revoke rights and parameters.
  - Evaluate any past Activity data to conditionally follow a workflow path using the Condition Activity.
  - Write different log messages that are conditionally triggered using a single Log Activity.

## 1.5 Documenting a Project

Designer provides a document generator that helps you quickly generate customized documentation for your Designer projects. You can define your own document style, but Designer ships with a default provisioning style. The default provisioning style includes sections for the user application, such as the directory abstraction layer and the provisioning request definitions. The directory abstraction layer documentation includes the following sections:

- Entities—including access properties, auxiliary classes, and LDAP classes.
- Global lists—including key and display label.
- Relationships—including key, parent key, parent attribute, child key, and child attribute.
- Configuration—including default entity key, default locale, and container classes.

The documentation for a provisioning request definition includes:

- A table containing the definition’s category, status, and e-mail notification.
- An image of the workflow’s structure.

- A section for each activity with a table that lists the data mappings for the activity or the expression (if supported by the activity type).
- A section for each form.

# Working with the Provisioning View

# 2

This section provides details on using the Provisioning view. Topics include:



## 2.1 Setting Up a Provisioning Project

The Provisioning View is only available for projects that contain a User Application driver. Follow these steps to set up a provisioning project:

***Table 2-1*** *Provisioning Project Setup Steps*

| Step | Task | Description |
|---|---|---|
| 1 | Create an Identity Manager project | For more information, see the section on creating a project in Designer help. |
| 2 | Configure the Identity Vault and the driver set | For more information, see the section on configuring objects in Designer help. |
| 3 | Add and configure a User Application driver | You can find the User Application driver in the provisioning folder of the Modeler palette.<br><br>For configuration details, see Section 2.1.1, “Completing the User Application Driver Configuration,” on page 11. |
| 5 | Open the provisioning view | See Section 2.1.2, “Accessing the Provisioning View,” on page 12. |

### 2.1.1 Completing the User Application Driver Configuration

Follow these steps to complete the User Application driver configuration:

1. Drop a User Application driver on the canvas.
2. When prompted select *UserApplication.xml* (the default) as the driver configuration file, then click *OK*.
3. Specify how the wizard should handle validation of your entries by clicking *Yes* or *No*.

4 Fill in the fields as follows:

| Property | What to Specify |
|---|---|
| Driver Name | The name of an existing User Application driver (the driver specified during the user application installation), or the name of a new User Application driver. |
| Authentication ID | The DN of the User Application Administrator. |
| Application password/Reenter password | The password for the User Application Administrator (above). |
| Application context | The name of the user application context, for example, IDM. |
| Host | The host name or IP address of the application server where the Identity Manager user application is deployed. This information is used:<br>• To trigger workflows on the application server to connect to access workflows (terminate, retract, and so on).<br>• To update cached data definitions. |
| Port | The port for the Host above. |

5 Click *OK*.

### 2.1.2 Accessing the Provisioning View

You can access the provisioning view in several ways.

- Select *Window* > *Show View* > *Provisioning View*.
- In the Modeler window, select *User Application*, right-click and select *Show View* > *Provisioning View*.
- Select *Project* > *Provisioning* > *Show Provisioning View*.
- Select *Provisioning View* from the FastView toolbar.

The provisioning view displays all of the provisioning projects located in the same workspace.

**TIP:** If you do not see the user applications that you expect, it might be because the project is corrupt. If your project is corrupt, you must re-create it.

## 2.2 Setting Provisioning View Preferences

You can customize some provisioning view behaviors by setting preferences. You access the preferences page through *Windows* > *Preferences* > *Provisioning*. The preferences include:

***Table 2-2*** *Provisioning View Preferences*

| Preference Category | Setting | Description |
|---|---|---|
| General | *Prompt for deletion of User Application Configuration* | When this is selected and you delete a User Application from the Modeler, Designer asks whether you want to delete the provisioning objects on disk as part of the delete operation. If you do not delete the provisioning objects, they are left on disk, even though the user application is deleted. |
| | *Set delete from Identity Vault as default for all "Confirm Delete" dialogs* | When you delete an object in the provisioning view or the directory abstraction layer editor, you are prompted to confirm the deletion. This preference determines whether the check box labeled *Delete object in Identity Vault on deploy* in the confirmation dialog box is selected by default.<br><br>Selecting this preference means that the check box is selected and the default is to delete the Identity Vault object. The local object is always deleted. |
| | *Show Provisioning View when new User Application is created or imported* | Select this option if you want Designer to launch the provisioning view when you create a new User Application driver or import an existing one. |
| Driver Configuration File | *Select Driver Configuration File for User Application Driver* | Select *Default* when you want to use the original User Application driver configuration file shipped with Designer.<br><br>Select *Custom* when you want to upload a new user application driver configuration file. You might obtain a User Application driver configuration file when performing an upgrade or installing localization or template updates. |

| Preference Category | Setting | Description |
|---|---|---|
| Import | *Delete local object on import when object has been deleted in Identity Vault* | Select this option if you want Designer to delete local objects if the corresponding Identity Vault objects were deleted. This ensures that the Identity Vault and local files are in sync.<br><br>Deselect this option if you want to leave the local files alone. |
| | *Import runtime configuration (objects used at runtime but not editable through Designer)* | Select this option if you are importing the driver from a test environment and want to deploy to a production environment. The User Application driver runtime relies on objects stored in the driver that you are not able to access in Designer. If you deploy a driver that does not contain these objects, it does not work properly.<br><br>Deselect this option if you are importing the driver, modifying it, and deploying it back to the same driver set because the driver already has the runtime configuration objects. |
| Deploy | *Allow deployment of objects with validation errors* | Select this option if you want to deploy objects that fail validation checks. At deployment, Designer validates the definitions being deployed following the validation rules outlined in Section 2.5, “Validating Provisioning Objects,” on page 17. Deselect this option to prevent deployment of definitions that fail validation. |
| | *Prompt for deployment of runtime configuration (objects used by User Application but not editable through Designer*) | Select this option if you want Designer to prompt you to include runtime configuration objects in the deploy. You should include runtime configuration objects when you are moving a driver set from a test environment to a production environment (or any time you want a complete copy of an existing environment). If you are simply redeploying some updated objects to an existing driver, then you do not need to include the runtime configuration objects at deployment because they already exist. |

| Preference Category | Setting | Description |
|---|---|---|
| Localization | *Locale and Language* | Establishes the set of languages that Designer developers are allowed to localize display labels and form control tooltips. The languages specified here are displayed in the localize dialog boxes used in the directory abstraction layer editor and provisioning request definition editor.<br><br>Click the button to access the *Add/Remove Languages* dialog box. Once you apply the change, the language is available in Designer. Removing a language from this list does not cause any translated strings for the language to be removed. They are still stored with the object, the language is just not displayed in the localization dialog boxes. |
| Workflows | *Form Templates* | Use this dialog box to remove or preview existing form templates. |
| | *Diagram Preferences* | Show Activity ID—Select this preference when you want the Workflow tab of the provisioning request definition editor to display the Activity IDs for each activity in the flow. Activity IDs are used by the ECMA expression builder and are written to the user application's error logs. |
| | *Connection* | This is the amount of time (in milliseconds) for Designer to connect to the Identity Vault. If this is set too low, you might encounter an error when trying to set Trustee Rights on a provisioning request definition or when trying to access the Identity Vault via the ECMA expression builder. |

## 2.3 Importing Provisioning Objects

The provisioning view's import feature lets you import provisioning objects from:

- A driver configuration file
- An Identity Vault

This feature is useful when you begin a new project based on one or more definitions from an existing project, or when you want to share definitions with other developers working on the same project.

**NOTE:** When you change the Identity Vault or DriverSet's deploy context, you must save the project before performing an import. If you do not save the change, Designer continues to use the old deploy context for import operations.

### 2.3.1 Importing from a Driver Configuration File

To import objects from a driver configuration file:

1. Open the *provisioning view*.
2. Select the root node representing the type of object you want to import.
3. With the container selected, right-click and select *Import from File*. Confirm the import operation (which might overwrite existing definitions of the same name) by clicking *OK*.
4. Specify the name of the driver configuration file you want to import, then click OK.

### 2.3.2 Importing from an Identity Vault

1. Open the *provisioning view* and select the container into which you want to import the definitions.

   To import a specific provisioning object, select that node in the provisioning view. To import all objects of a specific type, select the root node representing that type.
2. With the container selected, right-click and select:
   - *Import Object.*to import the specified object and any of its children.
   - *Import All* to import all of the objects of a selected container.
3. Provide the Identity Vault credentials and click *OK*.
4. Navigate to the Identity Vault container or object that you want to import and click *OK*.
5. Review the Import Summary page to determine how you want to proceed. To complete the import, click *Import*, or click *Cancel*. If you choose Import, Designer performs the operation and displays a summary of the completed operation.

## 2.4 Exporting Provisioning Objects

The provisioning view's export feature allows you to move project components from one project to another without re-creating the contents. It also allows you to clone a project. You can use it to export provisioning objects (and their children) to an XML-based driver configuration file. You use the resulting file as the input to the import from file feature enabling you to easily share the contents of your provisioning project with other developers.

### 2.4.1 Exporting to a Driver Configuration File

1. Open the *provisioning view* and select the object containing the definitions to export.

   To export a specific provisioning object, select that node in the provisioning view. To export all of the objects of a specific type, select the root node representing that type.
2. With the container or object selected, right-click and select *Export to File.*
3. Provide the name and location of the file to generate, then click *OK*.

   The default name for the file reflects the contents of the file. For example, if you export lists, the default name for the file is `lists.xml`. You can change the name as needed.

## 2.5 Validating Provisioning Objects

The validation feature allows you to validate provisioning objects on the local file system before you deploy. The validation runs Designer's project checker and displays the results in the Project Checker View.

For directory abstraction layer objects, Designer does the following:

- Verifies that the XML is well-formed and complies with the schema that defines the elements needed for entities, attributes, lists, relationships, and so on.
- Checks every entity to ensure that references to other entities and global lists are valid.

  For example when validating an entity and its attributes, the validator checks that all references to other entities via the Edit Entity, DNLookup, and Detail Entity reference entities that actually exist.
- Ensures that every entity has at least one attribute defined.
- Ensures that every local and global list contains at least one item.

For Provisioning Request Definitions, Designer does the following:

- Validates that every Provisioning Request Definition has at least one request form and one approval form.
- Ensures that the Condition Activity has both an outbound true flow path and an outbound false flow path.
- Ensures that the Entitlement Activity Data Item Mapping of DirXML-Entitlement-DN is valid.
- Ensures that the Final Timeout Action property (for User Activities) has a matching flow path link leading from the activity. For example, if Final Timeout Action=denied, there must be a denied link.
- For Branch and Merge activities, ensures that a workflow has an equal number of Branch and Merge activities. It also ensures that all paths descending from a Branch activity merge into one Merge activity, that all merge activities have a branch activity, and that all Merge activities have a branch-activity-id attribute.

To validate objects from the provisioning view, right-click a node and click *Validate*.

To validate objects from the directory abstraction layer editor, click *Validate Abstraction Layer* from the editor's toolbar, or select *DAL > Validate* from Designer's menu.

To validate objects from the provisioning request definition editor, select *PRD > Validate* from Designer's menu.

---

**NOTE:** Validation does not check the Identity Vault for the existence of any object.

---

## 2.6 Deploying Provisioning Objects

The provisioning view's deploy feature deploys your provisioning objects to the specified user application driver. You must deploy any changes you've made to the provisioning objects in the design environment before you see them reflected in the Identity Manager user application. The provisioning view allows you to deploy a container and all its children (for example, all entities or all lists), or to deploy just a single provisioning object (such as a single list element). When you select an item to deploy, Designer compares it to the same item in the Identity Vault. If they are

equal, Designer prevents you from deploying. When there are differences, Designer displays them and allows you to proceed or to cancel the deployment.

**NOTE:** When you change the Identity Vault or DriverSet's deploy context, you must save the project before performing a deploy. If you do not save the change, Designer continues to use the old deploy context for deploy operations.

## 2.6.1 Deploying Provisioning Objects

1. Save any changes.

   If the objects contain unsaved changes, Designer displays the unsaved definitions and prompts you to save them. If you do not, Designer still deploys the objects but does not deploy the unsaved changes. Choosing not to save the changes does not cancel the deployment.

2. Open the provisioning view, select the object to deploy, right-click and select *Deploy* or *Deploy All*.

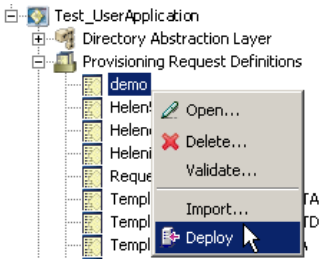
   To deploy a specific provisioning object, select that node in the provisioning view. To deploy all of the objects of a specific type, select the root node representing that type.

   Designer prompts you for Identity Vault credentials, validates the objects, and writes any messages to the project checker view.

## 2.6.2 Deploying Provisioning Request Definitions

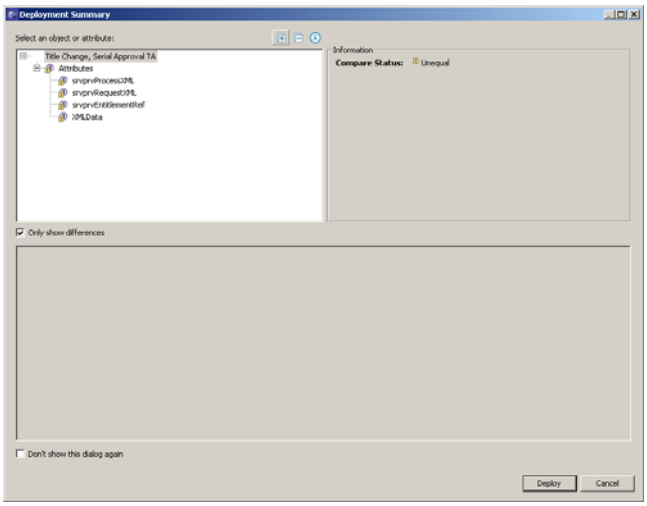
**NOTE:** If errors associated with activities are detected during deployment of a provisioning request definition, Designer identifies the activity in which the error occurred by activity ID. However, in the user interface, Designer by default displays activities by activity Name. To make it easier to identify the activity in an error message, you should turn on the display of activity IDs before you deploy the provisioning request definition. To turn on the display of activity IDs, right-click the Workflow canvas (see Section 7.1, "About the Workflow Tab," on page 109) and select *Show Activity IDs*.

1. In the provisioning request definition editor, right-click the *Overview* tab.
2. Select *Active* from the *Status* list.
3. In the provisioning view, right-click the name of the provisioning request definition that you want to deploy, then select *Deploy*.



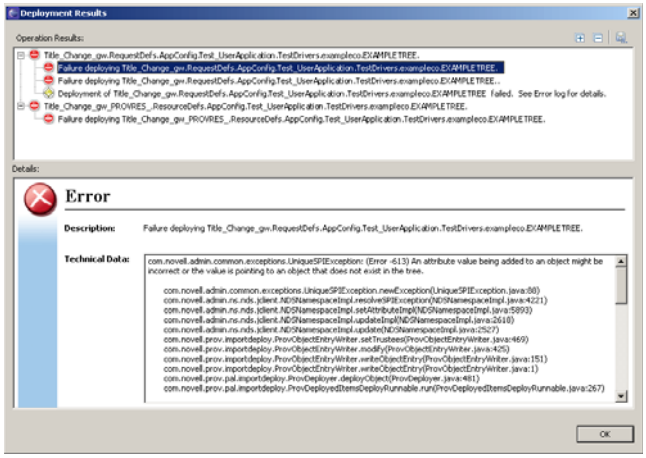

Designer performs a validity check. If there are certain fatal errors (for example, a missing Start or Finish activity) a message notifying you of the fatal error displays immediately. If this is the

case, correct the errors before attempting to deploy the provisioning request definition. If there are no fatal errors, the Deployment Summary is displayed.




If the provisioning request definition with the same CN already exists in the Identity Vault, Deployment Summary displays the differences, and allows you to examine the differences before you decide to proceed.

4 Click *Deploy* to deploy the provisioning request definition. If an error occurs during deployment, a description of the error is displayed in the Deployment Summary:




A common error (and the source of the error in the preceding illustration) occurs when you fail to replace a placeholder expression in an entitlement provisioning activity (see Section 8.1.8, "Entitlement Activity," on page 132 for information about entitlement provisioning activities). If this is the case, correct the error, then try deploying the provisioning request definition again.

---

**NOTE:** Designer cannot evaluate expressions at design time, so you may receive a warning if you've used an expression for an entitlement that must be resolved at runtime. This is not a fatal error and the deployment will succeed.

---

# 2.7 Comparing Provisioning Objects

The provisioning view's Compare feature allows you to see the differences between the provisioning objects in the local file system and those that are running in the deployed User Application driver. When Designer encounters a difference it allows you to specify what action you want to take on that difference. You can ignore or reconcile it.

**NOTE:** When you change the Identity Vault or DriverSet's deploy context, you must save the project before performing a compare. If you do not save the change, Designer continues to use the old deploy context for compare operations.

## 2.7.1 To Compare Provisioning Objects

1. Select a container or object in the provisioning view, right-click and select *Compare*.
2. If prompted, provide Identity Vault credentials, then click *OK*.

   Designer displays the results of the comparison. By default, only the differences are displayed, but you can show the full comparison by unselecting *Only show differences*.
3. If there are differences, select one of the following actions:

| Reconcile Status | Description |
|---|---|
| Do not reconcile | Do not change any definitions. |
| Update Designer | Import the definitions from the Identity Vault. |
| Update eDirectory | Deploy the definition from Designer to the Identity Vault. |
| Reconciled by parent | For informational purposes. Specifies whether one of the parent objects is already being reconciled. It is always disabled and is only set if the parent object is already being reconciled to Designer or the Identity Vault. |

# Configuring the Directory Abstraction Layer

3

This section provides details on configuring the directory abstraction layer. Topics include:



## 3.1 About the Directory Abstraction Layer

The directory abstraction layer is a set of XML-based files that define a logical view of an Identity Vault for the user application. The directory abstraction layer defines:

- The Identity Vault objects and attributes that the user application can display or modify.
- How the user application displays Identity Vault data.
- What relationships the user application can display.
- The provisioning request categories the user application can display.

You use the directory abstraction layer editor to define the contents of the directory abstraction layer.

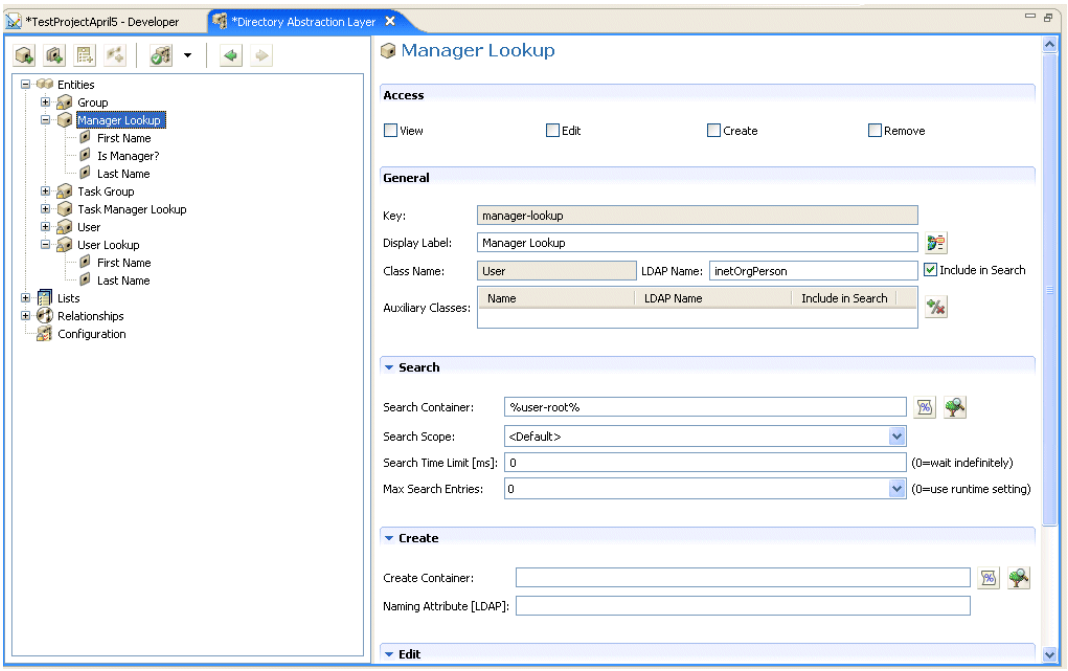
### 3.1.1 About the Directory Abstraction Layer Editor

The directory abstraction layer editor is a graphical tool for defining the directory abstraction layer files. When you add a User Application driver to an Identity Manager project, Designer creates an initial set of directory abstraction layer files. These base files are displayed when you start the directory abstraction layer editor.

To start the directory abstraction layer editor:

1. Open the *Provisioning view* and double-click the *Directory Abstraction Layer* node.

Designer displays a tree containing *Entities*, *Lists*, *Relationships*, and *Configuration* nodes.




The following table describes the nodes.

| Node | Description |
|---|---|
| *Entities* | Entities represent the Identity Vault objects available to the user application. There are two types of entities:<br>• **Entities mapped from the schema:** Entities that represent Identity Vault objects directly exposed to users via the user application. Users can typically create, search, and modify the attributes of these entities.<br>• **Entities representing LDAP relationships**: Called DN lookups, these entities represent indexed searches and are used to support particular types of attributes in the user application. DN lookup entities provide information about relationships between LDAP objects. DN lookup entities are:<br>  • Used by the Org Chart portlet to determine relationships.<br>  • Used in the Search List, Create, and Detail portlets to provide selection lists and DN contexts.<br>  • Available to the workflow request and approval flow forms you define using the provisioning request definition editor. |
| *Lists* | Defines the contents of global lists. Global lists are:<br>• Associated with an attribute. The user application displays the attribute values as a drop-down list in the user application.<br>• Used to display Resource Request categories. |
| *Relationships* | Lets you map hierarchical relationships among schema-based entities. Used by the Organization Chart action of the *Identity Self-Service* tab of the user application. |
| *Configuration* | General configuration parameters. |

2 Use the left pane to navigate the directory abstraction layer nodes. When you select an item in the left pane, the right pane displays the attributes and settings for the selection.

3 Use the right pane to define the properties for the selection. For more information about the properties, see Section 3.7, "Directory Abstraction Layer Property Reference," on page 36.

The following table describes the directory abstraction layer toolbar:

***Table 3-1*** *Directory Abstraction Layer Toolbar*

| Toolbar button | Description |
|---|---|
|  | Launches the Add Entity Wizard. |
|  | Launches the Add Attribute Wizard. |
|  | Launches the New List Wizard. |
|  | Launches the New Relationship Wizard. |
|  | Runs the Validation Checker. |
|  | Launches the Set Global Access Modifiers dialog box. |
|  | Launches the Set Global Localization dialog box. |
|   | Sets focus on the next or previous location. |

## 3.1.2 About Directory Abstraction Layer Editor Files

The directory abstraction layer files you work with are stored in the Designer project's `Provisioning\AppConfig\DirectoryModel` directory. The filenames are derived from the object key.

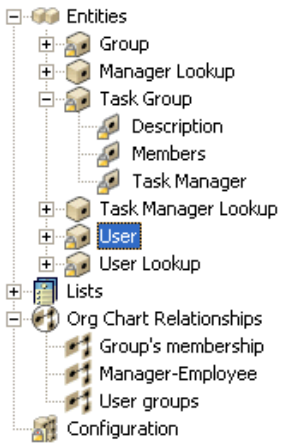
***Table 3-2*** *Local Directory Abstraction Layer Directories*

| Directory name | Description |
|---|---|
| `ChoiceDefs` | Contains the files that define global lists. Files have the `.choice` extension. |
| `EntityDefs` | Contains the files that define the entities and attributes. Files have the `.entity` extension. |
| `RelationshipDefs` | Contains the files that define the relationships available to the Org Chart portlet. These files have the `.relation` extension. |

Designer creates the base set of directory abstraction layer files for each provisioning project. An identical set is deployed to the User Application driver when the user application is installed.

To customize the Identity Manager user application, you change the directory abstraction layer objects and deploy the changes to the User Application driver. Some entities, attributes, lists, and relationships are required for the user application to function properly. The editor displays a lock next to the definitions that you should not delete. From the list below, you can see that you should not delete the Task Group entity or any of its attributes.

***Figure 3-1*** *The Task Group Entity Attributes*




# 3.2 Working with Entities and Attributes

You can customize your user application by adding objects and their attributes based on the content of your own Identity Vault. You do this by adding new entities and attributes to the directory abstraction layer and deploying them to the User Application driver.

To modify the entity files installed by default, see Section 3.2.1, “Adding Entities,” on page 25 and Section 3.2.2, “Adding Attributes,” on page 27. To modify the entity files of an already deployed project or a set of files defined by another developer, you must first import the files to your design environment. For information on importing files, see Section 2.3, “Importing Provisioning Objects,” on page 15.

### 3.2.1 Adding Entities

You add entities through the Add Entity Wizard (described next) or by clicking the *Add Entity* button (from the editor's toolbar).

---

**NOTE:** When using the *Add Entity* button, you are prompted to select the object class of the entity to create, and the editor automatically adds the required attributes to the entity. Use the Add Attribute dialog box to complete the entity definition.

---

To add an entity using the Add Entity Wizard:

1. Launch the Add Entity Wizard in one of these ways:

   From Designer's menus:

   - Select *File* > *New* > *Provisioning*. Choose *Directory Abstraction Layer Entity*, then click *Next*.

   From the provisioning view:

   - Select the *Entities* node, right-click, then choose *New*.

   From the directory abstraction layer editor:

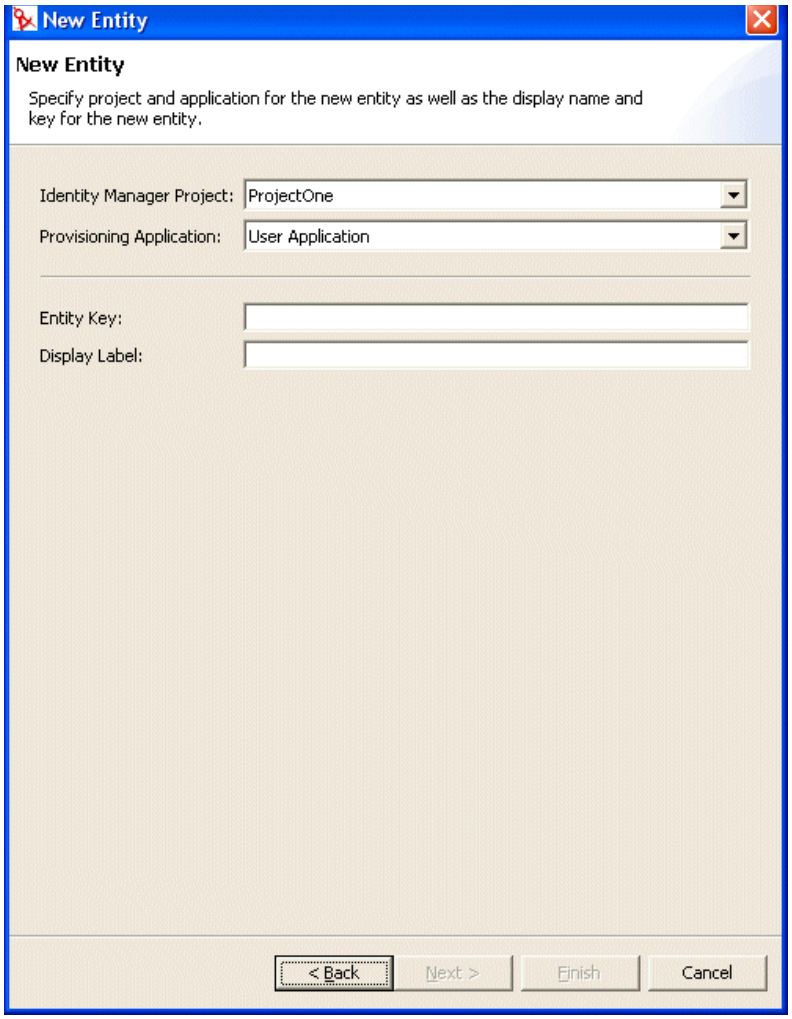
   - Select *DAL* > *New* > *Entity*, or
   - Select the *Entities* node, right-click, then choose *New Entity-Attributes Wizard*.

   The New Entity dialog box displays.

   ---

   **NOTE:** If launched from the *File* menu, the dialog box contains the additional fields shown below.

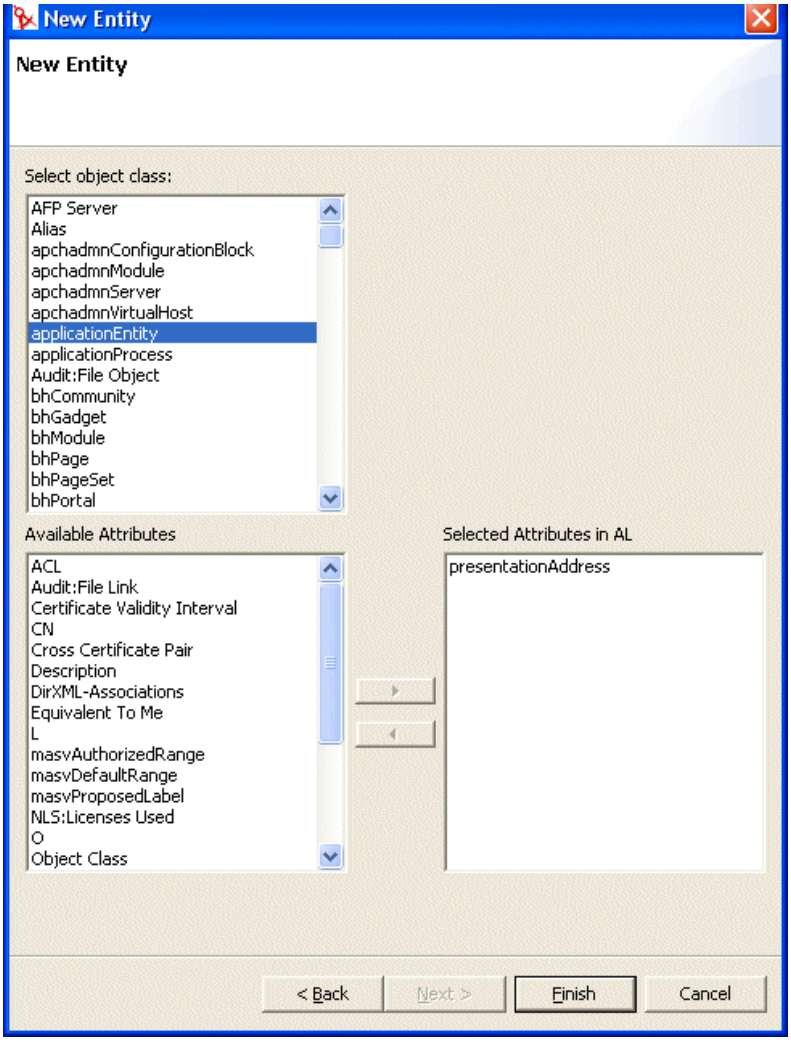
   ---




**2** Fill in the fields as follows:

| Field | Description |
|---|---|
| *Identity Manager Project and Provisioning Application* | The Identity Manager project and the provisioning application where you want to add the entity and attributes.<br><br>**NOTE:** These fields display when you launch the wizard from the *File* menu. |
| *Entity Key* | A unique identifier for the entity. |
| *Display Label* | The string displayed when the entity is displayed by the user application. You can localize this label. For more information, see Section 3.6, “Localizing Display Text,” on page 35. |

3 Click *Next*. The New Entity dialog box displays:




4 Choose the entity's object class and add the attributes you want by double-clicking them in the *Available Attributes* list.

---

**TIP:** If the entity's object class is not shown in the *Available Object Classes* list, you should update Designer's local schema file following the steps described in "Updating the Schema Elements List" on page 29.

---

5 Click *Finish*.

The property page displays for editing. For more information, see "Entity Properties" on page 36. You must deploy the entity before it is available to the user application.

## 3.2.2 Adding Attributes

1 Select an entity.

2 Add an attribute by:

- Right-click then select *Add Attribute.*

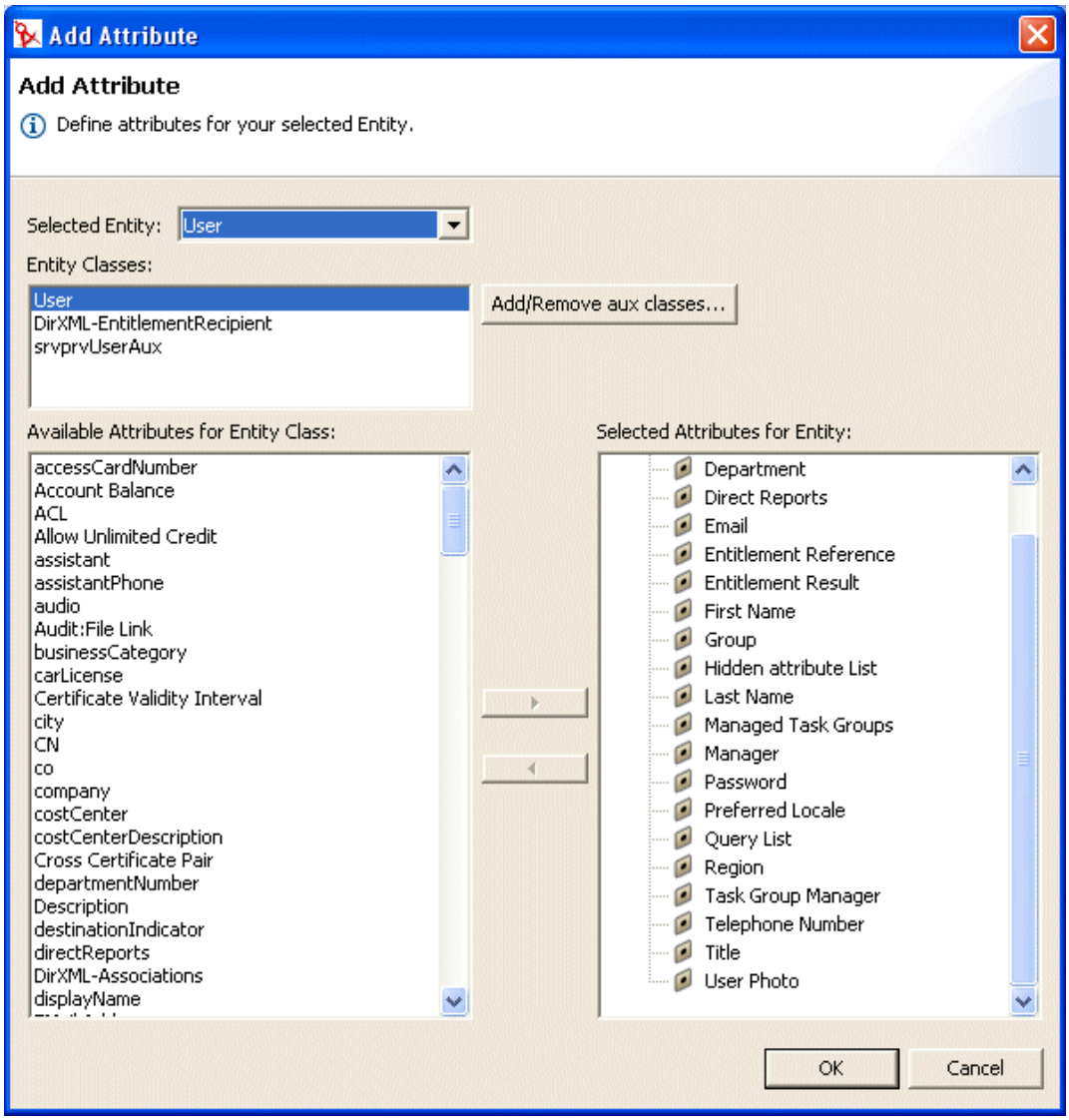
or

- Clicking the *Add Attribute* button.

or

- Select *DAL* > *New* > *Attribute*.

You are prompted to choose classes and attributes.




3 Add attributes by double-clicking them in the *Available Attributes for Entity Class* list.

---

**TIP:** If the attribute you want to add is not displayed in the Available Attributes from Entity Class list, you should update Designer's local schema file by following the procedure in "Updating the Schema Elements List" on page 29. LDAP operational attributes are not supported by the directory abstraction layer editor or user application.

---

4 Click *OK*. The property page displays for editing. For more information, see "Attribute Properties" on page 39. To make an attribute available to the user application, you must deploy it.

### 3.2.3 Updating the Schema Elements List

1. With the Identity Manager project open, select your Identity Vault, right-click and select *Live* > *Import Schema.*
2. Choose *Import from eDirectory* and provide the specifications for the eDirectory host.
3. Click *Next*.
4. Select the classes and attributes to import, then click *Finish.*

---

**NOTE:** LDAP operational attributes are not supported by the directory abstraction layer editor or user application so you will not see them in any attribute lists after importing schema.

---

## 3.3 Working with Lists

The lists node lets you define the contents of global lists. Global lists are used by the Identity Manager user application to:

- Provide a list of values for an attribute. When the attribute is displayed for editing in the user interface, the possible values are displayed as a drop-down list.
- Define the *Preferred Locale* list, for details, see Section 3.3.1, “About the Preferred Locale List,” on page 32.
- Define the categories available to the Provisioning Request Configuration plug-in to iManager. This is a special list. For details, see Section 3.3.2, “About the Provisioning Category List,” on page 32.

To create a new global list:

1. Launch the New List Wizard in one of these ways:

   From Designer’s menus:

   - Select *File* > *New* > *Provisioning*, then choose *Directory Abstraction Layer List*, then click *Next.*

     When launched from the File menu, the dialog box contains fields not displayed when launched in other ways.
   - Select *DAL* > *New* > *List*.

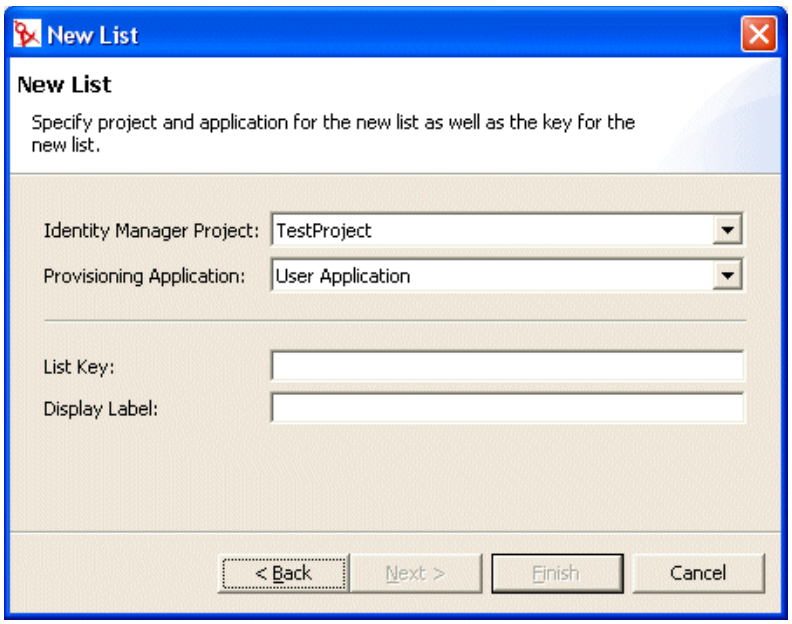
   From the provisioning view:

   - Select the Lists node, right-click and choose *New*.

   From the directory abstraction layer editor:

   - Click *New List.*
   - Select the Lists node, right-click and choose *Add List.*

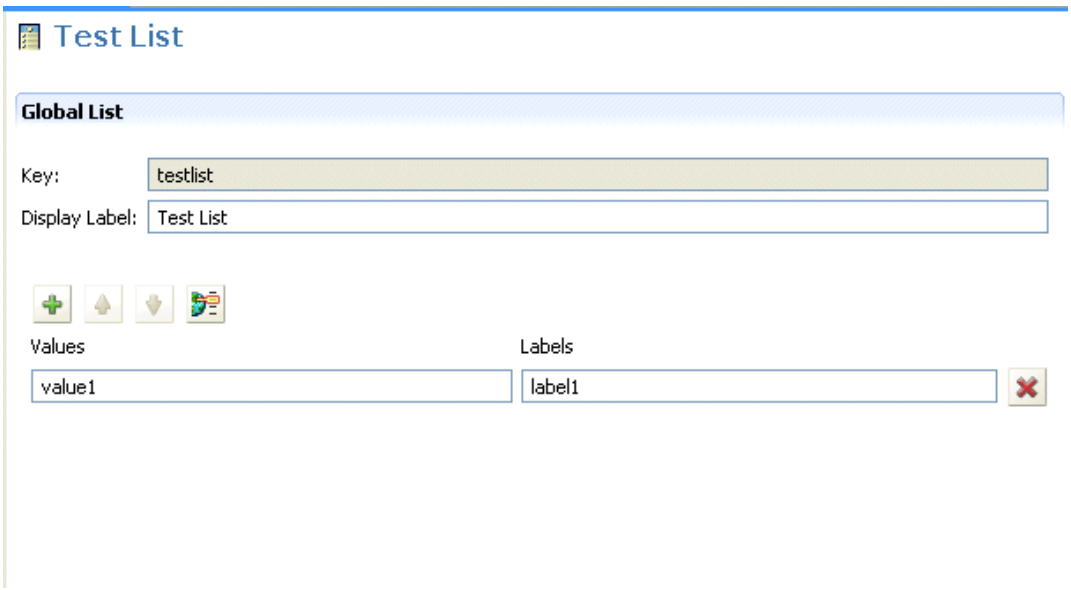
The New List dialog box displays.




2 Fill in the fields as follows:

| Field | Description |
|---|---|
| *Identity Manager Project and Provisioning Application* | Select the Identity Manager project and provisioning application where you want to add the list.<br><br>**NOTE:** These fields display when you launch the wizard from the *File* menu. |
| *List Key* | The unique identifier for the list. |
| *Display Label* | The string used when the list is displayed in the user application. You can localize this label. For more information, see Section 3.6, “Localizing Display Text,” on page 35. |

3 Click *Finish*.The Global Lists property page displays for editing.




4 Fill in the fields as follows:

| Field | Description |
|---|---|
| *Display Label* | The name of the list as shown in Designer. |
| *Labels* | The text for the list item to display in the user application. |
| *Values* | The list item value stored in the Identity Vault. Valid characters include letters, numbers, and the underscore (_) character. |

The following table describes the wizard's buttons:

| Button | Description |
|---|---|
|  | Adds a new Value |
|  | Moves the row up or down in the list. This order specifies how the labels are displayed in the user application at runtime. |
|  | |
|  | Displays the localization dialog box. For more information on using the dialog box, see Section 3.6.2, "Localizing Text," on page 35. |
|  | Deletes the row. |

5 Save the project.

6 Deploy the list to make it available in the runtime environment.

### 3.3.1 About the Preferred Locale List

The *Preferred Locale* list represents the default language for the user application when the browser language is not a supported language. The user application's default configuration of the *Edit User* action displays the *Preferred Locale* list.

### 3.3.2 About the Provisioning Category List

The *Provisioning Category* list defines the set of categories that help you organize provisioned resources (entitlements) and provisioning requests. The categories in this list display in:

- Designer—Provisioning request definition editor plug-in
- iManager—Provisioning Request Configuration plug-in
- user application—*Requests and Approvals* tab

You cannot change the *Provisioning Category* list key, but you can add more items to the list or change the existing category values and labels.

To modify the contents of the *Provisioning Category list*:

1 Choose the Lists node.
2 Select *Provisioning Category*.
3 Use the global list property pane to make your modifications.

---

**NOTE:** The *Values* field is used to populate the category key. The *Values* field restricts you to letters, numbers, and underscore (_)characters because these are the valid characters in the category key (which is used internally as an identifier for the category).

---

4 Save, then deploy your changes. Remember to update the application server's cache.

After your changes are deployed, they are reflected in the user application and the iManager plug-in. They are reflected in Designer when saved.

## 3.4 Working with Relationships

The Relationships node allows you to define relationships between entities defined in the directory abstraction layer. The relationship can be between like entities (such as user/user) or unlike entities (such as user/device). The relationships are used by the Org Chart portlet.

The following relationships are defined, by default, for the user application:

- Group's membership
- Manager-Employee
- User groups

To successfully deploy a relationship, all of the components (entities and attributes) of the relationship must already be deployed.

To create a new relationship:

1 You can create a new relationship in any of these ways:

From Designer's menus:

- Select *File* > *New* > *Provisioning*. Choose *Directory Abstraction Layer Relationship*, then click *Next*.
- Select *DAL* > *New* > *Relationship*.

From the provisioning view:

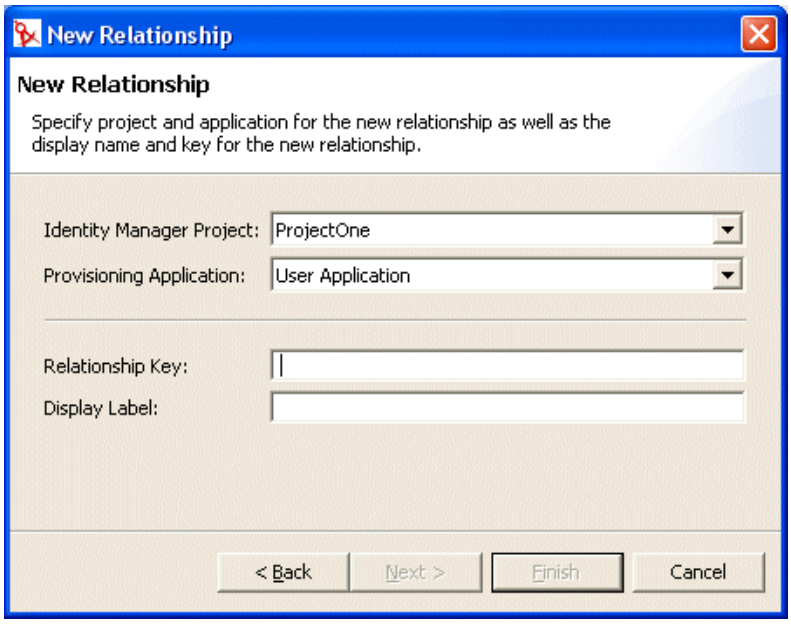
- Select *Relationships*, right-click and choose *Add*.

From the directory abstraction layer editor:

- Click the *Add Relationship* button.
- Select *Relationships* node right-click and choose *Add Relationship*.

The New Relationship dialog box displays.

**NOTE:** When launched from the *File* menu, the dialog box contains fields not displayed when launched in other ways.



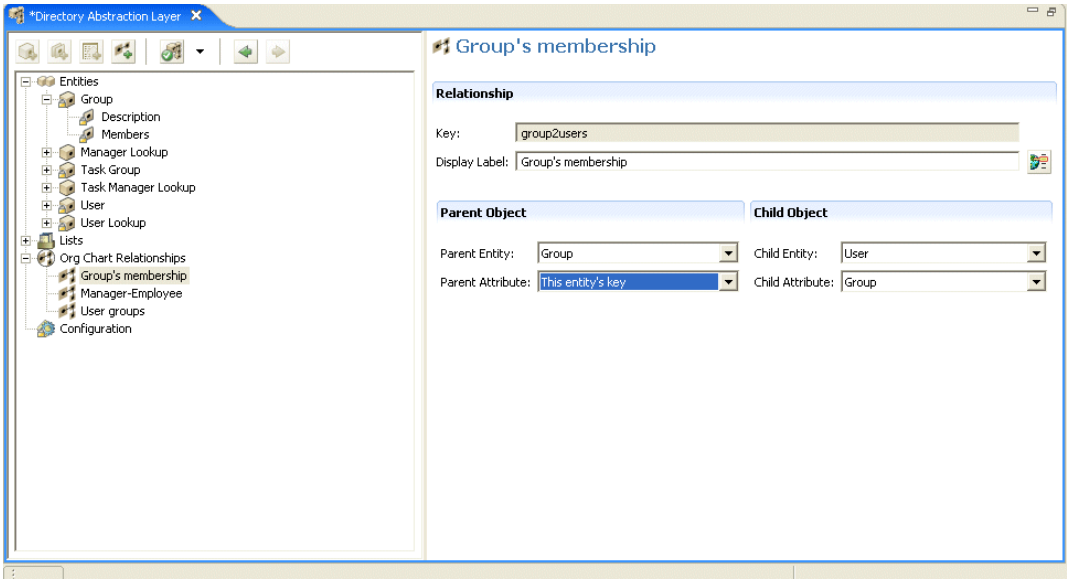

2 Fill in the fields as follows:

| Field | What to do |
|---|---|
| *Identity Manager Project and Provisioning Application* | Select the correct Identity Manager project and Provisioning Application.<br><br>**NOTE:** This field displays when you create relationships from the *File* menu. |
| *Relationship Key* | Type a unique value for the relationship key. |

| Field | What to do |
|---|---|
| *Display Label* | Type the string to display when the relationship displays in the user application. |

3 Click *Finish*.

The editor creates the relationship and opens the property page for editing.




For property definitions, see Section 3.7.3, “Relationship Properties,” on page 48.

To delete a relationship:

1 Select the relationship you want to delete.

2 Right-click and choose *Delete*.

# 3.5 Working with Configuration Settings

The Configuration node allows you to set general configuration properties for the user application.

***Table 3-3*** *Configuration Settings*

| Property | Description |
|---|---|
| Default ‘My Profile’ Entity | Defines the entity to display when the user clicks *My Profile* in the user interface.<br><br>This field is restricted to show only entities whose object class is user (or LDAP inetOrgPerson). |

| Property | Description |
|---|---|
| Default Locale | Defines the default language to use for the display labels in the user application. If the browser is set to an unsupported language this locale is used instead.<br><br>**NOTE:** The browser locale overrides the default locale for the supported languages. |
| Container Classes | This provides the *Create User or Group* action with the contents of a selection list of container classes. The user selects a container from the selection list as the location for the newly created object. |

# 3.6 Localizing Display Text

The directory abstraction layer editor provides an easy way to localize the display text for:

- Entity and attribute display labels
- Relationship names
- Global and local list items

## 3.6.1 Supported Languages

You can localize the display text in any language supported by the Java Locale class. You can set the languages you want to localize through Designer's localization preference (*Windows* > *Preferences* > *Provisioning* > *Localization*). For more information, see Section 2.2, "Setting Provisioning View Preferences," on page 12.

## 3.6.2 Localizing Text

The directory abstraction layer editor provides multiple ways to localize abstraction layer definitions. You can access the localization dialog boxes in these ways:

***Table 3-4*** *Ways to Access the Localization Dialog Boxes*

| To define the localization text for | Action |
|---|---|
| Every localizable item in the directory abstraction layer | Select *DAL* > *Set Global Localization*.<br><br>or<br><br>Click *Set Global Localization* (from the editor's toolbar), then select the *Target Language* before entering the localized text in the *Target* field. |
| A specific entity, relationship or list | From the tree view select the object to localize, right-click and select *Localize*, then select the *Target Language* before entering the localized text in the *Target* field. |
| A single display label | Select a specific entity or attribute, then click *Localize Display Label* (beside the *Display Label* field in the Property pane). |

Each dialog box prompts you for the following localization information:

***Table 3-5*** *Localization Information*

| Field | Description |
|---|---|
| *Origin* | This is typically the object type (such as entity, list, or relationship) and key. |
| *Source* | The text to translate (such as display label). |
| *Target Language* | One of the supported languages. |
| *Target* | The translation text. |

# 3.7 Directory Abstraction Layer Property Reference

The section provides definitions for the properties for the following abstraction layer nodes:

- Section 3.7.1, “Entity Properties,” on page 36
- Section 3.7.2, “Attribute Properties,” on page 39
- Section 3.7.3, “Relationship Properties,” on page 48

## 3.7.1 Entity Properties

You can set the following kinds of properties on entities:



### Entity Access Properties

Access Properties control how the user application interacts with the entity.

**NOTE:** You can also access the access properties by selecting *DAL > Set Global Access*.

***Table 3-6*** *Entity Access Properties*

| Property Name | Description |
|---|---|
| Create | When selected, this object can be created by the user application. |

| Property Name | Description |
|---|---|
| Edit | When deselected, this object is not changeable by the user application regardless of the underlying ACLs.<br><br>When selected, this object might be changeable, but the Identity Vault ACLs are used to determine this. |
| View | When selected, this object can be displayed by the user application. |
| Remove | When selected, this object can be deleted by the user application. |

## Entity General Properties

***Table 3-7*** *Entity General Properties*

| Property Name | Description |
|---|---|
| Key | The unique identifier for this entity. It defines the way the user application references this object. |
| Display Label | Defines how the object is shown in the user interface. |
| Class Name | The eDirectory object class name. |
| LDAP Name | The LDAP object class name. |
| Include in Search | When selected, this entity is searchable in the user application. Entities used in queries by identity portlets (such as Entity Search List or Entity Org Chart) must be selected (true). |
| Auxiliary Classes | A list of zero or more auxiliary classes for this entity. If adding auxiliary classes, you must specify the auxiliary class LDAP Name, Class Name, and whether or not it can be searched. |

## Entity Search Properties

***Table 3-8*** *Entity Search Properties*

| Property Name | Description |
|---|---|
| Search Container | The distinguished name of the LDAP node or container where searching starts (the search root). For example:<br><br>`ou=sample,o=ourOrg`<br><br>You can browse the Identity Vault to select the container, or you can use one of the predefined parameters described in “Using Predefined Parameters” on page 39. |

| Property Name | Description |
|---|---|
| Search Scope | Specifies where the search occurs in relation to the search root. Values are:<br><br>**<Default>**: This search scope is the same as selecting *Containers and subcontainers*.<br><br>**Container**: Search occurs in the search root DN and all entries at the search root level.<br><br>**Container and subcontainers**: Search occurs in the search root DN and all subcontainers. This is the same as selecting **<Default>.**<br><br>**Object**: Limits the search to the object specified. This search is used to verify the existence of the specified object. |
| Search Time Limit [ms] | Specify a value in milliseconds or specify 0 for no time limit. |
| Max Search Entries | Specify the maximum number of search result entries you want returned for a search. Specify 0 if you want to use the runtime setting. Recommendations: Set between 100 and 200 for greatest efficiency. Do not set over 1000 |

## Entity Create Properties

***Table 3-9*** *Entity Create Properties*

| Property Name | Definition |
|---|---|
| Create Container | The name of the container where a new entity of this type is created.<br><br>You can browse the Identity Vault to select the container, or you can use one of the predefined parameters described in "Using Predefined Parameters" on page 39.<br><br>If you do not specify this value, then the Create portlet prompts the user to specify a container for the new object. The portlet uses the search root specified in the entity definition as the base and allows the user to drill down from there. If there is no search-root specified in the entity definition then it uses the root DN specified during the user application installation. |
| Naming Attribute | The naming attribute of the entity. It is the relative distinguished name (RDN). This value is only necessary for entities where the access parameter Create is selected. |

## Entity Edit Properties

***Table 3-10*** *Entity Edit Properties*

| Property Name | Definition |
|---|---|
| Alternate Edit Entity | This entity defines the attributes displayed in the edit mode of the Detail portlet.<br><br>Select a name from the drop-down list, or select *None* if this entity is not displayed by the Detail portlet. |

### Entity Password Management Properties

***Table 3-11*** *Entity Password Management Properties*

| Property Name | Definition |
|---|---|
| Password Attribute | Choose the attribute for storing the password for this entity. |
| Password required when attribute is created | If this property is selected, a password is required when this entity is created. |

### Using Predefined Parameters

The directory abstraction layer editor allows you to use predefined parameters for certain values.

***Table 3-12*** *Predefined Parameters*

| Predefined Parameter | Description |
|---|---|
| %driver-root% | Represents the Provisioning Driver DN. This value is specified during the user application configuration during installation or a later configuration. It is stored in the user application's realm configuration. |
| %user-root% | Represents the User Container DN. This value is specified during the user application configuration during installation or a later configuration. It is stored in the user application's realm configuration. |
| %group-root% | Represents the Group Container DN.This value is specified during the user application configuration during installation or a later configuration. It is stored in the user application's realm configuration. |

## 3.7.2 Attribute Properties

You can set the following kinds of properties on attributes:


- "Attribute Access Properties" on page 39
- "Attribute Required Properties" on page 41
- "Attribute Filter and Format Properties" on page 41
- "Attribute UI Control Properties" on page 41
- "Understanding DNLookup Attributes" on page 45


### Attribute Access Properties

**NOTE:** You can set attribute access for all of an entity's attributes selecting *DAL > Set Attribute Access*.

***Table 3-13*** *Attribute Access Properties*

| Name | Description |
|---|---|
| Edit | When selected, this attribute can be edited/modified by the user application. Even if it is selected (true), the attribute might still not be editable if the underlying Identity Vault ACLs/effective rights prevent it. |
| Enable | When deselected, this attribute cannot be used by the user application. It is the same as removing the entry from the file. |
| Hide | Controls whether the Hide check box in the user application is enabled or disabled. The Hide check box allows users to control whether an attribute (such as their photo) is displayed by the application.<br><br>When deselected, the Hide check box is disabled for this attribute, so the user cannot choose to hide this attribute.<br><br>When selected, the Hide check box can be enabled in the user application. However, the following must also be true of the logged-in user. They:<br><br>• Are either the owner of the attribute or a User Application Administrator.<br>• Have Trustee rights to update the srvprvHideAttributes attribute on the Identity Vault.<br><br>If these requirements are not met, then the Hide check box is disabled in the user interface even if this setting is selected (true).<br><br>**TIP:** When a user hides an attribute that contains an image, users who have viewed the image might continue to see it until their browser cache is refreshed. |
| Multivalue | Specifies whether this attribute can be multivalued, for example, a phone number.<br><br>When selected, the attribute can be multivalued. |
| Read | When selected, the user application can query this attribute. For most attributes this should be selected (true), but for some attributes, like password, it should be deselected. |
| Require | When selected, the attribute must be supplied. |
| Search | When selected, the user application can search on this attribute. Attributes that are used in queries by identity portlets (such as Entity Search List or Entity Org Chart) or request and approval forms must be selected.<br><br>**TIP:** If an attribute used in a search is also indexed in eDirectory, the search is faster. |
| View | When selected, the user application can display this attribute. In most cases this is selected, but for attributes like password, it should be deselected. If you specify it in a request or approval form, view must be selected. |

### Attribute Required Properties

***Table 3-14*** *Attribute Required Properties*

| Property Name | Description |
|---|---|
| Key | The unique identifier for the attribute. |
| Display Label | The label that is displayed in the user application. |
| Attribute Name | The eDirectory name for this attribute. |
| LDAP Name | The LDAP name for this attribute. |

### Attribute Filter and Format Properties

***Table 3-15*** *Attribute Filter and Format Properties*

| Property Name | Description |
|---|---|
| Filter: WHERE Attribute | Lets you specify an LDAP filter on the Identity Vault search for this attribute. |
| Enable | When selected, the filter is enabled. |

### Attribute UI Control Properties

***Table 3-16*** *Attribute UI Control Properties*

| Property Name | Description |
|---|---|
| Data Type | Choose a data type from the following list:<br>• Binary<br>• Boolean<br>• DN<br>• Integer<br>• LocalizedString<br>• String<br>• Time |

| Property Name | Description |
| --- | --- |
| Format Type | Used by the user application to format data. Format types include:<br>• None<br>• AOL IM<br>• Email<br>• Groupwise IM<br>• Image<br>• Phone Number<br>• Yahoo IM<br>• Image URL<br>• Date<br>• DateTime<br>The Format Types are dependent on the data type. For example, a Time data type can only be associated with Date and DateTime formats. |

| Property Name | Description |
|---|---|
| Control Type | Types include:<br><br>**DNLookup**—Defines that this attribute contains a DN reference. Use when you want to:<br><br>• Populate a list with the results of a DN search among related entities.<br>• Maintain referential integrity across DN referenced attributes during updates and deletes.<br>• Use the attribute in an object selector dialog box. Object selectors are used by certain identity portlets, such as detail, and are also availabl to the form controls you can define for provisioning request and approval forms.<br><br>The user application uses this information to generate special user interface elements (such as an object selector), and to perform optimized searches based on the DNLookup definition.<br><br>For more information on defining this property, see the "DNLookup Property Reference" on page 44. For more information on the object selector dialog box, see Section 6.6.2, "Controlling the Object Selector," on page 104.<br><br>**Global List**—Display this attribute as a drop-down list whose contents are defined in a file outside of this attribute definition.<br><br>For more information, see Section 3.3, "Working with Lists," on page 29.<br><br>**Local List**—Display this attribute as a drop-down list whose contents are defined with this attribute. To define a local list:<br><br>1. With the attribute selected, set the control type to Local List.<br><br>  <br><br>2. Click the *Add* button to add more values. Use the up-arrow and down-arrow buttons to change the position of the item in the list.<br><br>In the *Value* column, type the value to write to the Identity Vault. It can include letters, numbers, and underscore (_) character.<br><br>3. In the *Labels* column, type the text you want displayed in the user interface.<br><br>**Range**—Use the Range control type with Integer data types to restrict user input to a sequential range of values. You supply the range's start and end values. |

***Table 3-17*** *DNLookup Display Properties*

| Property Name | Description |
|---|---|
| Lookup Entity | The name of the entity to search, for example, the Task Group entity contains an attribute for Task Manager. To populate that field, you'd need to know which users are Task Managers. |
| Lookup Attributes | Choose one or more attributes to display when a search is performed. |
| Perform Automatic Query | Defines how the Lookup Attributes are displayed.<br>• When selected, the form or portlet performs an automatic query of the entity and presents the results in a selectable list. This option is not recommended if a large amount of data can be returned because it forces the user to scroll through a large result set.<br>• When deselected, allows the user to specify the search criteria for the entity query, then presents the results in a selectable list. |

***Table 3-18*** *DNLookup Detail Properties*

| Property Name | Description |
|---|---|
| Detail entity | The key of the entity whose details you want displayed if the user requests more information by clicking a hypertext link in the user application. When you define a DNLookup, the identity portlets are able to provide a hypertext link that allows users to display the details of the linked object. |

The DNLookup Relational Integrity properties are used for synchronizing data between two objects such as groups and group members.

***Table 3-19*** *DNLookup Relational Integrity Properties*

| Property Name | Description |
|---|---|
| Source Attributes to Update | Name of the attribute to update. The attribute must contain a DN reference to the *Target Attributes to Update*. This is required to synchronize attributes on two different objects. |
| Target Attributes to Update | Name of the attribute that must be updated along with the Source Attributes to Update. This is an LDAP attribute name. This is required to synchronize attributes on two different objects. The attribute must contain a DN reference. |
| Target Auxiliary Classes Needed, if any | Name of the auxiliary class that contains the *Target Attributes to Update*. |

### Understanding DNLookup Attributes

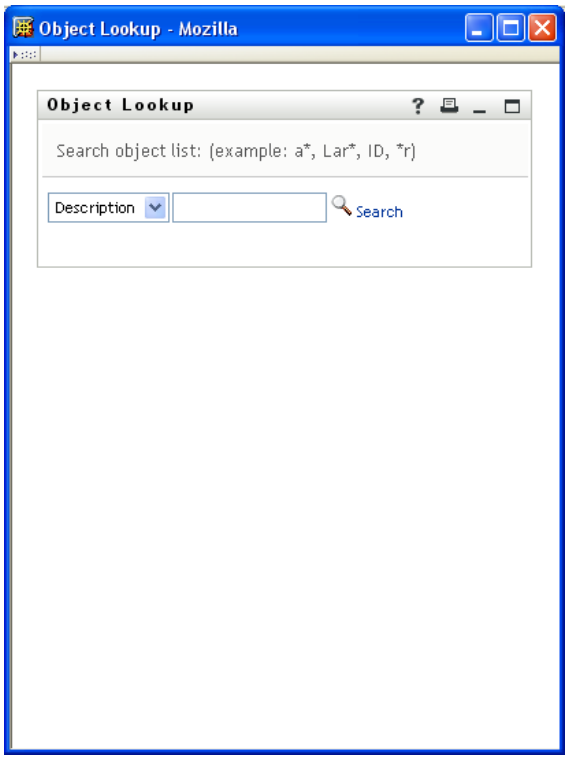
When you define an attribute as a DNLookup control type, it means that:

- This attribute can be used in an object selector dialog which allows users to select from a list of possible values when searching on this attribute.
- When this attribute is created, populated, or deleted through the user application, an attribute on a related entity is updated appropriately depending on the user action (create, delete, update) to maintain referential integrity.

#### DNLookups for Object Selectors

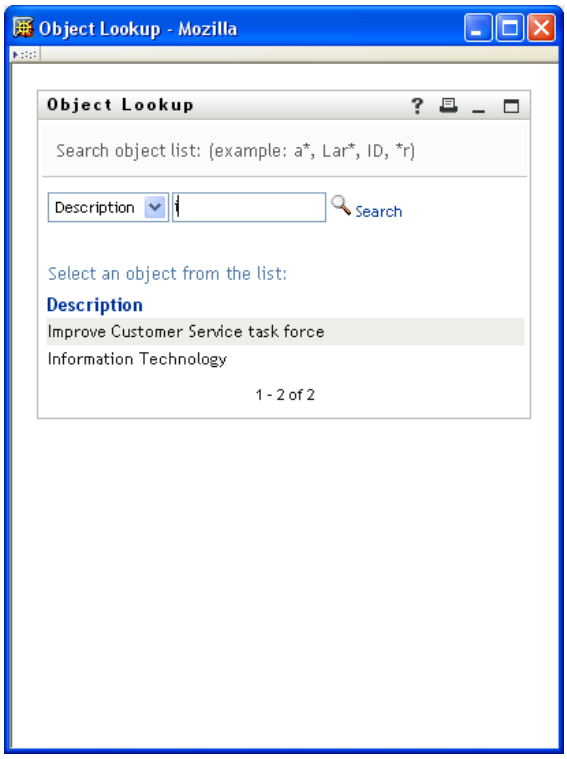
The DNLookup Display properties for a particular attribute define the contents of the object selectors in the user application. Object selectors are displayed by the Identity Self-Service portlets and in workflow request and approval forms. They provide a convenient way for users for users to search and select objects that represent DNs (such as users or groups). The object selector displays a drop-down list of attributes; the user can select one of the attributes and then enter search criteria for that attribute. In this example, the user searches for groups by group description.

***Figure 3-2*** *Sample Object Selector*



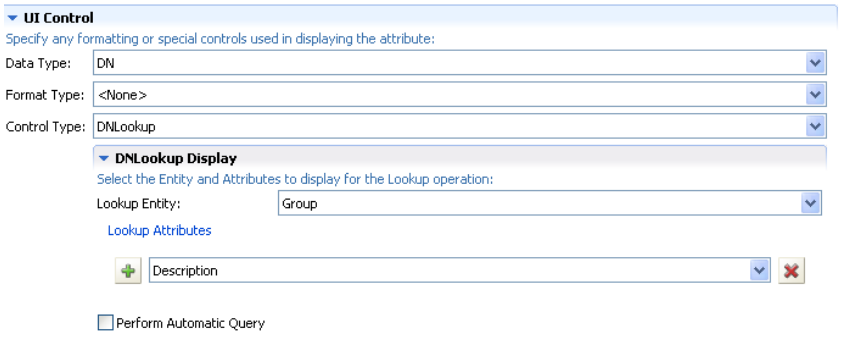
The result of the users selection looks like this:

***Figure 3-3*** *Sample Object Selector Results*



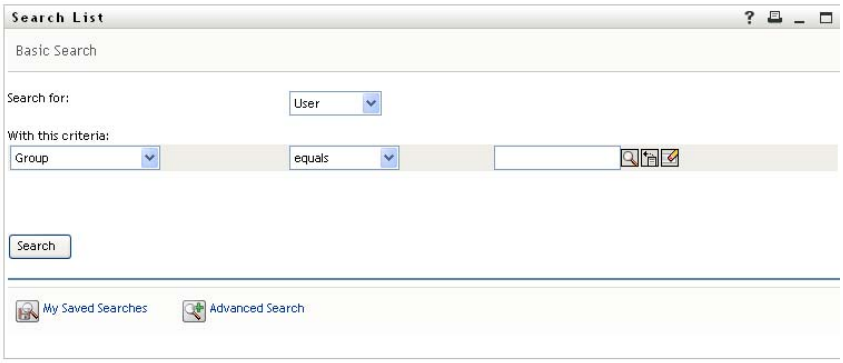

The DNLookup display properties control the contents of the object selector and the result set. The object selector, shown above, displays this way because it was based on the group attribute of the user entity. The group attribute is defined as a DNLookup control type as shown here:

***Figure 3-4*** *Group DNLookup Definition*

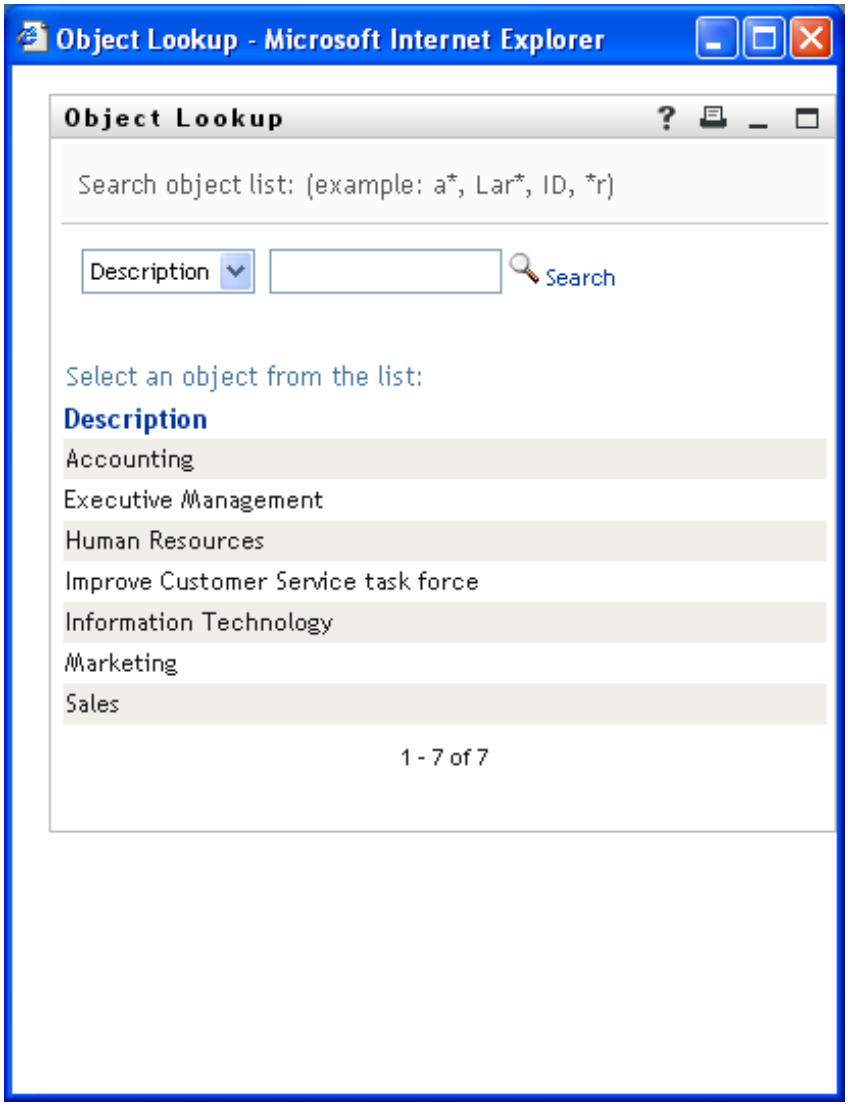



This definition also controls the way identity portlets provide a selection lists of groups for a user. For example, a user might choose to do a Directory Search to find a user in a group, but the group

name is unknown. The user would select User as the object to search for and select group as the search critieria like this:




Because the members attribute is a DNLookup for the user entity, the *Lookup* icon displays. If the user selects it, then a list of possible groups displays.



The user can select a group from the list and all of the members of that group are displayed.

**NOTE:** When the Perform Automatic Query property is not selected (false), the object selector is not populated when first displayed to the user and the user must enter selection criteria. The example above illustrates the object selector that displays when the Perform Automatic Query property is selected (true).

### DNLookups for Referential Integrity

DNLookups for updates and synchronization are important because LDAP allows group relationships to map in both directions. For example, your data might be set up so that:

- The User object contains a group attribute. The group attribute is multi-valued and lists all of the groups to which a user belongs.
- The Group object contains a user attribute. The user attribute is multi-valued and lists all of the users that belong to the group.

This means that you can have an attribute on the user object that shows all the groups a user belongs to, and on the Group object you have a DN attribute that includes all the members of that group.

When the user requests an update, the user application must honor the relationships and ensure that the target and source attributes are synchronized. In the DNLookup, you specify both attributes that must be synchronized. You can use this technique to provide synchronization between any objects that are related not just group structural objects. Create this kind of DNLookup control type by specifying the advanced DNLookup properties described in the *DNLookup Relational Integrity properties* reference.

## 3.7.3 Relationship Properties

***Table 3-20*** *Relationship Properties*

| Property Name | Description |
|---|---|
| Key | The read-only unique identifier for the relationship.<br><br>**TIP:** You specify this value in the Org Chart Portlet preference sheet. |
| Display Label | Specify a name to display when this relationship is displayed in the user application. For example, this value is displayed when users click *Choose Org Chart* from the Detail portlet.<br><br>Click *Localize* to provide the translation for the display label text. |
| Parent Entity | Choose an entity from the drop-down list.<br><br>The entity that you choose becomes the parent object in the organization chart hierarchy. In a Manager-Employee relationship, the Parent Entity is User. For a Group-Member relationship, the Parent Entity is Group.<br><br>Directory abstraction layer requirements—The entities in this list are a subset of the entities defined in the directory abstraction layer. Parent entities must have the view access property selected (true). |

| Property Name | Description |
|---|---|
| Parent Attribute | Choose an attribute from the drop-down list.<br><br>This attribute is used to find matching child entities. When the value of this attribute matches a corresponding value on an attribute of the child entity (see Child Attribute below), then a relationship can be established.<br><br>Directory abstraction layer requirements—This list of attributes is populated using the selected Parent Entity's attributes. It includes only the attributes defined as the DNLookup control type |
| Child Entity | Choose the entity for the child object in the hierarchy. In a Manager-Employee relationship, it is user. For an Employee-Resources relationship, it is Devices.<br><br>This entity must contain the attribute that is related to the Parent attribute. |
| Child Attribute | Choose the attribute that matches the Parent Attribute.<br><br>This is the child entity's attribute used to find matching parent entities. When the value of this attribute matches a corresponding value on the parent entity (see Parent Attribute above), then a relationship can be established. |

**NOTE:** The Org Chart portlet does not fully support dynamic groups; you cannot define a dynamic group as the Parent entity, but you can define a dynamic group as the child entity.

# Working with the Provisioning Request Definition Editor

# 4

This section provides general guidelines for using the provisioning request definition editor. Topics include:



## 4.1 About the Provisioning Request Definition Editor

The provisioning request definition editor allows you to create custom provisioning request definitions by using a rich set of Eclipse-based design tools. The provisioning request definition editor lets you define the basic characteristics of the provisioning request, design the associated workflow, and model the initial request and approval forms.

Identity Manager ships with a set of provisioning request *templates* that you can use to create your definitions. The templates model some common workflow design patterns. However, if you want complete control over the behavior of your workflows, you can create your provisioning request definitions from scratch.

---

**NOTE:** For details on using the templates, see Section 4.4, “Working with the Installed Templates,” on page 63.

---

### 4.1.1 How the Provisioning Request Definition Editor Fits into the Identity Manager Architecture

A key feature of Identity Manager is *workflow-based provisioning*, which is the process of managing user access to secure resources in an organization. These resources can include digital entities such as user accounts, computers, and databases. Provisioned resources are mapped to Identity Manager entitlements or to entities in the directory abstraction layer.

Identity Manager can service a wide range of *provisioning requests*. Provisioning requests are user or system actions intended to grant or revoke access to organizational resources. They can be initiated directly by the end user through the Identity Manager user application, or indirectly in response to events occurring in the Identity Vault (eDirectory™).

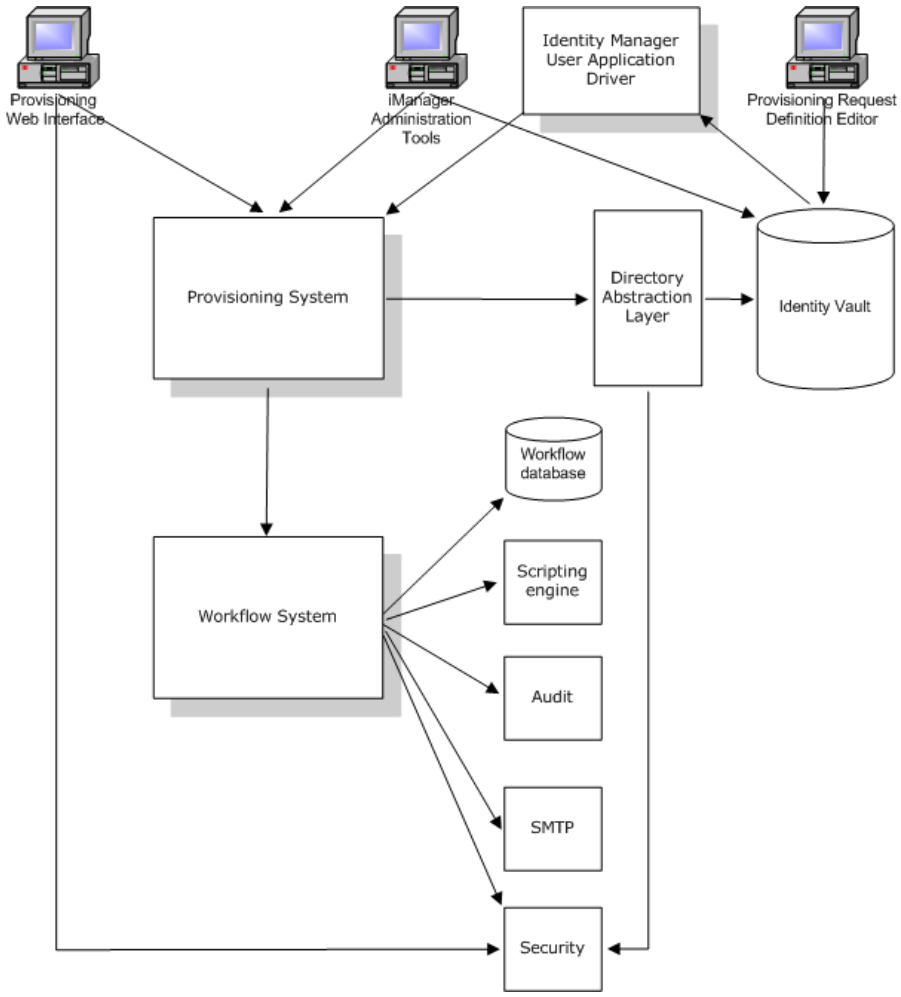
When a provisioning request requires permission from one or more individuals in an organization, the request starts a workflow. The workflow coordinates the *approvals* needed to fulfill the request. Some provisioning requests require approval from a single individual; others require approval from several individuals. In some instances, a request can be fulfilled without any approvals.

Some workflows require that processing proceed in a *sequential* fashion, with each approval step being performed sequentially. Other workflows provide support for *parallel* processing. When you define a provisioning request, you specify whether you want the workflow to support sequential or parallel processing.

To configure a provisioning request, you create a *provisioning request definition*, which binds a resource to a workflow. Identity Manager provides the provisioning request definition editor to give the designer complete control over the behavior of a provisioning request and its associated workflow. Identity Manager also includes a set of iManager plug-ins that the administrator can use to customize provisioning request definitions that have already been deployed. The iManager tools let the administrator make minor changes to the behavior of a provisioning request definition and also manage workflows that are in process.

Figure 4-1 on page 52 shows how the provisioning request definition editor fits into the workflow-based provisioning system included with Identity Manager:

***Figure 4-1*** *Provisioning Request Definition Editor and the Workflow Architecture*




## 4.1.2 Provisioning and Workflow Example

Suppose a user needs an account on an IT system. To set up the account, the user initiates a request through the Identity Manager user application. This request starts a workflow, which coordinates an

approval process. When the necessary approvals have been granted, the request is fulfilled. The process includes four basic steps:



### Step 1: Initiating the Request

In the Identity Manager user application, the user browses a list of resources by *category* and selects one to provision. In the Identity Vault, the *provisioned resource* selected is associated with a provisioning request definition. The provisioning request definition is the most prominent object in a provisioning system. It binds a provisioned resource to a workflow, and acts as the means by which the workflow process is exposed to the end user. The provisioning request definition provides all the information required to display the initial request form to the user, and to start the flow that follows the initial request.

In this example, the user selects the New Account resource. When the user initiates the request, the Web application retrieves the initial request form and the description of the associated *initial request data* from the Provisioning System, which gets these objects from the provisioning request definition.

When a provisioning request is initiated, the Provisioning System tracks the initiator and the recipient. The *initiator* is the person who made the request. The *recipient* is the person for whom the request was made. In some situations, the initiator and the recipient can be the same individual.

Each provisioning request has an *operation* associated with it. The operation specifies whether the user wants to *grant* or *revoke* the resource.

### Step 2: Approving the Request

After the user has initiated the request, the Provisioning System starts the workflow process. The *workflow process* coordinates the approvals. In this example, two levels of approvals are required, one from the user's manager, and a second from the manager's supervisor. If approval is denied by any user in a workflow, the flow terminates and the request is denied.

Workflows can process approvals in a sequential manner, or in a parallel manner. In a *sequential workflow*, as shown in Figure 4-2, each approval task must be processed before the next approval

task begins. In a *parallel workflow*, as shown in Figure 4-3, users can work on the approval tasks simultaneously.

***Figure 4-2*** *Sequential Workflow with Two Approvals*

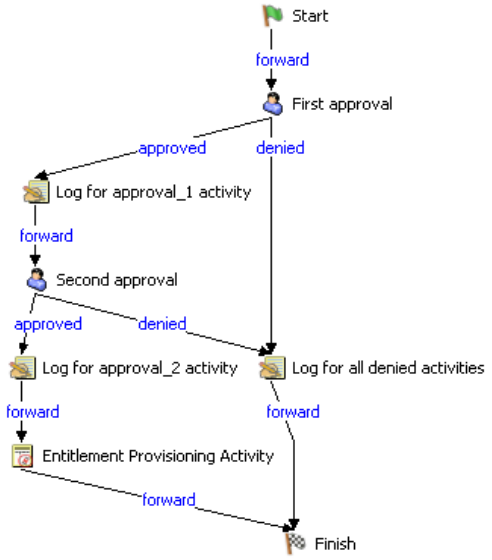
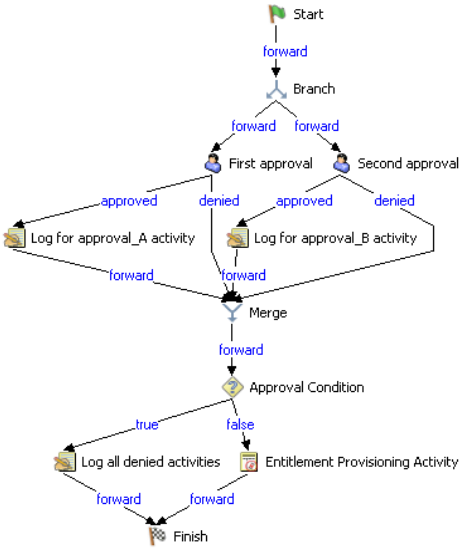



***Figure 4-3*** *Parallel Workflow with Two Approvals*




---

**NOTE:** The display labels (First approval, Second approval, and so on) can easily be changed to suit your application requirements. For parallel flows, you might want to specify labels that do not imply sequential processing. For example, you might want to assign labels such as One of Three Parallel Approvals, Two of Three Parallel Approvals, and so on.

---

The workflow definition is made up of the components shown in Table 4-1 on page 55:

***Table 4-1*** *Workflow Definition Components*

| Process Components | Description |
|---|---|
| Activities | An activity is an object that represents a task. An activity can present information to the user and respond to user interactions, or perform background functions that are not visible to the user.<br><br>In a workflow diagram, the activities are represented by boxes.<br><br>In the Identity Manager user application, the activities that handle the approval process are referred to as tasks. An end user can see the list of tasks in his or her queue by clicking *My Tasks* in the *My Work* group of actions. To see which workflow activities have been processed for a particular task, the user can select the task and click the *View Comment History* button on the Task Detail form.<br><br>To see which workflow activities have been processed for a particular provisioning request, the user can click *My Requests*, select the request, and click the *View Comment and Flow History* button on the Request Detail form.<br><br>For more information on the *My Tasks* and *My Requests* actions, see the Identity Manager User Application: User Guide (http://www.novell.com/documentation/idm/index.html). |
| Flow paths | Flow paths are what tie the activities in a workflow together. A flow path represents a path to be followed between two activities.<br><br>An activity can have multiple incoming flow paths and multiple outgoing flow paths. When an activity has more than one outgoing flow path, the flow path selected often depends on the outcome of the activity. The outcome is the end result of processing performed by the activity. For example, an approval activity can have an outcome of approved or denied, depending on the action taken by the user.<br><br>In a workflow diagram, the flow paths are represented by arrows. |

**Start activity:** The workflow process begins with the execution of the Start activity. This activity displays the initial request form to the user. Once the user has provided the initial request data, it initializes a work document using this data. The Start activity also binds several system values such as the initiator and recipient, so that these can be used in script expressions.

**Approval activities:** After the Start activity finishes execution, the Workflow System forwards processing to the first Approval activity in the flow. The Approval activity sends an e-mail to the approver, notifying this user that their attention is needed. When the user claims the task, the Approval activity displays an approval form, which gives the user the ability to act on the request. In the workflow examples shown above, “First approval” and “Second approval” are examples of Approval activities. The display labels for Approval activities can be localized to satisfy international requirements.

An Approval activity has five possible outcomes, each represented by a different flow path exiting the activity:

- Approved
- Denied
- Refused

- Error
- Timeout

**NOTE:** The Error and Timeout outcomes can occur without any action being taken by the user.

If the user approves the request, the workflow follows the approved flow path to the next activity in the flow. If no further approvals are needed, the resource can be provisioned. If the user denies the request, the workflow follows the denied flow path to the next activity in the flow. Alternatively, the user can reassign the task (if he or she is an Organizational Manager or User Application Administrator), which puts the task in another user's queue.

The user to whom an Approval activity has been assigned is referred to as the addressee. The addressee for an activity can be notified of the assigned task via e-mail. To perform the work associated with the activity, the addressee can click the URL in the e-mail, find the task in the work list (queue), and claim the task.

The addressee must respond to an Approval activity within a specified amount of time, or the activity times out. Typically the timeout interval is expressed in hours or days, to allow the user sufficient time to respond.

When an activity times out, the workflow process might try to complete the activity again, depending on the retry count specified for the activity. In some situations, the workflow process might be configured to escalate an activity that has timed out to another user. In this case, the activity is reassigned to a new addressee (the user's manager, for example) to give this user an opportunity to finish the work of the activity. If the last retry times out, the activity might be marked as approved or denied, depending on how the workflow was configured.

**Log activity:** The Log activity is a system activity that writes messages to a log. To log information about the state of a workflow process, the Workflow System interacts with Novell® Audit. During the course of its processing, a workflow might log information about various events that have occurred. Users can use the Novell Audit reporting tools to look at logging data.

**Condition activity:** During the course of execution, a workflow process might perform a test and check the outcome to see what to do next. The Condition activity provides this capability. Condition activities use a scripting expression to define the condition to evaluate. In the workflow examples shown above, "Approval Condition" is an example of a Condition activity.

The Condition activity supports three possible outcomes or exit paths:

- True
- False
- Error

**Branch and Merge activities** In a workflow that supports parallel processing, the Branch activity allows two users to act on different areas of the work item in parallel. Once the users have completed their work, the Merge activity synchronizes the incoming branches in the flow.

### Step 3: Fulfilling the Request

When a provisioning request has been approved, the Workflow System can begin the provisioning step. At this point, control passes back to the Provisioning System.

To fulfill the provisioning request, the Provisioning System can execute an Identity Manager entitlement or directly manipulate an eDirectory object and its attributes. These actions are performed by either the Entitlement activity or the Entity activity.

**Entitlement activity:** The Entitlement activity fulfills the provisioning request by granting or revoking an entitlement. This activity is not normally executed unless all of the necessary approvals had been given.

**Entity activity:** The Entity activity fulfillls the provisioning request by directly manipulating an eDirectory object and its attributes. This activity is not normally executed unless all of the necessary approvals have been given.

### Step 4: Completing the workflow

When all other activities have terminated, the workflow executes the Finish activity.

**Finish activity:** The Finish activity is the final activity in a workflow. When all the activities in a flow have been completed and the final result of the flow is available, the Finish activity is executed. The Finish activity sends a final e-mail notification to inform participants of the completion of the workflow.

## 4.2 Basic Steps for Creating a Provisioning Request Definition

Table 4-2 walks you through defining a provisioning request.

***Table 4-2*** *Basic Steps for Defining a Provisioning Request*

| Task | Action | For More Information |
|---|---|---|
| Step 1: Use the wizard to create the provisioning request definition | Provide a name for the provisioning request and define its basic characteristics. Then, specify whether you want to use a template to create the request.<br><br>Provisioning request definitions are stored locally in the `Provisioning\AppConfig\RequestDefs` directory within your workspace. | See Chapter 5, “Creating a Provisioning Request Definition,” on page 67. |
| Step 2: Create the forms | Create the initial request and approval forms for the workflow. By creating the forms first, you can ensure that the user interface is correct before proceeding to the implementation details. In addition, you can greatly simplify the process of mapping the form fields to the application data. | See Chapter 6, “Creating Forms for a Provisioning Request Definition,” on page 77. |
| Step 3: Create the workflow diagram | Add the activities to the workflow diagram and connect them with flow paths. | See Chapter 7, “Creating the Workflow for a Provisioning Request Definition,” on page 109 |

| Task | Action | For More Information |
|---|---|---|
| Step 4: Configure the activities and flow paths | Specify the properties, data item mappings, and e-mail notification settings for the activities. Then, define the semantics for the flow paths. | See Chapter 8, “Configuring the Workflow Activities and Flow Paths,” on page 119 |

# 4.3 Guidelines for Creating Workflows

To create well-formed workflows, you need to understand the rules for adding activities and flow paths. In addition, you need to understand how to manipulate workflow data. These topics are discussed below:

- Section 4.3.1, “Rules for Activities,” on page 58
- Section 4.3.2, “Rules for Flow Paths,” on page 58
- Section 4.3.3, “Understanding Workflow Data,” on page 60

**NOTE:** You can validate a provisioning request definition before you deploy it. For more information, see Section 2.5, “Validating Provisioning Objects,” on page 17.

## 4.3.1 Rules for Activities

When you’re adding activities to a workflow, you need to follow these rules:

- A workflow must have only one Start activity and only one Finish activity.
- A workflow can have zero or more of the following activity types:

  Approval activity
  Branch activity
  Condition activity
  Log activity
  Merge activity
- Each Branch activity must have a corresponding Merge activity.
- To ensure that the provisioning step is performed, a workflow must have at least one Entitlement or Entity activity.

## 4.3.2 Rules for Flow Paths

When you’re adding flow paths to a workflow, you need to follow these rules:

- With the exception of the Start activity, all activities can have one or more incoming flow paths. The Start activity cannot have any incoming flow paths.
- The Finish activity cannot have any outgoing flow paths.
- There can be only one flow path out of the Start activity. The flow path type must be forward.
- There can be between one and five flow paths out of the Approval activity. The valid flow path types are approved, denied, refused, timedout, and error. At runtime, only one of the flow paths is executed.

- There can be only one flow path out of the Entitlement, Entity, Log, and Merge activities. The flow path type must be forward.
- There can be two or three flow paths out of the Condition activity. The valid flow path types are true, false, and error. The true and false flow paths are required; the error flow path is optional.
- There can be one or more flow paths out of the Branch activity. The flow path type must be forward for each path. At runtime, all of the flow paths execute.

The table below summarizes the rules for adding flow paths into and out of an activity:

***Table 4-3*** *Number of Flow Paths Permitted for Each Activity*

| Activity | Inbound Paths | Outbound Paths |
|---|---|---|
| Start | 0 | 1<br>Must always be forward. |
| Approval | 1 to n | 1 to 5<br>Approved, denied, refused, timedout, or error. |
| Entitlement | 1 to n | 1<br>Must always be forward. |
| Entity | 1 to n | 1<br>Must always be forward. |
| Log | 1 to n | 1<br>Must always be forward. |
| Condition | 1 to n | 2 to 3<br>True and false are required; error is optional. |
| Branch | 1 to n | 1 to n |
| Merge | 1 to n | 1<br>Must always be forward. |
| Finish | 1 to n | 0 |

The table below summarizes which activity types can be a source or target for each of the available flow path types:

***Table 4-4*** *Flow Path Types Allowed for Each Activity*

| Activity | Forward | Approved | Denied | Refused | Timedout | True | False | Error |
|---|---|---|---|---|---|---|---|---|
| Start | Source | | | | | | | |
| Approval | Target | Source/ Target | Source/ Target | Source/ Target | Source/ Target | Target | Target | Source/ Target |

| Activity | Forward | Approved | Denied | Refused | Timedout | True | False | Error |
|---|---|---|---|---|---|---|---|---|
| Entitlement | Source/ Target | Target | Target | Target | Target | Target | Target | Target |
| Entity | Source/ Target | Target | Target | Target | Target | Target | Target | Target |
| Log | Source/ Target | Target | Target | Target | Target | Target | Target | Target |
| Condition | Target | Target | Target | Target | Target | Source/ Target | Source/ Target | Source/ Target |
| Branch | Source/ Target | Target | Target | Target | Target | Target | Target | Target |
| Merge | Source/ Target | Target | Target | Target | Target | Target | Target | Target |
| Finish | Target | Target | Target | Target | Target | Target | Target | Target |

## 4.3.3 Understanding Workflow Data

When you're creating a workflow, you can manipulate workflow data to suit the needs of your provisioning application.

The workflow uses a single process object to manage information about the process. A separate activity object is created for each activity in the workflow and form data is maintained for each activity that provides for user interaction.

The data objects associated with each user interface control on a form (text field, drop down list, and so forth) can be modified immediately prior to the execution of the corresponding activity (Start activity or Approval activity). In addition, this data can be retrieved immediately after execution of the activity. Once control has been passed to the next activity, the form control data is no longer available. For this reason, the workflow provides a special object called flowdata that allows you to define your own data items. You can add your own variables to this object to keep track of information that is important to your workflow, including form data that would otherwise be lost.

The following table summarizes the categories of workflow data:

***Table 4-5*** *Categories of Workflow Data*

| Data object | Lifetime | Editable | Creator |
|---|---|---|---|
| process | Workflow | No | System |
| activities | Workflow | No | System |
| activity forms | Activity | Yes | System and workflow designer |
| flowdata | Workflow | Yes | Workflow designer |

**NOTE:** The workflow designer is the person who creates the workflow in Designer.

The following table describes the variables for each type of object:

***Table 4-6*** *Data Variables in a Workflow*

| Object | Variable | Description |
|---|---|---|
| process | approvalStatus | The current status of the process. |
| | initiator | The distinguished name of the person who initiated the request. |
| | locale | The current locale. |
| | name | The workflow process name. |
| | recipient | The distinguished name of the intended target of the provisioned resource. |
| | requestID | The ID for the provisioning request. |
| | timestamp | The time the process was initiated. |
| *approval-activity-name* | action | The action taken by the user. |
| | addressee | The current addressee for the approval activity. |
| | name | The name of the activity. |
| | timestamp | The time that the activity was queued on the work list. |
| *form-name* | *custom-form-controls* | Any user interface control you add to a form. |
| flowdata | *custom-variables* | Any custom variables you create to hold data needed for the workflow. |
| | | If you use one of the installed templates to create your workflow, the flowdata object can have a variable called reason, which contains text copied from the reason field on the initial request form. |

You can reference these objects in ECMAScript expressions. Script expressions in a workflow can at any time refer to data items that are bound upstream in the flow. However, workflow expressions cannot refer to data items that are created downstream (because these data items don't exist yet), or to data bound on other branches in a flow that supports parallel processing (because these branches could be executing concurrently with the current activity).

### Creating New Data Items

You can create a new data item on the flowdata object by specifying a post-activity target expression on the *Data Item Mapping* tab for the Start or Approval activities. If you specify a name for a new data item in the *Target Expression* column, this automatically creates the variable. Any activity executed after this activity can then access the data item.

For example, you might want to map the form field called *reason* to the target expression flowdata.myReason. The variable myReason then becomes a new data item that is available to all activities executed later in the workflow.

### Modifying Data Items

You can modify a data item by specifying a pre-activity expression on the *Data Item Mapping* tab for the Start or Approval activities. For example, to prepend a dollar to a price, you might map the following source expression to a target form field called *Price*:

```
"$" + flowdata.get('cost')
```

When the form displays to the user, the Price data appears as follows:

```
$xx.xx
```

Another example might be computing the total cost by adding the tax to the base cost. To do this, you could map the following source expression to a target form field called TotalCost:

```
flowdata.get('cost') + flowdata.get('tax')
```

### Working with Complex Data Item Mappings

All data in the flowdata object is maintained in XML, so you can create data items in a hierarchical fashion as well. For example, suppose you have a workflow form that allows a user to ask for access to two internal systems, one for accounts payable and one for receivables. Suppose the form has (among other fields) two Yes/No fields named *Acct_Pay* and *Acct_Rec*. In the post-activity data item mappings, you might create two mappings as follows:

***Table 4-7*** *Complex Data Item Mapping Examples*

| Source Form Field | Target Expression |
|---|---|
| *Acct_Pay* | flowdata.SystemAccess/AcctPay |
| *Acct_Rec* | flowdata.SystemAccess/AcctRec |

This would create an XML element named `SystemAccess` with two child elements named `AcctPay` and `AcctRec`. One reason to structure data in this way is for clearer organization and management of data in complex work flows containing many forms and data items. To retrieve data from these hierarchies, the following syntax would be used:

```
flowdata.get('SystemAccess/AcctPay').
```

For complete details on building ECMAScript expressions, see Chapter 9, "Working with ECMA Expressions," on page 139.

### Moving Form Control Data to Flowdata

All form controls you create are automatically made available for use in pre-activity and post-activity expressions on the *Data Item Mapping* tab for the activity that uses the form. For example, suppose you want to make a user's entry data in control ACONTROL on form AFORM in AACTIVITY available for use in a subsequent activity. To do this, you would select AACTIVITY in the workflow, select the *Data Item Mapping* tab, and click the *Post Activity Mapping* radio button.

Next to the source form field ACONTROL, you would then enter a target expression in the following format:

```
flowdata.my_ACONTROL
```

Any subsequent activity in the workflow would then be able to access this data by using pre-activity source expressions such as these:

```
flowdata.get('my_ACONTROL')
```

```
flowdata.getObject('my_ACONTROL')
```

### Moving Flowdata to Form Controls

You can also move flowdata values into form controls. The simplest case is moving a single text value into a form control. For example, in the example above, suppose ACONTROL is a simple text entry field. In this case, to move it into another text entry field in an activity called ZACTIVITY, you would select ZACTIVITY in the workflow, select the *Data Item Mapping* tab, and click the *Pre Activity Mapping* radio button. Next to the target form field, you would then enter this source expression:

```
flowdata.my_ACONTROL
```

To move more complex form control data (for example, a MultiValue DN control) into another form control, you can use the getObject() expression syntax. For example, assuming ACONTROL is a MultiValue DN control, you could use this source expression:

```
flowdata.getObject('my_ACONTROL')
```

To move data into a form control, you need to be aware of type constraints. For example, you should not try to move text-based data into a numeric control, or a boolean value into a DN control.

## 4.4 Working with the Installed Templates

Identity Manager ships with a set of preconfigured provisioning request definitions and workflows. You can use these as templates for building your own provisioning system.To set up your system, you define new objects based on the installed templates and customize these objects to suit the needs of your organization.

The installed templates let you determine the number of approval steps required for the request to be fulfilled. You can configure a provisioning request to require:

- No approvals
- One approval step
- Two approval steps
- Three approval steps
- Four approval steps
- Five approval steps

You can also specify whether you want to support sequential or parallel processing, and whether you want to approve or deny the request in the event that the workflow times out during the course of processing.

Table 4-8 lists the templates included with Identity Manager.

***Table 4-8*** *Preconfigured Provisioning Request Definitions and Workflows*

| Template | Description |
|---|---|
| Self Provision Approval | Allows a provisioning request to be fulfilled without any approvals. |
| One Step Approval (Timeout Approves) | Requires a single approval for the provisioning request to be fulfilled. If an activity times out, the activity approves the request and the work item is forwarded to to the next activity. |
| Two Step Sequential Approval (Timeout Approves) | Requires two approvals for the provisioning request to be fulfilled. If an activity times out, the activity approves the request and the work item is forwarded to to the next activity.<br><br>This template supports sequential processing. |
| Three Step Sequential Approval (Timeout Approves) | Requires three approvals for the provisioning request to be fulfilled. If an activity times out, the activity approves the request and the work item is forwarded to to the next activity.<br><br>This template supports sequential processing. |
| Four Step Sequential Approval (Timeout Approves) | Requires four approvals for the provisioning request to be fulfilled. If an activity times out, the activity approves the request and the work item is forwarded to to the next activity.<br><br>This template supports sequential processing. |
| Five Step Sequential Approval (Timeout Approves) | Requires five approvals for the provisioning request to be fulfilled. If an activity times out, the activity approves the request and the work item is forwarded to to the next activity.<br><br>This template supports sequential processing. |
| One Step Approval (Timeout Denies) | Requires a single approval for the provisioning request to be fulfilled. If an activity times out, the workflow denies the request. |
| Two Step Sequential Approval (Timeout Denies) | Requires two approvals for the provisioning request to be fulfilled. If an activity times out, the workflow denies the request.<br><br>This template supports sequential processing. |
| Three Step Sequential Approval (Timeout Denies) | Requires three approvals for the provisioning request to be fulfilled. If an activity times out, the workflow denies the request.<br><br>This template supports sequential processing. |
| Four Step Sequential Approval (Timeout Denies) | Requires four approvals for the provisioning request to be fulfilled. If an activity times out, the workflow denies the request.<br><br>This template supports sequential processing. |

| Template | Description |
|---|---|
| Five Step Sequential Approval (Timeout Denies) | Requires five approvals for the provisioning request to be fulfilled. If an activity times out, the workflow denies the request.<br><br>This template supports sequential processing. |
| Two Step Parallel Approval (Timeout Approves) | Requires two approvals for the provisioning request to be fulfilled. If an activity times out, the activity approves the request and the work item is forwarded to to the next activity.<br><br>This template supports parallel processing. |
| Three Step Parallel Approval (Timeout Approves) | Requires three approvals for the provisioning request to be fulfilled. If an activity times out, the activity approves the request and the work item is forwarded to to the next activity.<br><br>This template supports parallel processing. |
| Four Step Parallel Approval (Timeout Approves) | Requires four approvals for the provisioning request to be fulfilled. If an activity times out, the activity approves the request and the work item is forwarded to to the next activity.<br><br>This template supports parallel processing. |
| Five Step Parallel Approval (Timeout Approves) | Requires five approvals for the provisioning request to be fulfilled. If an activity times out, the activity approves the request and the work item is forwarded to to the next activity.<br><br>This template supports parallel processing. |
| Two Step Parallel Approval (Timeout Denies) | Requires two approvals for the provisioning request to be fulfilled. If an activity times out, the workflow denies the request.<br><br>This template supports parallel processing. |
| Three Step Parallel Approval (Timeout Denies) | Requires three approvals for the provisioning request to be fulfilled. If an activity times out, the workflow denies the request.<br><br>This template supports parallel processing. |
| Four Step Parallel Approval (Timeout Denies) | Requires four approvals for the provisioning request to be fulfilled. If an activity times out, the workflow denies the request.<br><br>This template supports parallel processing. |
| Five Step Parallel Approval (Timeout Denies) | Requires five approvals for the provisioning request to be fulfilled. If an activity times out, the workflow denies the request.<br><br>This template supports parallel processing. |

## 4.5 Debugging a Workflow

When you're testing a workflow, you might need to see the values of the variables you're using in the flow. There are several ways to do this. One approach is to use the Log activity to display messages containing the variables you need to look at. Once you've configured the Log activity, you can then see the messages in the console. In the Log activity, you can use scripting expressions in the Message property to retrieve the values you need. For example, you might use this expression to log a message containing the value of a variable defined on the flowdata object:

```
flowdata.get('my_variable')
```

For details on using the Log activity, see Section 8.1.3, "Log Activity," on page 128.

Another approach is to look in the workflow database to see how the data associated with the flowdata object changes as the workflow progresses from one activity to the next. To see this data, you can look at the afdocument table.

A final approach you can use during the debugging process is to change the log levels associated with the workflow system (com.novell.soa.af.impl) and the provisioning requests component of the user application (com.novell.srpr.apwa). This approach may generate more information than you need, but sometimes it can be helpful. To change logging levels, go to the Logging page within the Administration tab of the user application.

# Creating a Provisioning Request Definition

5

This section provides details about creating a provisioning request definition. Topics include:



## 5.1 About the Wizard and the Overview Tab

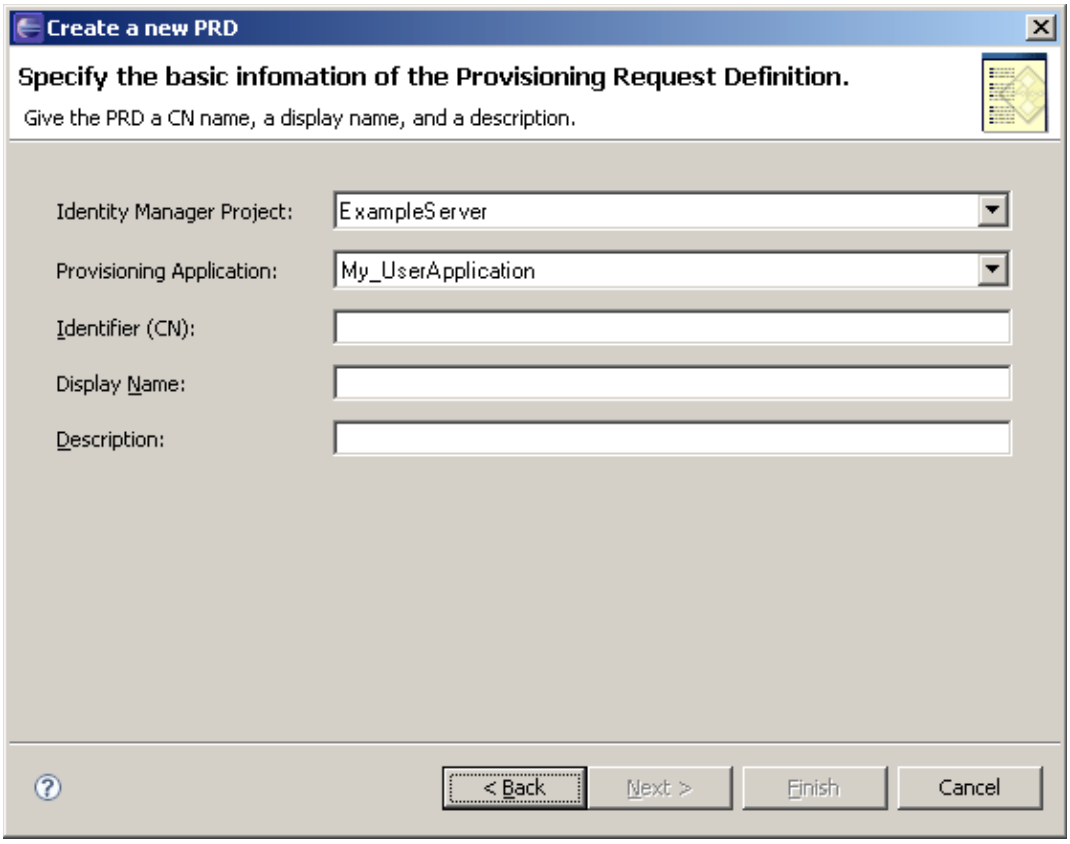
You create provisioning request definitions in three main steps:

- You create the basic information about the provisioning request definition (for example, the name of the provisioning request definition, the category to which it belongs, who can access it) using the Create A New PRD Wizard. After you have created the basic provisioning request definition, the basic information is displayed in the *Overview* tab.
- You create the forms that interact with the workflow participants using the *Forms* tab.
- You design the workflow using the *Workflow* tab.

To add a provisioning request definition:

1. Launch the Create A New PRD Wizard in one of these ways:
   - From the Provisioning view, right-click the *Provisioning Request Definitions* node and choose *New*.
   - From the Provisioning view, click a user application or provisioning request container, then select *Insert* > *Provisioning Request Definition*.
   - Select *File* > *New* > *Provisioning*.
   - Choose *Provisioning Request Definition*, then click *Next*.

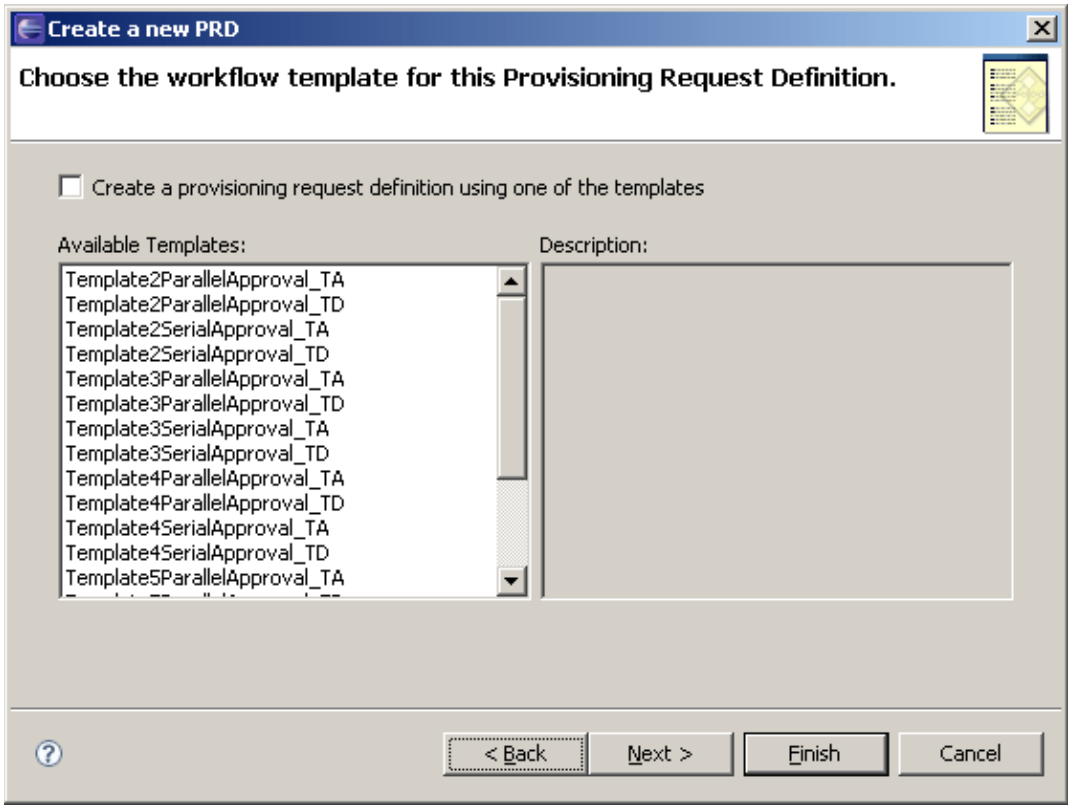
The first page of the Create A New PRD Wizard is displayed.




**2** Fill in the fields as follows:

| Field | Description |
|---|---|
| *Identity Manager Project*<br>*Provisioning Application* | Select the Identity Manager project and Provisioning Application in which you want to add the provisioning request definition.<br>**NOTE:** These fields display when you launch the wizard from the *File* menu. |
| *Identifier (CN)* | The CN identifier for the provisioning request definition. |
| *Display Name* | The display name for the provisioning request definition. This is the name that is displayed in the provisioning view. |
| *Description* | A description of the provisioning request definition. |

3 Click *Next*. The next page of the wizard is displayed.

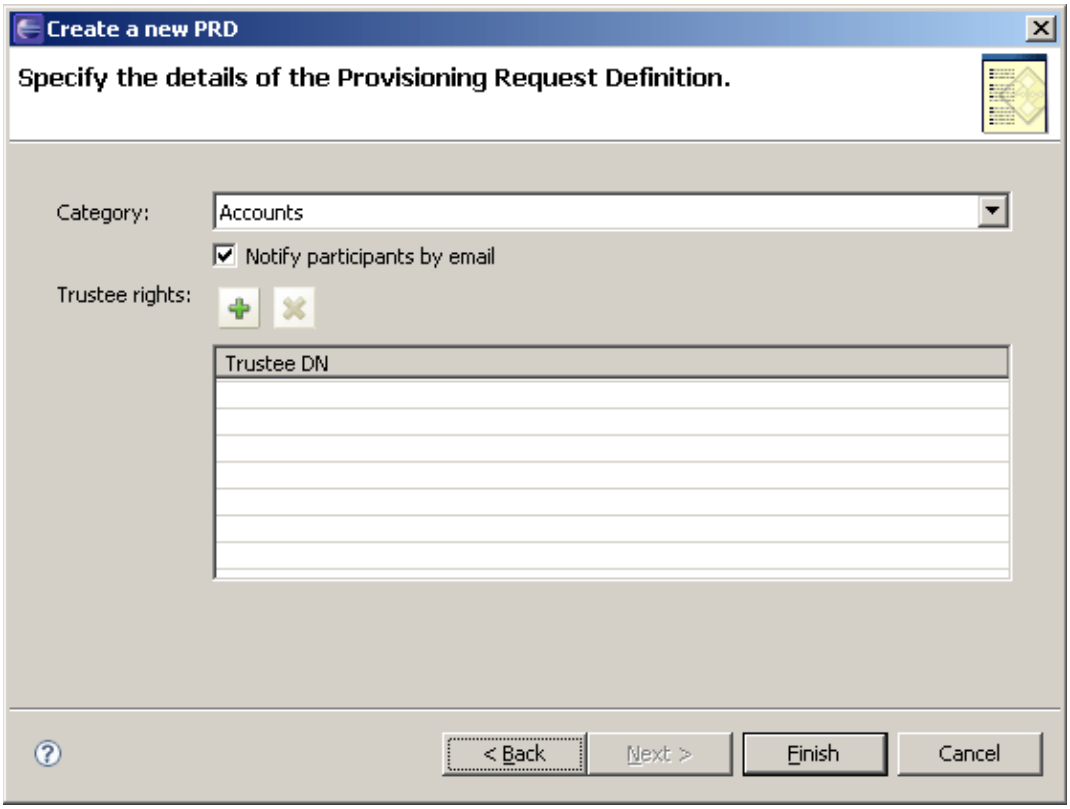



You can create new provisioning request definitions based on a template, or you can build the provisioning request definition from concept to finished product. You use the next panel of the wizard to specify whether or not to base this provisioning request definition on a template.

4 Perform one of these steps:

- If you want to base this provisioning request definition on a template, select *Create a provisioning request definition using one of the templates*, then select the desired template (for example, TemplateSingleApproval_TA) from the *Available Templates* list, then click *Next*.
- If you want to build this provisioning request definition from concept to finished product, click *Next*.

You use the next panel of the wizard to specify the trustees who can access the provisioning request definition.




5 Click the plus (+) icon to add a trustee.

Designer displays a panel that you use to browse the Identity Vault to select a trustee. You can select an individual trustee, or select a group.

6 Select the trustee, then click *OK*.

Designer returns you to the previous panel. If desired, add additional trustees by repeating the previous step. When you have finished adding trustees, click *Finish*. Designer displays the Provisioning Request Definition Details panel on the *Overview* tab (see Section 5.3, "Using the Overview Tab to Modify Basic Settings," on page 73).

## 5.2 Using the Wizard to Create a Provisioning Request Definition

You can create a provisioning request definition using a template, or from concept to finished product. We recommend that you use an existing template to create new provisioning request definitions. This saves time and allows you to make targeted changes to an existing provisioning request definition. However, if no existing provisioning request definition resembles new work that you want to do, you can create a new provisioning request from concept to finished product.

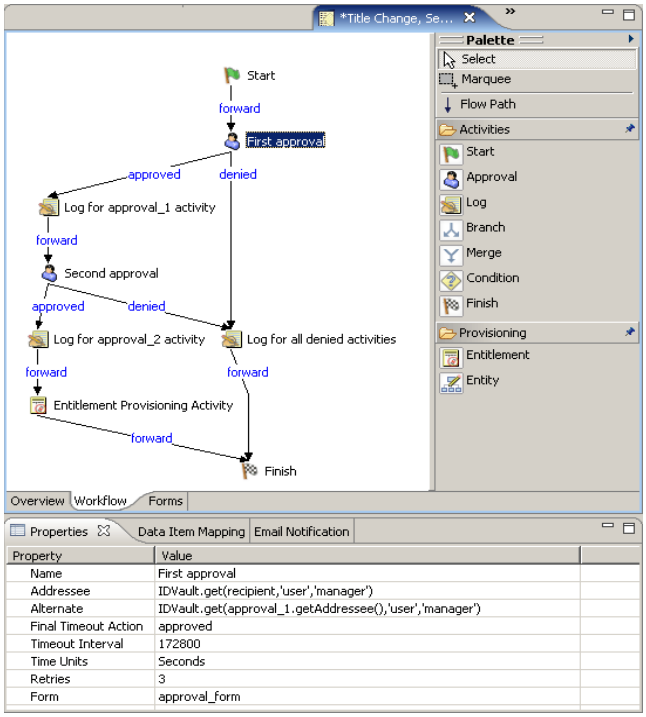
This section includes the following topics:

- Section 5.2.1, "Using a Template," on page 71
- Section 5.2.2, "From Concept to Finished Product," on page 72

## 5.2.1 Using a Template

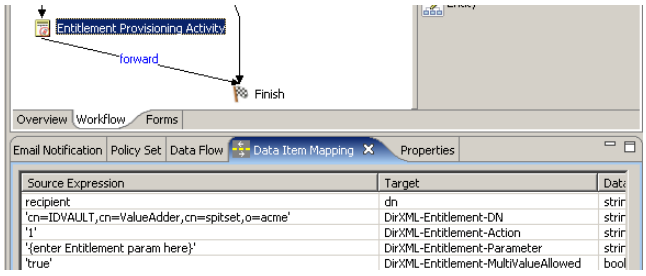
To create a provisioning request definition using a template:

1. Create the basic information for a new provisioning request definition (see Section 5.1, “About the Wizard and the Overview Tab,” on page 67). In step Step 4 on page 69, select *Create a provisioning request definition using one of the templates*, then select the desired template. When you are finished, the *Overview* tab for the new provisioning request is displayed.
2. Click the *Workflow* tab. The Workflow panel is displayed.




The provisioning request definition template includes some default values that you will want to customize for your environment. For example, the Entitlement Provisioning Activity contains placeholder values for several data item mapping properties. You need to replace the placeholder values with the actual values for your provisioning request.

3. Click the Entitlement Provisioning Activity, then click the *Data Item Mapping* tab.




4. Double-click in the *Source Expression* field for the *DirXML-Entitlement-DN* target field, then click the button that appears in the field to display the ECMA expression builder.

See Chapter 9, "Working with ECMA Expressions," on page 139 for information about the ECMA expression builder.

5 Use the ECMA expression builder to replace the placeholder expression with an expression that specifies the entitlement that you would like to provision with this provisioning request.

6 Replace the placeholder expression in the *Source Expression* field for the *DirXML-Entitlement-Parameter*.

7 Click on the *Forms* tab and customize the forms for the provisioning request to your needs.

See Chapter 6, "Creating Forms for a Provisioning Request Definition," on page 77.The template includes predefined request and approval forms. You may want to add additional forms, or to add or remove form controls.

8 Click on the *Workflow* tab and customize the properties of the workflow to your needs.

See Chapter 7, "Creating the Workflow for a Provisioning Request Definition," on page 109 and Chapter 8, "Configuring the Workflow Activities and Flow Paths," on page 119.

## 5.2.2 From Concept to Finished Product

Whenever possible, use an existing template (or save an existing provisioning request definition under a new name) to create new provisioning request definitions. This saves you time and allows you to make targeted changes to an existing provisioning request definition. However, if no existing provisioning request definition resembles the new work that you want to do, then you need to build a provisioning request definition from concept to finished product. You can still save time and effort by re-using forms from other workflows.

To create a provisioning request definition:

1 Create the basic information for a new provisioning request definition (see Section 5.1, "About the Wizard and the Overview Tab," on page 67). In step Step 4 on page 69, do not select *Create a provisioning request definition using one of the templates*, and do not select a template. When you are finished, the *Overview* tab for the new provisioning request is displayed.

2 Create the forms for the provisioning request definition. After you have created the basic provisioning request definition, the next step is to create the forms that are presented to the provisioning request users. It's important to define forms *before* you create the workflow topology, so that data bindings can be set up automatically for each activity when you create activities.

There are two types of forms:

**request:** Used by the person requesting the resource to specify the item or capability that is being requested. One request form can be defined for a workflow. The request form is always associated with the Start Activity.

**approval:** Used by the person receiving the provisioning request to approve, refuse, or comment on the provisioning request. One or more approval forms can be defined. You must associate each form with an Approval activity. You associate an approval form with an Approval activity using the properties for the activity.

To create the forms, see Section 6.3, "Creating forms," on page 80.

3 Click the *Workflow* tab and create the workflow topology.

You create the topology of a workflow by creating and linking activities into the desired workflow pattern, and by assigning rules to the flowpaths between activities. For information

about creating a workflow topology, see Chapter 7, “Creating the Workflow for a Provisioning Request Definition,” on page 109.

4 Specify the details (properties, data item mappings, e-mail notification) for each workflow activity.

   To specify workflow activity details, see Section 8.1, “Configuring Activities,” on page 119.

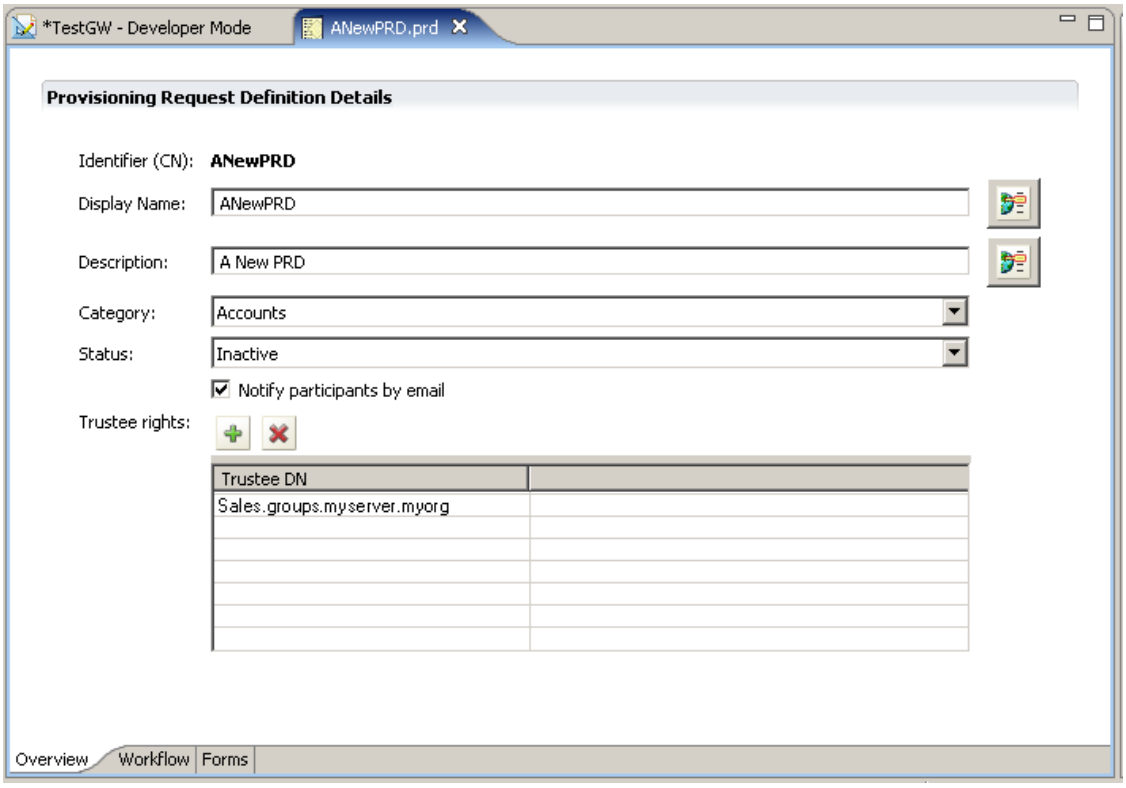
5 Configure the flowpaths between workflow activities.

   To configure flowpaths, see Section 8.2, “Configuring Flow Paths,” on page 135.

## 5.3 Using the Overview Tab to Modify Basic Settings

You use the *Overview* tab to define the basic information (for example, the name of the provisioning request definition, the category to which it belongs, who can access it) about the provisioning request definition.

***Figure 5-1*** *Overview Tab*




The following table describes each property that you can configure on the *Overview* tab.

***Table 5-1*** *Overview Properties*

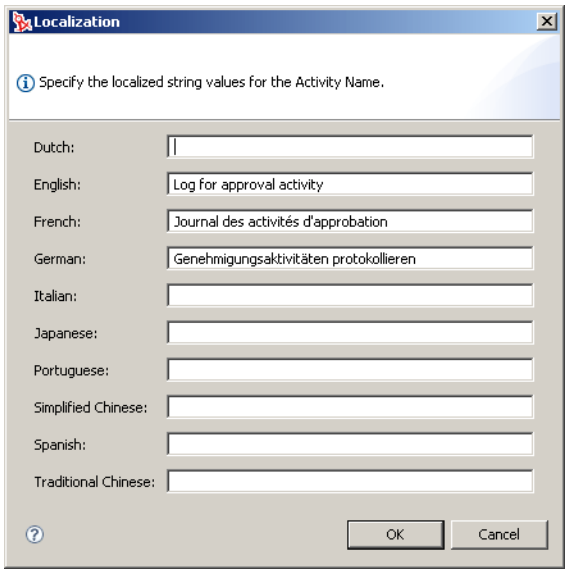
| Property | Description |
|---|---|
| *Identifier (CN)* | Displays the CN of the provisioning request definition. The CN cannot be changed. |

| Property | Description |
|---|---|
| *Display Name* | Specifies the display name of the provisioning request definition. This is the name that is displayed to the user in Designer and Identity Manager. |
| *Description* | Specifies the description of the provisioning request definition. |
| *Category* | Specifies the category to which the provisioning request definition belongs from the list of Provisioning Categories defined in the directory abstraction layer. |
| *Status* | Specifies the status of the provisioning request definition:<br><br>**Active:** Select to make the provisioning request definition available for use in the user application.<br><br>**Inactive:** Select to make the provisioning request definition temporarily unavailable for use in the user application. You can use this option when you want keep the roles of the person who develops and deploys the provisioning request definition separate from the person who activates the provisioning request definition. For example, a developer could deploy the provisioning request definition as Inactive, and an administrator could be responsible for changing the status to Active.<br><br>**Template:** Select if you want to use this provisioning request definition as the basis for other provisioning request definitions. Templates are not available for use in the user application.<br><br>**Retired:** Select to mark the provisioning request definition as permanently unavailable for use in the user application (you can still change the status of the provisioning request definition at any time). This status provides a way of keeping a historical record of a provisioning request definition that is no longer in use. |
| *Notify participants by email* | Specifies whether approvers are notified by e-mail about pending approval tasks, and whether initiators are notified by e-mail of workflow completion. If *Notify participants by email* is not checked, then users must look at the *Requests and Approvals* tab in the User Application for notifications about tasks.<br><br>For information about selecting an e-mail template and customizing e-mail template parameters, see Section 8.1.7, “Finish Activity,” on page 131. |
| *Trustee Rights* | Specifies the users and groups that can use the provisioning request definition. |

# 5.4 Localizing Display Text

You can provide localized string values for activity and form properties that are displayed to the user. Localized string values are stored within the provisioning request definition .prd file. You can provide localized string values whenever you see this button displayed in a property field:

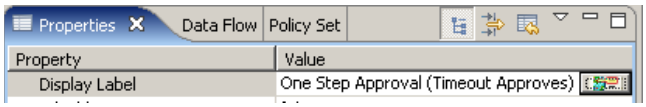
When you click this button, the Localization dialog box is displayed:




You can select the languages that are listed in this dialog box using *Preferences* (see Section 2.2, "Setting Provisioning View Preferences," on page 12).

To provide localized string values:

1. Select the item (for example, a form control or a workflow activity) for which you would like to specify localized string values.
2. Click the *Properties* tab.
3. Click in the *Value* column for a text attribute, then select the button that is displayed in the *Value* column:




   A dialog box that you use to provide localized strings is displayed.

4. Specify the localized strings, then click *OK*.

# Creating Forms for a Provisioning Request Definition

# 6

This section provides details on how you can create and customize the user application's request and approval forms. Topics include:



## 6.1 About Forms

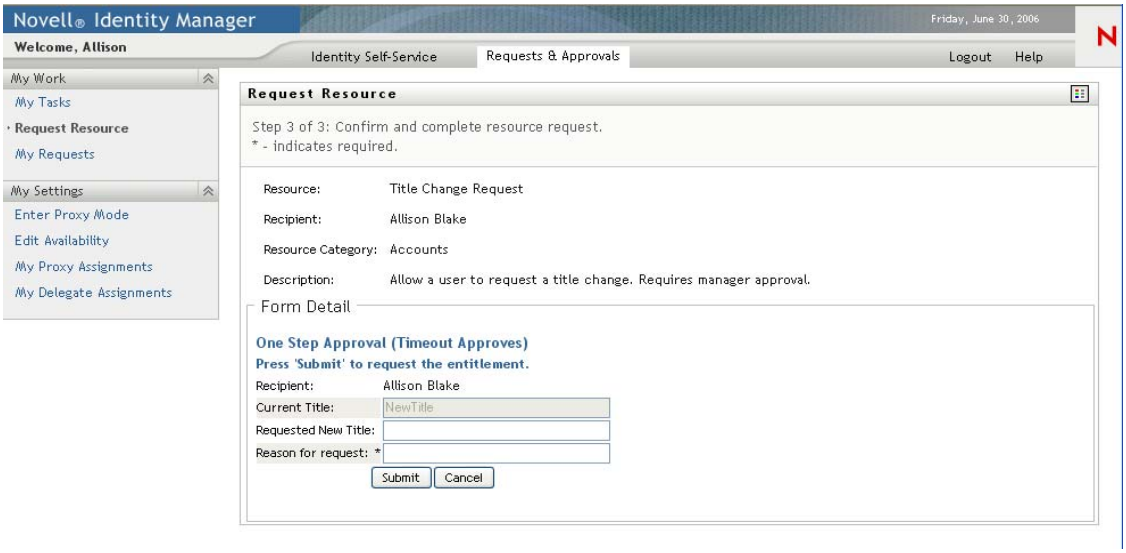
The *Forms* tab of the provisioning request definition editor lets you define the *Form Detail* section of the user application's provisioning request definition forms. You use the Forms page to define:

- Request forms—Allows users to initiate a resource request.
- Approval forms—Allows users to approve or deny resource requests.

### 6.1.1 About Request Forms

You can create one request form for a provisioning request definition.The request form is associated with the workflow's start activity.

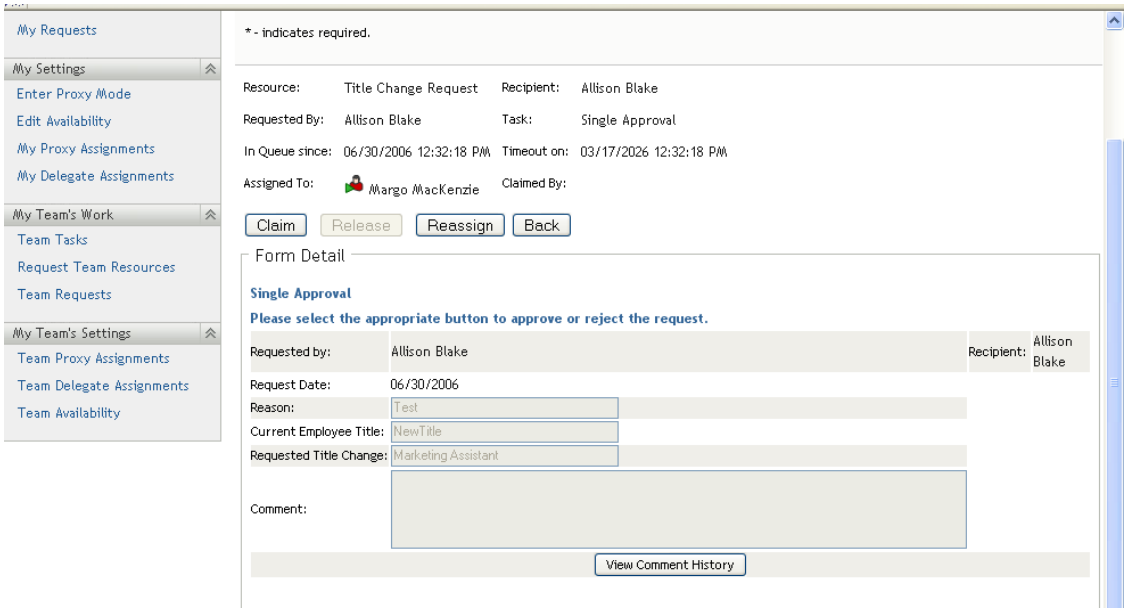
***Figure 6-1*** *Sample Resource Request Form*



### 6.1.2 About Approval Forms

You can define multiple approval forms for a provisioning request definition, but only one form per approval activity. You link an approval form to an approval activity in the properties for the activity.

***Figure 6-2*** *Sample Resource Approval Form*




### 6.1.3 About Form Control Data Binding

All of the fields you define for a form are automatically available for data binding in the Data Item Mapping property sheet. Two bindings, or mappings, are possible for each form field: a pre-activity mapping to initialize or pre-load a form field with data, and a post-activity mapping to move modified form data into the work flow data-item called flowdata. For more information on data item mappings, see Section 7.2.2, “Defining the Data Item Mappings,” on page 113.

Some form controls allow you to initialize their values from data sources other than workflow data. For example, some list controls allow you to specify the initial value as a property of the control. For more information about defining initial values, see Section 6.5, “Form Control Reference,” on page 85.

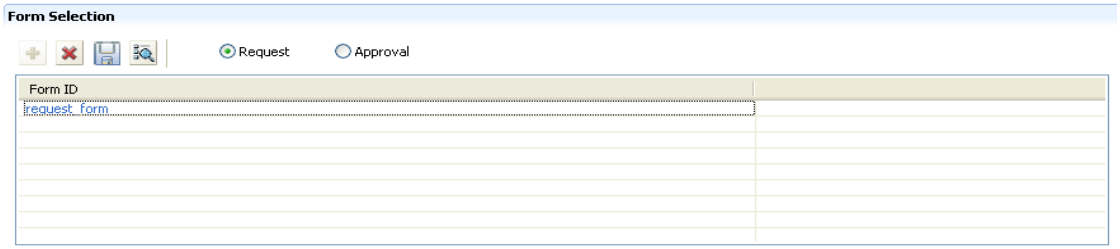
## 6.2 About the Forms Tab

You use the *Forms* tab of the provisioning request definition editor to define the appearance and behavior of your request and approval forms. The Forms page contains two sections: *Form Selection* and *Form Controls*.

## 6.2.1 About Form Selection

Use the *Form Selection* section to create, delete, or preview a form, or to create a form template.

***Figure 6-3*** *Form Selection*




The *Form Selection* toolbar contains these options:

***Table 6-1*** *Form Selection Toolbar Options*

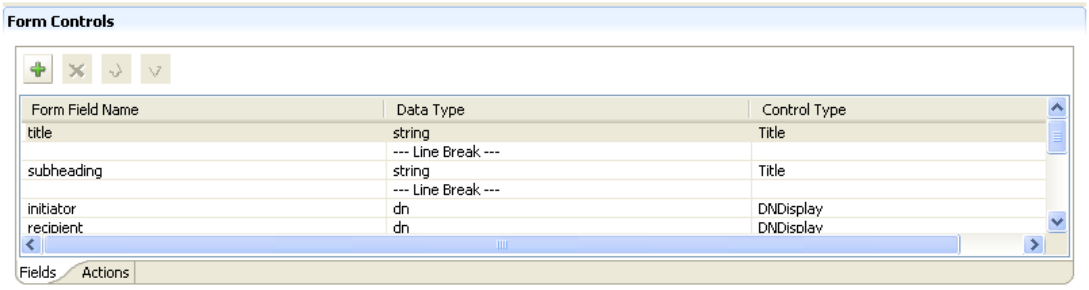
| Button | Description |
|---|---|
|  | Click to launch the New Form wizard. |
|  | Click to delete an existing form. |
|  | Click to save the form as a template. You can then base other forms on this template. Forms are saved as XML documents in the project directory.<br>Templates are available only within the project in which you create them. |
|  | Click to preview the form. |
| Request | Select to access or create the request form. |
| Approval | Select to access or create an approval form. |
| Form ID | A name to identify the form. This name is used only within the design environment. |

If you create a provisioning request definition from an existing template, and the template has forms associated with it, the *Form Selection* section displays them. You can modify the form instance using the *Form Controls* section.

### 6.2.2 About Form Controls

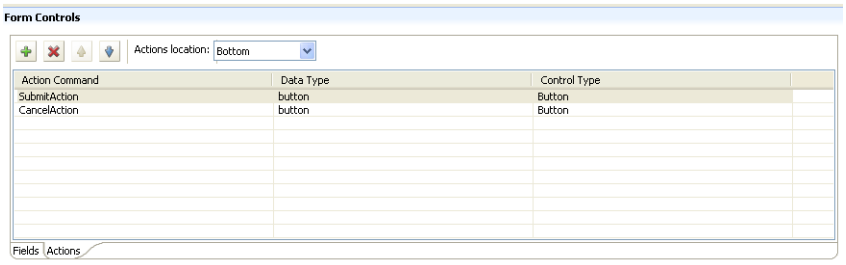
Use the *Form Controls* section to define or modify the form's appearance and behavior. Use the Fields tab to add, delete, and change the data type, control type, and layout order of the controls on the form.

***Figure 6-4*** *Fields Tab*




Define the actions the user can perform on the form in the *Actions* tab, shown below. Use the *Actions* toolbar to add, delete, and change the actions and layout order of the actions on the form.

***Figure 6-5*** *Actions Tab*




## 6.3 Creating forms

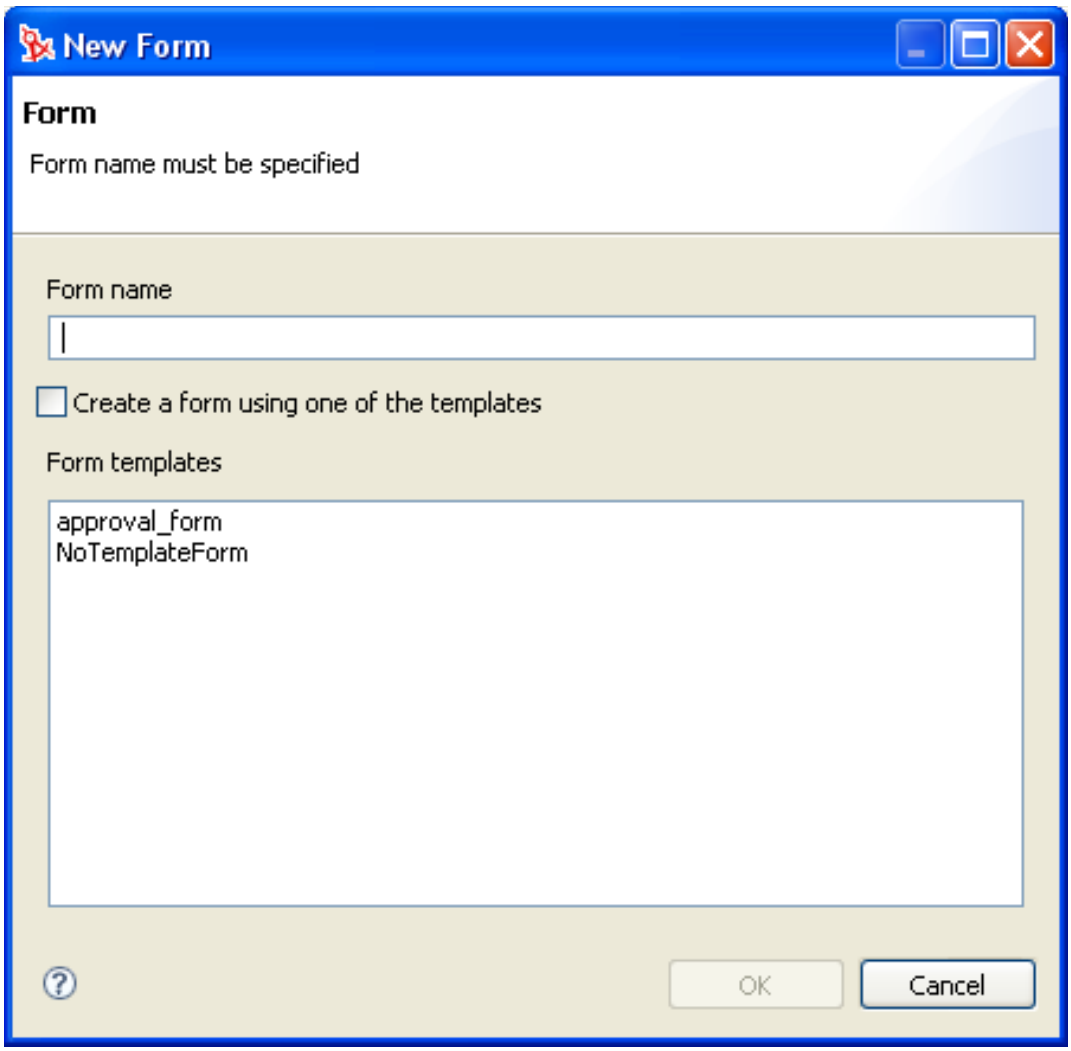
This section describes how to create new forms and add controls to it. It includes these sections:

- Section 6.3.1, "Creating New Forms," on page 80
- Section 6.3.2, "Working with Form Controls," on page 81

### 6.3.1 Creating New Forms

1 With provisioning request definition editor open, click the *Forms* tab.

2 In the Form Selection section of the page, click *Add* to access the New Form wizard:




3 Fill in the fields as follows:

| Field | Description |
|---|---|
| Form Name | Type the name of the form as you want it to appear in Designer. |
| Create a form using one of the templates | If you want to base the new form on an existing template, select this option and select one of the forms from the Form templates list. |

4 Click *OK* to save the form or *Cancel* to exit without saving.

## 6.3.2 Working with Form Controls

Use the *Form Controls* section to define the content and layout of the form.

To add a control to a form:

1. Click *Add*. Designer adds a control named `Field` to the bottom line of the form.

   If you add more than one control of the same name to the form, Designer adds a unique number to the end of the control name.

2. Define the following properties for the control:

| Field | Description |
|---|---|
| *Form Field Name* | A unique name for the field. The name is used in:<br>• The *Workflow* tab's *Data Item Mapping* dialog box.<br>• The *ECMA expression builder* dialog box<br>• An internal XML reference in the provisioning request definition file.<br>Consider the naming conventions you want to use for form fields to avoid confusion in the Data Item Mapping and ECMA expression builder dialog boxes. For example, the request and approval forms might both contain a field called *Reason*. To make it clear which field you are working with while performing data mappings, you can preface the field name with the name of the form where it is used. You might name one reason field *Req_Reason* and the other *App_Reason*. |
| *Data Type* | The field's data type. The data type determines the valid control types and the type of validation performed. |
| *Control Type* | The type of control used to display or edit the data. The selection list is filtered based on the selected data type. |

**NOTE:** Form field controls do not have Data Item Mappings or E-mail notifications property sheets.

3. For each control, specify its properties in the *Properties* tab (available via *Window* > *Show View* > *Properties*). For more information, see Section 6.5, "Form Control Reference," on page 85.

4. Click the *Actions* tab to define what the user can do with the form. For example, you can add actions that allow the user to submit a form or cancel it.

**NOTE:** A request form must have, at a minimum, a SubmitAction. Without a SubmitAction, the request will not process. It is also recommended that every form also have a CancelAction. Each approval form must have at least one action defined.

5. In the Actions page, click *Add* to add a new Action. Fill in the fields as follows:

| Field | Description |
|---|---|
| *Actions Location* | Choose the location for the action buttons you add to the form.<br><br>*Bottom*. Places the action buttons on the bottom of the form. (Default.)<br><br>*Top*: Places the action buttons on the top of the form.<br><br>*Top and Bottom*: Places the buttons at both the top and bottom of the form. |
| *Action Command* | Choose an action for the button. For more information, see Section 6.4, “Action Reference,” on page 83. |
| *Data Type* | The data type associated with the action command. Valid choices are:<br><br>Button—Adds a button to the form.<br><br>Line Break—Allows you to define the layout of your action buttons. Adding a Line Break forces the buttons to the next line. |
| *Control Type* | The visual representation of the action command and data type. Button is the only valid entry. |

### Controlling Form Layout

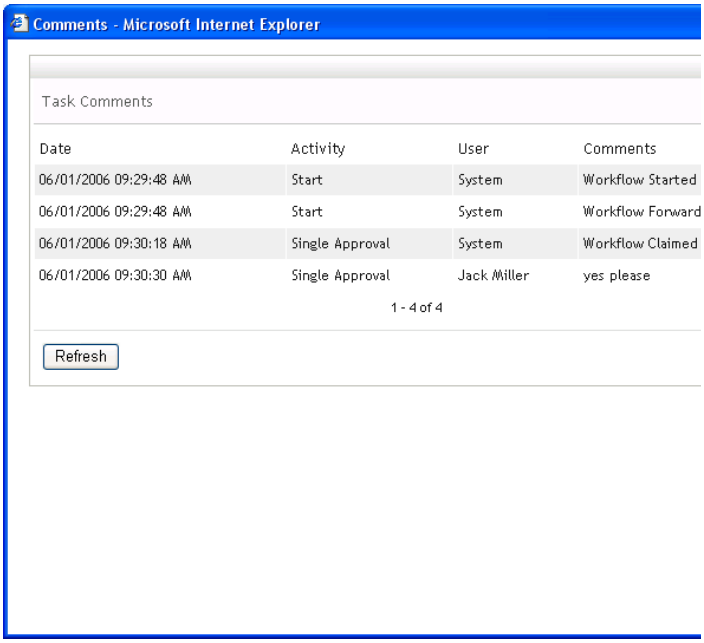
The Designer places form controls on the form top to bottom and left to right. Use the Line Break control type to force the controls to the next line of the form.

# 6.4 Action Reference

This section describes the actions you can add to forms. The actions are implemented as buttons. You can specify a custom display label for each button.

***Table 6-2*** *Valid Actions*

| Action Name | Form Type | Description |
|---|---|---|
| ApprovalAction | Approval | Causes the Approval activity to complete and follow the approved flow path. When you use this action, you must set the Hide If Read Only form property to True or the form fails validation on deploy.<br><br>**TIP:** An ApprovalAction requires the Approval Activity associated with the form to have an approved flow path exiting the activity. |
| CancelAction | Request and Approval | Cancels a request. Returns an approval to the previous state. |

| Action Name | Form Type | Description |
|---|---|---|
| CommentAction | Approval forms | Generates a button with the default label set to *View Comment History*. The button launches a Comments dialog box displaying the processing history for each activity from the workflow start to the present time. Data displayed includes: Date, Activity Name, User, and Comment as shown in the following example.<br>  <br><br>Comments are updated and persisted to the workflow database through the UpdateAction (described below).<br>**NOTE:** Any forms containing this action must also contain a field named `apwaComment`. |
| DenyAction | Approval | Causes the Approval activity to complete and follow the denied flow path. When you use this action, you must set the Hide If Read Only form property to True or the form fails validation on deploy.<br>**TIP:** A DenyAction requires the Approval Activity associated with the form to have an deny flow path exiting the activity. |
| RefusalAction | Approval | Causes the Approval activity to complete and follow the refused flow path. When you use this action, you must set the Hide If Read Only form property to True or the form fails validation on deploy.<br>**TIP:** A RefusalAction requires the Approval Activity associated with the form to have a refusal flow path exiting the activity. |
| SubmitAction | Request and Approval | Initiates the workflow and causes the workflow to execute the forward flow type. The workflow passes any user-entered data to the next activity in the workflow. |

| Action Name | Form Type | Description |
|---|---|---|
| UpdateAction | Approval | Causes the Approval activity to write a user comment to the workflow database. You typically have a text area associated with an `apwaComment`. If the user enters text in this field and clicks this action, it is persisted to the afcomment table in the workflow database. The comment can be retrieved and viewed through the CommentAction (described above). This example shows a text area and an update action button (labeled UpdateAction):<br>  <br><br>**NOTE:** The form must contain a field named `apwaComment` or the provisioning request definition fails validation.<br>For more information about `apwaComment`, see "Controls for User Entered Comments" on page 86. |

The following table describes the properties you can set on actions.

***Table 6-3*** *Action Properties*

| Property Name | Description |
|---|---|
| Display Label | Specifies the text to display on the button. |
| Visible | If true, specifies whether the action is visible at runtime. |
| Block On Error | If true, specifies that the action is blocked if any of the form's controls fail validation. This is recommended for the SubmitAction.<br>Do not set to false if the action button submits data, or invalid data can be submitted causing unexpected results. |
| Hide If Read Only | If true, specifies that the action is hidden when the form is read-only. A form can be read-only when the user opens a task without claiming it first. If your form contains the ApprovalAction, DenyAction, or RefusalAction, this property must be set to true. If it is set to false, you will encounter a validation error and will not be able to deploy it. |

# 6.5 Form Control Reference

This section describes the controls you can add to a form.

***Table 6-4*** *Control Types and Supported Data Types*

| Control Type | Data Types | | | | | | |
|---|---|---|---|---|---|---|---|
| | **Boolean** | **Date** | **Decimal** | **DN** | **Integer** | **String** | **Time** |
| DatePicker | | x | | | | | x |
| DNDisplay | | | | x | | | |
| DNLookup | | | | x | | | |
| DNMaker | | | | x | | | |
| GlobalList | | | | | | x | |
| MVCheckbox | | | x | x | x | x | |
| MVEditor | | | x | x | x | x | |
| PickList | | | x | x | x | x | |
| StaticList | x | | x | | x | x | |
| Text | | | x | x | x | | |
| TextArea | | | | | | x | |
| Title | | | | | | x | |
| TrueFalseRadiobuttons | x | | | | | | |
| TrueFalseSelectbox | x | | | | | | |

## 6.5.1 Controls for User Entered Comments

Designer supports a special internal control you can add to a form to allow users to add comments to a workflow or to view previously entered comments. They are required on forms that use *CommentAction* or *UpdateAction*. The comments are not part of the workflow data so you cannot access them via the flowdata object. The comments are special data items stored in the `afcomment` table of the workflow database. The comments are persisted as long as the row for the requestid in the afprocess table exists.

To create a form that supports user comments:

1. Add a control to your form. Select Comment as the data type. The Form Field name is automatically defined as apwaComment and the Control Type is TextArea. A single form can contain only one comment field.
2. Add a *CommentAction* or *UpdateAction* to the form.

   For more information, see Section 6.4, “Action Reference,” on page 83.

## 6.5.2 General Properties

The properties in the following table are available for each control.

***Table 6-5*** *General Properties*

| Property Name | Description |
|---|---|
| Display label | Specifies the label to display to identify the control. It is localizable. |
| Editable | Specifies if the control is editable (true). Otherwise, it displays as read-only. |
| Multivalued | This is a read-only property. It specifies if the control supports multivalue attributes (true). |
| Required | Specifies whether the control requires user input (true). |
| Tooltip | Specifies the text for the control's tooltip. It is localizable. |
| Visible | Specifies whether the control is displayed in the user interface (true). |

### Sort Order

List-based controls sort content alphabetically. For DN-based lists, the sort order is alphabetical based on the Display expression property result. For all other types, the sort order is based on the display label.

## 6.5.3 DatePicker

Use this control for display and entry of a date and time. It allows users to choose a date from a pop-up calendar or type a date in a text field. At runtime, the form automatically validates the date using the format for the user's locale and timezone. If the user enters an incorrect format, the form displays an error message. The DatePicker control's tooltip displays the valid date format. The default DatePicker control looks like this:

***Figure 6-6*** *Sample DatePicker Control*

When the Show date picker property is true, the form displays the date field along with a button. When the user clicks the button, the form launches a calendar for the user to select the date. The calendar pop-up is shown here:

***Figure 6-7*** *Sample Calendar Control*

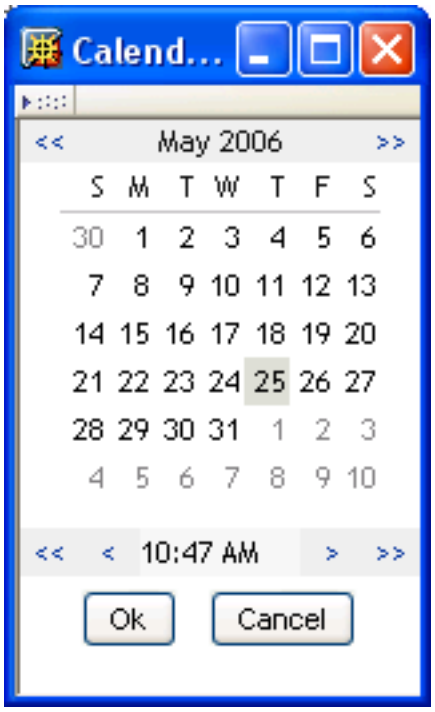



***Table 6-6*** *DatePicker Control Properties*

| Property name | Description |
|---|---|
| Datetime indicator | When false, the Calendar pop-up does not display the time. |
| Day headers | A comma-separated, single-quoted list of values displayed by the Calendar pop-up to indicate the day of the week. This value is localizable. |
| Month names | A comma-separated, single-quoted list of values displayed by the Calendar pop-up to indicate the month name. This value is localizable. |
| Show date picker | When true displays the calendar pop-up. If false, the calendar pop-up does not display. The user must type the date in the text field using the proper format for their locale. |

## 6.5.4 DNDisplay

Use this control to display a read-only DN. You populate the control from flowdata. The control can display the full DN or a set of attributes associated with the DN depending on the properties you set.

***Figure 6-8*** *Sample DNDisplay*

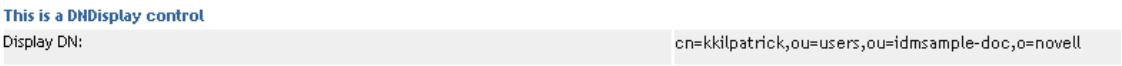


***Figure 6-9*** *Sample DNDisplay with Display Expression Specified*

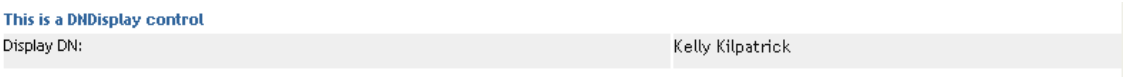


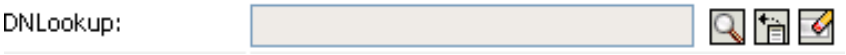

***Table 6-7*** *DNDisplay control properties*

| Property name | Description |
|---|---|
| Display expression | Leave this value blank if you want to display the full DN or CN value.<br><br>If you want to mask the DN by displaying attributes instead, launch the expression builder and select the desired attributes from the list. (You must first specify an *Entity key for DN expression lookup*.)<br><br>For example, to show the user entity's first and last name attributes, construct an expression like this: `FirstName LastName`.<br><br>Make sure the attribute's View, Read, Search, and Required properties are set to true in the directory abstraction layer. See Section 3.7.2, "Attribute Properties," on page 39. |
| Entity key for DN expression lookup | Leave this value blank if you want to display the full DN or CN value retrieved from the Identity Vault.<br><br>If you want to mask the DN or CN by displaying attributes instead, choose the entity from the drop-down list and specify a set of attributes in the *Display expression* property.<br><br>The entity you choose must:<br><br>• Have the directory abstraction layer View property set to true.<br>• Be the entity of the DN you are working with.<br><br>See for more information, see Section 6.6, "Working with Distinguished Names," on page 104. |

## 6.5.5 DNLookup

Use this control to allow users to search and retrieve DNs from the Identity Vault. You can initialize the control with a DN from the flowdata. You set properties to control the entities and containers that the user can search and the format of the DN.

***Figure 6-10*** *Sample DNLookup Control*




The buttons associated with the DNLookup control are described in the following table.

***Table 6-8*** *DNLookup Control Buttons*

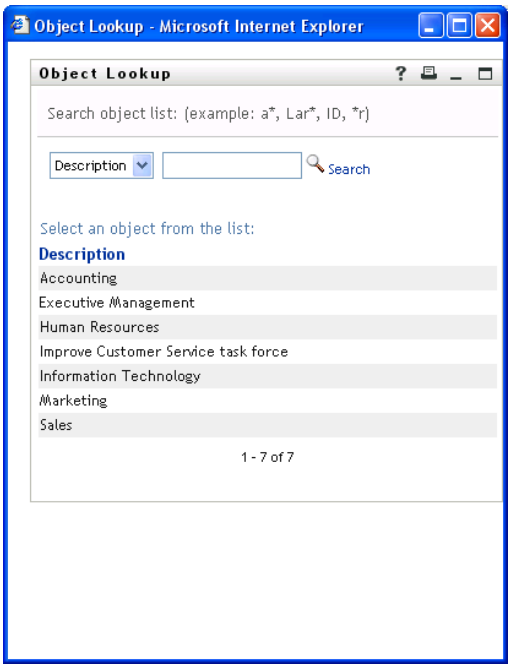
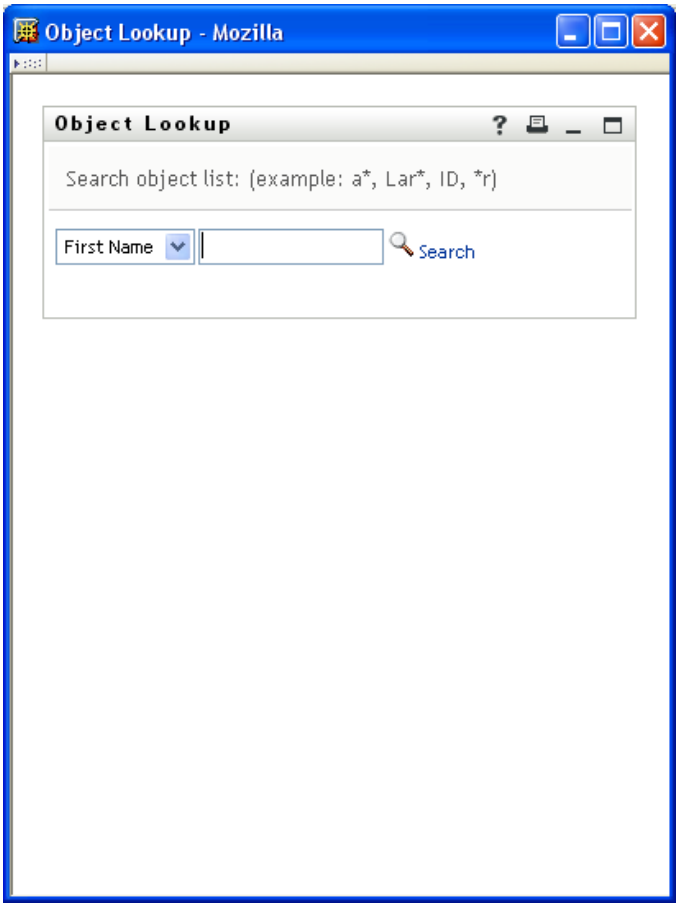
| Button | Description |
|---|---|
|  | Launches a search dialog box called an Object Selector. You can define whether the Object Selector displays containers or objects via the *Object Selector type* property. This is an example of an object lookup.<br>  <br><br>The attribute shown in the Object Selector's drop-down list (*Description* in the above example) is specified in the directory abstraction layer. For more information, see the *Entity Attribute Key for object lookup* property. The availability of this button is controlled by the Show object selector property. |
|  | Show history. Allows users to view the history of objects that they've searched. They can select from this list or clear its contents. The availability of this button is controlled by the Show object history button property. |
|  | Reset field. Deletes the field contents. The availability of this button is controlled by the Show clear history button property. |

***Table 6-9*** *DNLookup control properties*

| Property Name | Description |
|---|---|
| Display expression | This property only applies when you initialize the control from flowdata. Leave this value blank if you want to display the full DN or CN value.<br><br>If you want to mask the DN by displaying attributes instead, launch the expression builder and select the desired attributes from the list. (You must first specify an *Entity key for DN expression lookup*.)<br><br>For example, to show the user entity's first and last name attributes, construct an expression like this: `FirstName LastName`.<br><br>Make sure the attribute's View, Read, Search, and Required properties are set to true in the directory abstraction layer. See Section 3.7.2, "Attribute Properties," on page 39. |
| Entity attribute key used for object lookup | Choose an attribute of the selected *Entity key used for object lookup* property. The attributes in the drop-down are the attributes whose directory abstraction layer *Control type* property is set to DNLookup. The DNLookup control type property controls the attributes displayed in the Object Selector dialog at runtime.<br><br>If you leave this blank, the Object Selector displays all of the entity's attributes that have the directory abstraction layer search and required properties set to true. |
| Entity key for DN expression lookup | This property only applies when you initialize the control from flowdata. Leave this value blank if you want to display the full DN or CN value retrieved from the Identity Vault.<br><br>If you want to mask the DN or CN by displaying attributes instead, choose the entity from the drop-down list then specify a set of attributes in the *Display expression* property.<br><br>The entity you choose must:<br><br>• Have the directory abstraction layer View property set to true.<br>• Be the entity of the DN you are working with.<br><br>For more information, Section 6.6, "Working with Distinguished Names," on page 104. |
| Entity key used for object lookup | A required field. Specify an entity to display in the Object Selector dialog. If you do not specify a value in the *Entity attribute key used for object lookup*, then you create a "general Object Selector". You can find out more about general object selectors in Section 6.6, "Working with Distinguished Names," on page 104.<br><br>This property is related to *Entity attribute key used for object lookup* and the *Object selector type* properties. |

| Property Name | Description |
| --- | --- |
| Object Selector type | Determines whether the Object Selector dialog box performs an Object Lookup or a Container Lookup. This is an example of an Object Lookup:<br>  <br><br>*paramlist*: Causes the Object Selector dialog to perform an object lookup. You specify the lookup criteria via the *Entity key used for object lookup* and the *Entity attribute key used for object lookup* properties.<br>*container*: Causes the Object Selector dialog to display one or more containers for selection. The containers for searching are determined by the *Search container* property specified in the directory abstraction layer for the entity named in the *Entity key used for object lookup* property. For example, if the *Entity key used for object lookup* property is Group, the search container is set to %group-root% by default. If no search container is used, the search root specified during the user application install is used. |
| Show clear button | If true, the form displays the *Reset field* button. |
| Show object history button | If true, the form displays the *Show history* button. |
| Show object selector button | If true, the form displays the *Object Selector* button. |

## 6.5.6 DNMaker

Use this control to allow users to construct a DN value by specifying a naming value and choosing a container.

***Figure 6-11*** *Sample DNMaker Control*


***Table 6-10*** *DNLookup Control Buttons*

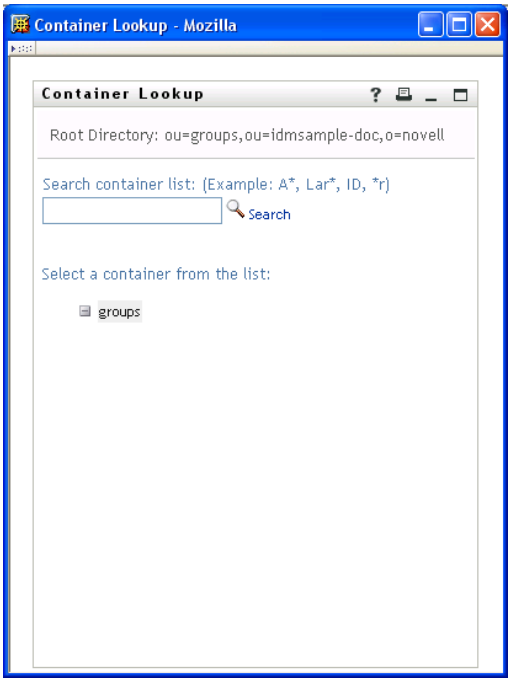
| Button | Description |
|---|---|
|  | Launches an Object Selector for container searches like the one shown below.<br>  <br>The container search root is defined for the entity specified in the *Entity used for object lookup property*. The availability of this button is controlled by the Show object selector property. |
|  | Show history. Allows users to view the history of objects that they've searched. They can select from this list or clear its contents. The availability of this button is controlled by the Show object history button property. |
|  | Reset field. Deletes the field contents. The availability of this button is controlled by the Show clear history button property. |

**Table 6-11** *DNMaker Control Properties*

| Property | Description |
|---|---|
| Entity key used for object lookup | A required field. Choose an entity from the drop-down. This will determine the root for the container search that is launched when the user clicks the Object Selector button. The root container for the entity is defined in the directory abstraction layer property called Search Container. If a search container is not specified for this entity, then the Root Container DN specified during the user application installation is used instead. |
| Naming attribute | The naming attribute used to construct the final DN. This value display next to the control's display label as an extra hint to the user. |
| Show clear button | If true, the form displays the Reset field button. |
| Show object history button | If true, the form displays the Show history button. |
| Show object selector button | If true, the form displays the Object Selector button. |

## 6.5.7 Global List

Use this control to allow users to select a single entry from a drop-down list. The contents of the list are defined in a directory abstraction layer global list element.

**Figure 6-12** *Sample Global List Control*


**Table 6-12** *Global List Properties*

| Property Name | Description |
|---|---|
| DAL global list key | Specifies the unique identifier of the global list. This must correspond to the key specified in the directory abstraction layer. |

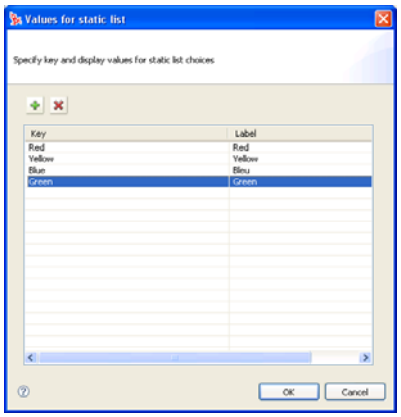
For more information about global lists, see Section 3.3, "Working with Lists," on page 29.

## 6.5.8 MVCheckbox

Use this control to display a set of labelled check boxes. You specify the label and its associated values through the List item property. A sample MVCheckbox control is shown below.

***Figure 6-13*** *Sample MVCheckbox Control*


***Table 6-13*** *MVCheckbox Control Properties*

| Property Name | Description |
|---|---|
| List item | Allows you to define a set of static values that comprise the check box labels and values. Click the List property button to launch the list value dialog box shown here:<br>  |

**TIP:** To retrieve user-entered values for this control, use flowdata.getObject() and not flowdata.get(). If you use flowdata.get(), you get only the first value.

For more information on preselecting values, see the Section 9.2.3, “Form Control Examples,” on page 148.

## 6.5.9 MVEditor

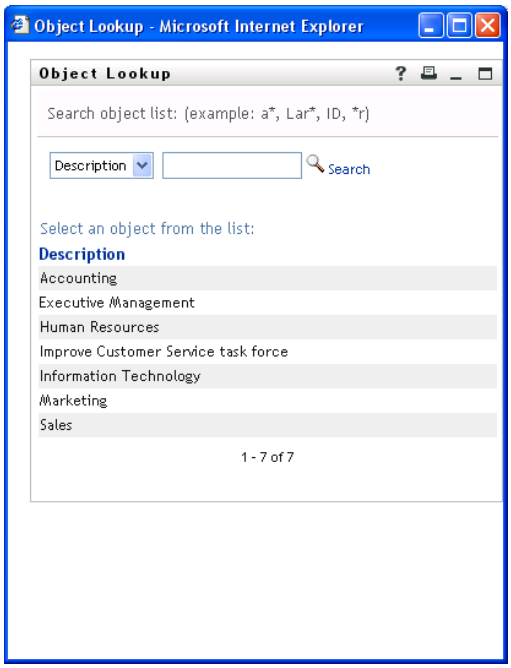
Use this control to allow users to display, edit, or add multiple values in a drop-down list box. You can load the data dynamically from the Identity Vault, or allow users to enter the values.

The control’s appearance varies depending on the data type of the control and the properties that you specify. For example, if the data type is a DN, you can set properties that displaying specific attributes related to the DN. You can also enable an Object Selector button that allows users to

search and select values by setting the *Entity key used for object lookup* and the *Entity Attribute key used for object lookup* properties.

***Figure 6-14*** *Sample MVEditor with Object Lookup Properties Set*


***Table 6-14*** *MVEditor with Object Selector Properties Set Control Buttons*

| Button | Description |
|---|---|
|  | Launches a search dialog box called an Object Selector. The Object Selector dialog box looks like this:<br>  <br>The user can select a value from the list to populate the control. The attribute displayed in the drop-down list (*Description* in the above example) is specified in the directory abstraction layer. You specify it in the attribute's UIControl property. See "Attribute UI Control Properties" on page 41. The availability of this button is controlled by the Show object selector property. |
|  | Show history. Allows users to view the history of objects that they've searched. They can select from this list or clear its contents. The availability of this button is controlled by the Show object history button property. |

| Button | Description |
|---|---|
|  | Reset field. Deletes the field contents. The availability of this button is controlled by the Show clear history button property. |

If you do not set the object lookup properties, the MVEditor displays a simple edit control.

***Figure 6-15*** *Sample MVEditor without Object Lookup Properties Set*


The buttons associated with the simple edit control are:

***Table 6-15*** *MVEditor Control Buttons*

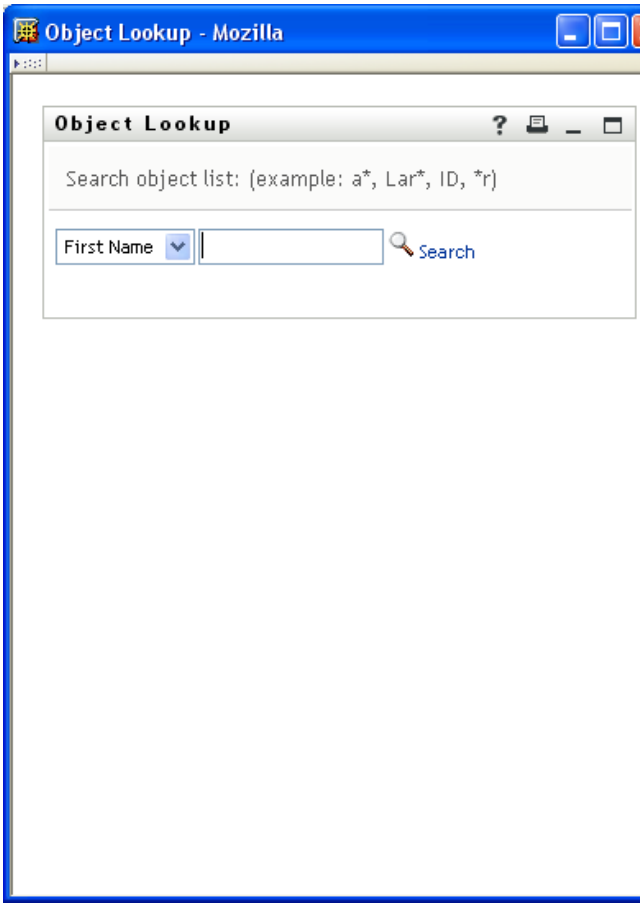
| Button | Description |
|---|---|
|  | Adds an item to the end of the list. |
|  | Deletes the selected list item. |
|  | Edits the selected list item. |

**TIP:** When the MVEditor control's Editable property is false, this control is read-only and the form does not display any MVEditor control buttons.

***Table 6-16*** *MVEditor Control Properties*

| Property Name | Description |
|---|---|
| Display expression | Leave this value blank if you want to display the full DN or CN value.<br><br>If you want to mask the DN or CN by displaying attributes instead, launch the expression builder and select the desired attributes from the list. (You must first specify an *Entity key for DN expression lookup*.)<br><br>For example, to show the user entity's first and last name attributes, construct an expression like this: `FirstName LastName`.<br><br>Make sure the attribute's View, Read, Search, and Required properties are set to true in the directory abstraction layer. See Section 3.7.2, "Attribute Properties," on page 39. |
| Enforce uniqueness | Forces user-entered list items to be unique. |

| Property Name | Description |
|---|---|
| Entity attribute key used for object | Choose an attribute of the selected *Entity key used for object lookup* property. The attributes in the drop-down are the attributes whose directory abstraction layer Control type property is set to DNLookup. The DNLookup control type definition controls the attributes displayed in the Object Selector dialog at runtime.<br><br>If you leave this blank, the Object Selector displays all of the entity's attributes that have the directory abstraction layer search and required properties set to true. |
| Entity key for DN expression lookup | Leave this value blank if you want to display the full DN or CN value retrieved from the Identity Vault.<br><br>If you want to mask the DN or CN by displaying attributes instead, choose the entity from the drop-down list and specify a set of attributes in the *Display expression* property.<br><br>The entity you choose must:<br><br>• Have the directory abstraction layer View property set to true.<br>• Be the entity whose DN you are retrieving from the Identity Vault.<br><br>See Section 6.6, "Working with Distinguished Names," on page 104 for more information. |
| Entity key used for object lookup | A required field. Specify an entity to display in the Object Selector dialog. If you do not specify a value in the *Entity attribute key used for object lookup*, then you create a general Object Selector. You can find out more about general Object Selectors in Section 6.6, "Working with Distinguished Names," on page 104. |
| Ignore case | If true, ignore case when enforcing uniqueness. |
| Lower bound (for numbers only) | Minimum integer or decimal value. |
| Maximum length | Maximum number of characters for string values. |
| Minimum length | Minimum number of characters for string values. |
| Number of lines displayed | The number of rows displayed. |
| Numbers only | If true, only numbers can be entered. |

| Property Name | Description |
| --- | --- |
| Object Selector type | Determines whether the Object Selector dialog box performs an Object Lookup or a Container Lookup. This is an example of an Object Lookup:<br>  <br><br>*paramlist*: Causes the Object Selector dialog to perform an object lookup. You specify the lookup criteria via the *Entity key used for object lookup* and the *Entity attribute key used for object lookup* properties.<br>*container*: Causes the Object Selector to display one or more containers for selection. The containers for searching are determined by the *Search container* property specified in the directory abstraction layer for the entity named in the *Entity key used for object lookup* property. For example, if the *Entity key used for object lookup* property is Group, the search container is set to %group-root% by default. If no search container is used, the search root specified during the user application install is used. |
| Show object history button | When true, displays the *Object History* button next to the control. |
| Show object selector button | When true displays the *Object Selector* button next to the control. |

| Property Name | Description |
|---|---|
| Upper bound (for Numbers only) | The maximum numeric value users can enter. |

**TIP:** To retrieve user-entered values for this control, use flowdata.getObject() and not flowdata.get(). If you use flowdata.get(), you get only the first value.

For more information about preselecting items, see Chapter 9, "Working with ECMA Expressions," on page 139.

## 6.5.10 PickList

Use the PickList control to allow users to view and choose one or more values from a dynamically-generated list of choices. The list items are DN or CN values retrieved from the Identity Vault. You can display the full DN or CN or use the PickList properties to specify the attributes to display instead.

***Figure 6-16*** *Sample PickList Control without DN Masking*


***Figure 6-17*** *Sample PickList Control with DN Masking*

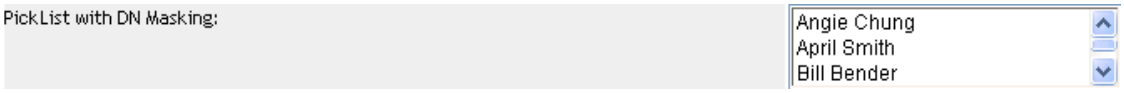



***Table 6-17*** *PickList control properties*

| Property Name | Description |
|---|---|
| Allow multiple selections | When true, the user can select more than one list value using their platform-specific multi-select keys. |
| Display expression | Leave this value blank if you want to display the full DN or CN value.<br><br>If you want to format the DN or CN by displaying attributes instead, launch the expression builder and select the desired attributes from the list. (You must first specify an *Entity key for DN expression lookup*.)<br><br>For example, to show the user entity's first and last name attributes, construct an expression like this: `FirstName LastName`.<br><br>Make sure the attribute's View, Read, Search, and Required properties are set to true in the directory abstraction layer. See Section 3.7.2, "Attribute Properties," on page 39. |

| Property Name | Description |
|---|---|
| Entity key for DN expression lookup | Leave this value blank if you want to display the full DN or CN value retrieved from the Identity Vault.<br><br>If you want to mask the DN or CN by displaying attributes instead, choose the entity from the drop-down list and specify a set of attributes in the *Display expression* property.<br><br>The entity you choose must:<br><br>• Have the directory abstraction layer View property set to true.<br>• Be the entity whose DN you are retrieving from the Identity Vault. |
| Number of lines displayed | The number of lines displayed by the control. This is not the number of records retrieved or displayed, but the vertical size of the control. If you set this number to 10 and there are only 5 records to display, the control size is still 10 lines. |

**TIP:** To retrieve user-entered values for this control, use flowdata.getObject() and not flowdata.get(). If you use flowdata.get(), you get only the first value.

For more information on displaying the control with a preselected option, see Section 9.2.3, "Form Control Examples," on page 148.

## 6.5.11 Static List

Use this control to display a list of items in a drop-down list from which users can select a single item. The list items are static and are stored with the provisioning request definition. The text "Click here to select" only appears if the field is not set to Required.

***Figure 6-18*** *Sample Static List Control*

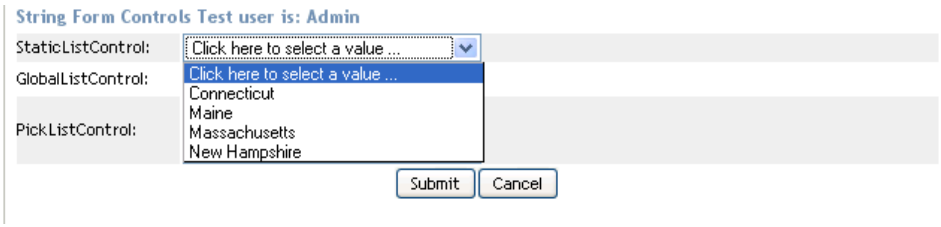
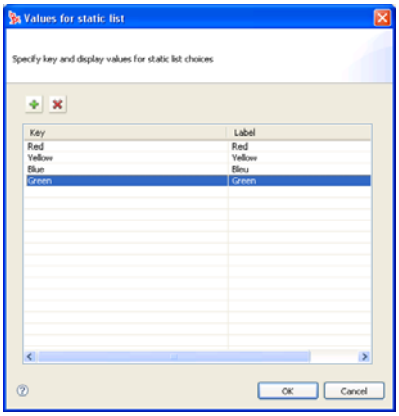


**Table 6-18** *Static List Properties*

| Property Name | Description |
|---|---|
| List item | Allows you to define a set of labels and values that comprise the static list. Click the *List property* button to launch the list value dialog box shown here:<br>  <br><br>Click *Add* to add list items. Each list item must have a unique key. The dialog automatically generates a unique key when you insert a new list item. You can click on the key name and change it. A blank key (null) is valid, so it is possible to have a list item with a blank key and a blank label. The displayed label is the one defined for the default language. |

## 6.5.12 Text

Use the Text control for data display or user input. User input is validated depending on the control's data type.

**Figure 6-19** *Sample Text Control*


**Table 6-19** *Text Control Properties*

| Property Name | Description |
|---|---|
| Lower bounds (for numbers only) | The lowest number allowed for decimal or integer values. |
| Maximum length | The maximum length for string values. |
| Minimum length | The minimum length for string values. |
| Upper bound (for numbers only) | The highest number allowed for decimal or integer values. |
| Validation Mask (regular expression) | An expression used for validating the field's data. Valid masks include:<br>• [a-zA-Z]*$ |

## 6.5.13 Text Area

Use this control to display or accept input of multi-line data. Users can select multiple lines of data using multi-select key combination for their platform.

***Figure 6-20*** *Sample Text Area Control*


***Table 6-20*** *Text area control properties*

| Property Name | Description |
|---|---|
| Number of columns displayed | The width of the control; the number of characters wide. |
| Number of lines displayed | The number of lines to display at one time. |

## 6.5.14 Title

Use this read-only control to label your form or provide instructions.

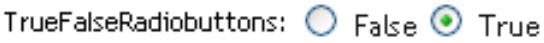
***Table 6-21*** *Title Control Properties*

| Property Name | Description |
|---|---|
| Font-size | Specify one of these: small, medium, and large. |
| Style class | Choose font style (such as bold) and colors from a palette. |

## 6.5.15 TrueFalseRadioButtons

Use this control to display a choice of True or False as a set of radio buttons.

***Figure 6-21*** *Sample TrueFalseRadioButtons Control*

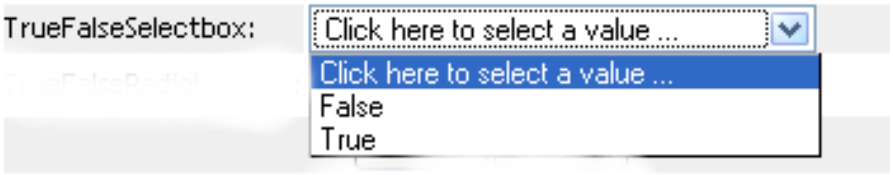



This control has no custom properties.

### 6.5.16 TrueFalseSelectBox

Use this control to display a choice of True or False in a drop-down. The text “Click here to select a value” displays only when the field is not required

***Figure 6-22*** *Sample TrueFalseSelectBox Control*




This control has no custom properties

## 6.6 Working with Distinguished Names

The following controls provide specialized support for DNs:

- DNDisplay
- DNLookup
- DNMaker
- MVEditor
- PickList

This section describes the specialized support including:

- Section 6.6.1, “Formatting a DN,” on page 104.
- Section 6.6.2, “Controlling the Object Selector,” on page 104.

### 6.6.1 Formatting a DN

If you have a DN value, you can display either the DN or a set of attributes related to that DN. For example, if the control displays the DN of a user entity, you could display the user entity's First Name and Last Name attributes instead. The control's that support this feature are: DNDisplay, DNLookup, MVEditor and Picklist.

You define the attributes to display in the control’s Display Expression property. This display expression resolves at runtime by replacing the attribute keys with the attribute values.

### 6.6.2 Controlling the Object Selector

The Object Selector dialog box provides a convenient way to allow users to search and select objects or containers. You control the Object Selector’s contents through the properties described in the following table.

***Table 6-22*** *Properties for Defining the Object Selector Dialog Box*

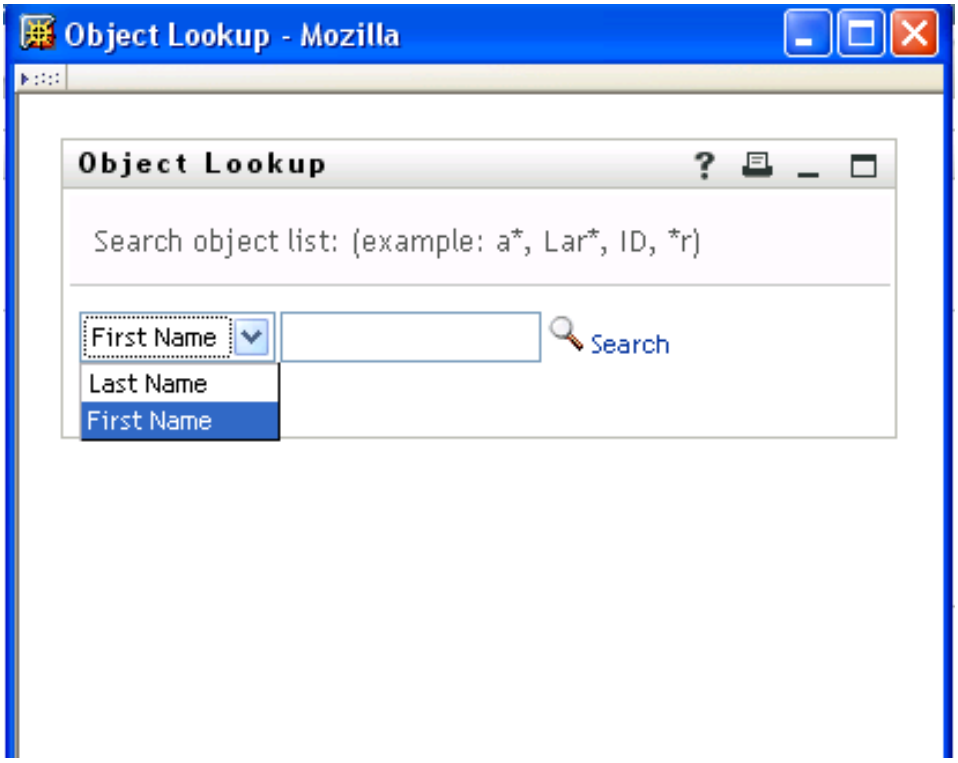
| Property | Description |
|---|---|
| Entity key used for object lookup | This is the key to the directory abstraction layer entity whose DN you want to search for or display. This is a required field. |
| Entity attribute key used for object lookup | This is the key to the attribute that you want to use in the Object Selector.<br><br>When you leave this field blank, you create a general Object Selector. This means that the Object Selector displays all of the attributes of the entity (defined by the *Entity key used for object lookup*) whose Search and Required properties are set to true.<br><br>When you specify an attribute, the contents of the Object Selector are defined by the attribute's DNLookup Control type property (in the directory abstraction layer). |
| Object selector type | paramlist: Causes the Object Selector dialog to perform an object lookup. You specify the lookup criteria via the Entity key used for object lookup and the Entity attribute key used for object lookup properties.<br><br>container: Causes the Object Selector dialog to display one or more containers for selection. The containers for searching are determined by the Search container property specified in the directory abstraction layer for the entity named in the Entity key used for object lookup property. For example, if the Entity key used for object lookup property is Group, the search container is set to %group-root% by default. If no search container is used, the search root specified during the user application install is used. |
| Show object selector button | If true, the Object Selector button shows up on the control. Otherwise, it does not. |

### DNLookup Control type definitions and Object Selector Contents

When you specify an Entity key used for object lookup and an Entity attribute key used for object lookup, the Object Selector's are defined the attribute's DNLookup control type definition (in the directory abstraction layer). For example, if you specified the User entity as the object lookup and

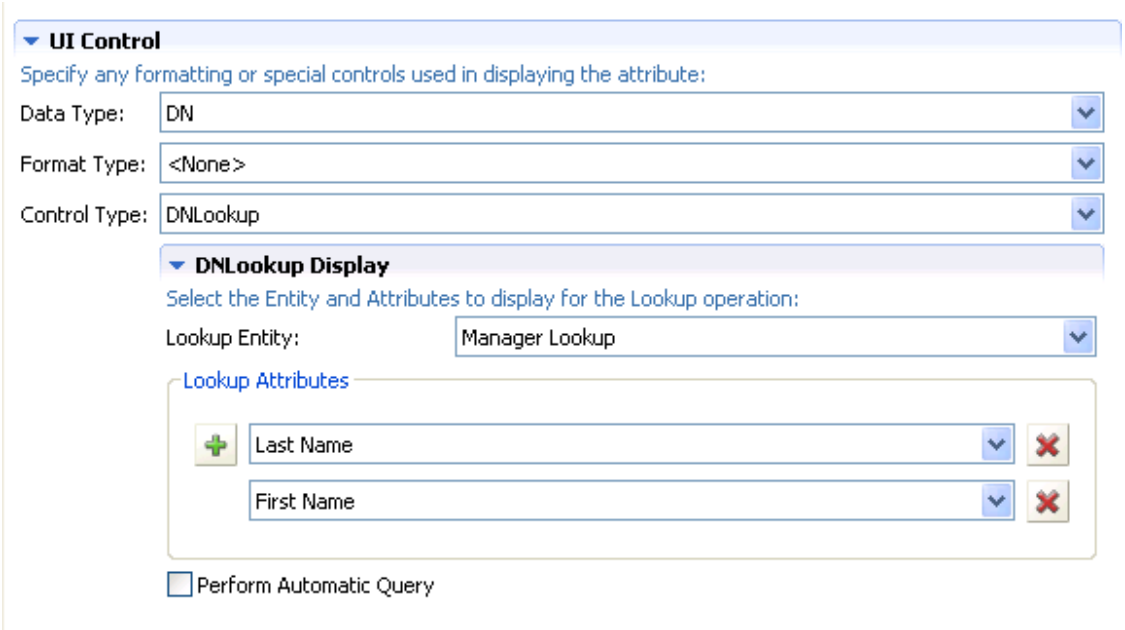
the manager as the attribute. The Object Selector would allow the user to search on the First Name and Last Name attributes.

***Figure 6-23*** *Sample Object Selector*




The Object Selector uses the manager's DNLookup control type definition, to determine the lookup criteria.The DNLookup definition for the manager entity is shown below.

***Figure 6-24*** *Manager Attribute on User DNLookup Property Definition*



You can change the attributes that are used by the Object Selector by changing the Lookup attributes. To allow other attributes in the Object Selector:

1. Determine if the desired attribute is defined for the entity specified as the Lookup Entity. (In this example it is Manager Lookup.)
2. If the attribute you want is available on the lookup entity, you can just add it to the Lookup Attributes. Make sure that it has the Search and Read properties set to true or they do not show in the Object Selector dialog box.
3. If the attribute does not already exist for the Lookup Entity, you must:
   - Add the entity to the Lookup Entity. In the above example, you would add it to the Manager Lookup entity. For more information, see Section 3.2.2, “Adding Attributes,” on page 27.
   - Add the attribute to the DNLookup definition.
   - Deploy the changed definitions. In this example, you’d redeploy the Manager-Lookup entity (if you added a new attribute to its definition) and the User entity because you changed the definition of the manager attribute).
   - Refresh the application server’s DirectoryAbstractionLayerDefinitions cache.

# Creating the Workflow for a Provisioning Request Definition

# 7

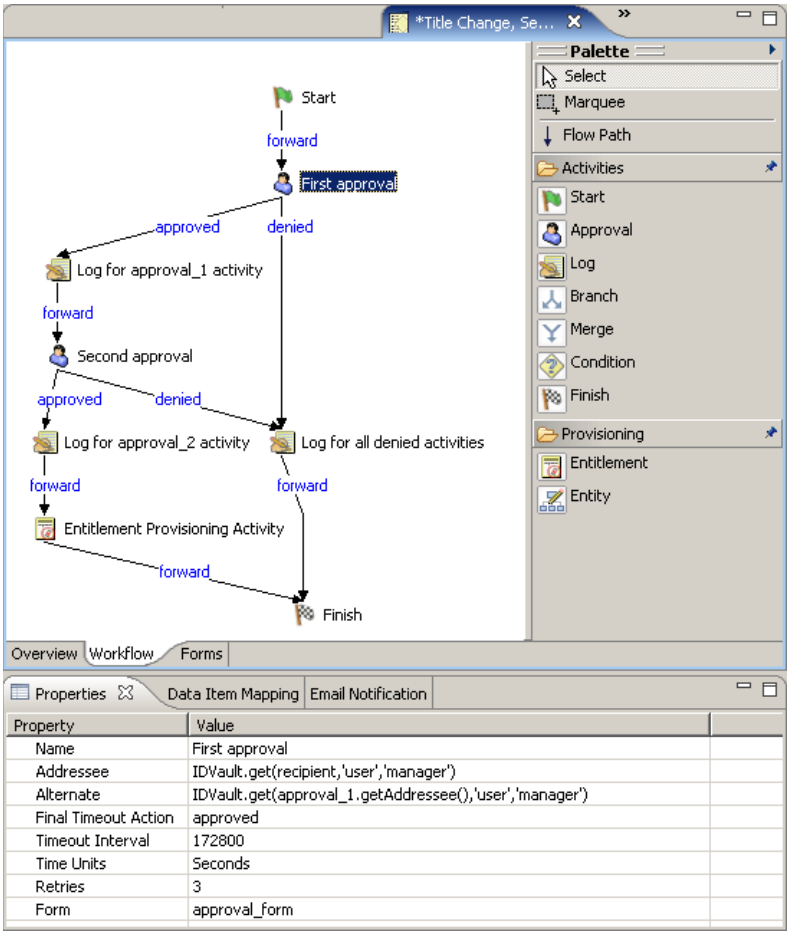
This section provides details on creating the workflow for a provisioning request definition. Topics include:



## 7.1 About the Workflow Tab

You use the *Workflow* tab to display the Workflow page. You use the Workflow page to define the behavior of the workflow for the provisioning request definition. The Workflow page consists of a canvas, a palette, and associated views.

***Figure 7-1*** *Workflow Page*



### 7.1.1 Canvas

The canvas provides a graphical view of the activities in the workflow. When you create a new provisioning request definition that is not based on a template, the canvas is blank except for a Start and Finish activity.

If you right-click anywhere on the canvas, a menu is displayed. The menu includes the following commands:

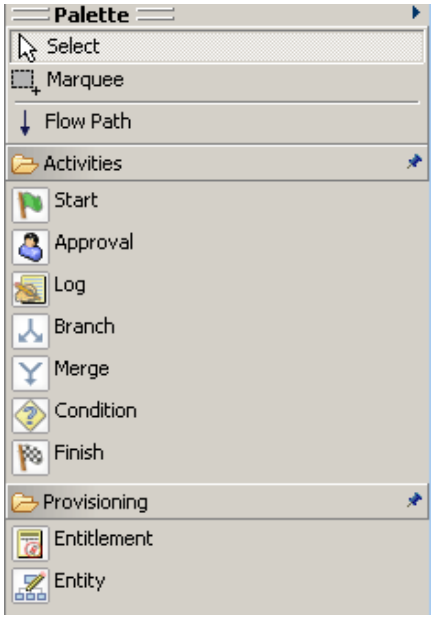
***Table 7-1*** *Workflow Menu*

| Item | Description |
|---|---|
| Delete | Deletes the selected activity or flow path. |
| Show Activity IDs | Switches the workflow editor between displaying activity names and activity IDs. Activity IDs are system defined and are not editable. However, if errors associated with activities are detected during validation, Designer identifies the activity in which the error occurred by activity ID. When this is the case, you need to turn on the display of activity IDs in order to locate the activity on the canvas. You can specify whether activity names or activity IDs are displayed by default by choosing *Window > Preferences > Provisioning > Workflows > Diagram Preferences > Show Activity IDs*. |
| Show Flow Path Types | Turns the display of flow path types (for example, forward, approved, denied) on and off. When Show Flow Path Types is turned on, a label is displayed on each flow path indicating the flow path type. |
| Show Properties | Displays the Properties view for the selected activity. |
| Show Data Item Mapping | Displays the Data Item Mapping view for the selected activity. |
| Show Email Notification | Displays the Email Notification view for the selected activity. |

## 7.1.2 Palette

The palette provides icons for activities that can be dragged onto the canvas to create the workflow, and also provides tools for manipulating the icons and for linking activities:

***Figure 7-2*** *Workflow Palette*




The palette includes the following tools:

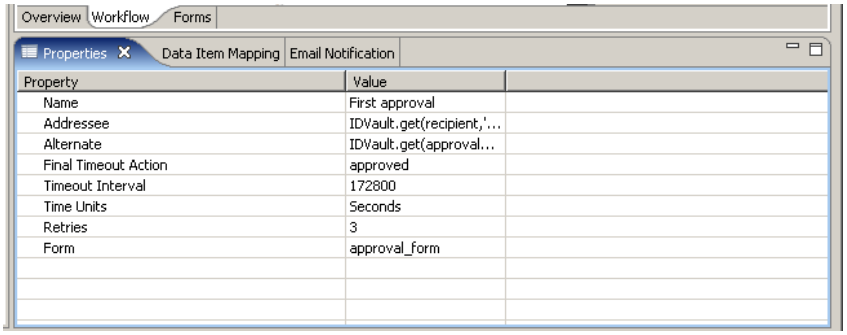
***Table 7-2*** *Workflow Palette*

| Tool | Description |
|---|---|
| Select | Selects individual nodes or flow paths. To select a node, click the Select tool, then click a node. |
| Marquee | Selects multiple nodes or flow paths. Use this tool to move items as a group. To select multiple items, click the Marquee tool, then click in an area outside of the items that you want to select. Hold down the mouse button and drag over the items that you want to select, then release the mouse button.<br><br>When multiple items are selected, only the properties for the first item selected are displayed in the Properties view (see Section 7.1.3, “Views,” on page 112 for information about Views). |
| Flow Path | Creates flow paths between nodes. Flow paths provide connection logic for connecting nodes. For information about connecting nodes, see Section 7.4, “Adding the Flow Paths,” on page 117. |
| Activities (for example, Start, Approval, Log.) | Inserts the selected activity into the workflow. For information about adding activities, see Section 7.2, “Adding Activities to a Workflow,” on page 112. For detailed descriptions of the activities, see Chapter 8, “Configuring the Workflow Activities and Flow Paths,” on page 119. |

### 7.1.3 Views

The Workflow page also includes the Properties, Data Item Mapping, and Email Notification views:

***Figure 7-3*** *Workflow Views*




You can right-click the icon for an activity to select a view from a context menu. Not all activities utilize all views. The following table identifies the views and the activities that use them:

***Table 7-3*** *Views for Activities*

| Activity | Properties | Email Notification | Data Item Mapping |
|---|---|---|---|
| Start | X | | X |
| Approval | X | X | X |
| Log | X | | |
| Branch | X | | |
| Merge | X | | |
| Condition | X | | |
| Finish | X | X | |
| Entitlement | X | | X |
| Entity | X | | X |

## 7.2 Adding Activities to a Workflow

1 Click the Workflow tab. A graphical representation of the workflow for the provisioning request definition is displayed:

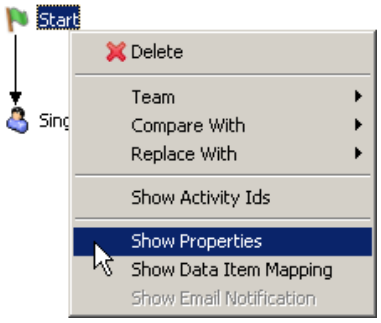

Because every workflow must have a Start and Finish activity, these activities are added to the canvas automatically. The Start Activity is connected to the Finish Activity with a forward link.

2 To add an activity to the workflow, click the icon for the desired activity in the palette and drag the icon onto the workspace.

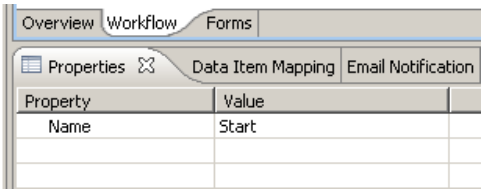
You can insert an activity between activities that are linked by a flow path by dropping the activity onto the flow path. For information about defining flow paths between activities, see Section 7.4, “Adding the Flow Paths,” on page 117. After you have added an activity to the workflow, you should set the properties of the activity (see Section 7.2.1, “Setting the General Properties of an Activity,” on page 113).

## 7.2.1 Setting the General Properties of an Activity

1 Right-click the activity icon for which you want to set properties and select *Show Properties* from the menu.




The Properties view is displayed:



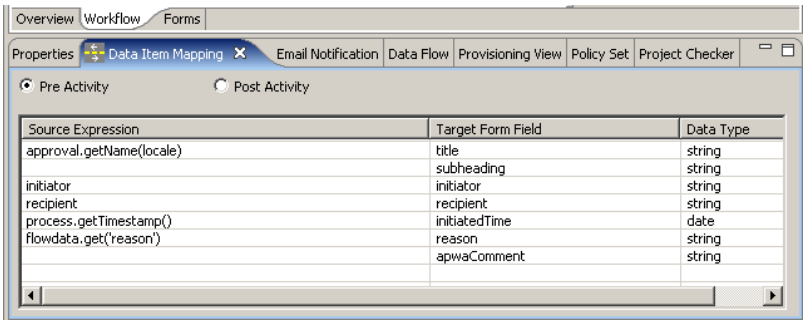

2 Click in the column for a property to set the property. For information about the properties for each activity, see Section 8.1, “Configuring Activities,” on page 119.

## 7.2.2 Defining the Data Item Mappings

You use the Data Item Mapping view to map data from the data flow into fields in a form (pre-activity mapping), and from the form back to the data flow (post-activity mapping).

1 Right-click the activity icon for which you want to set properties and select *Show Data Item Mapping* from the menu.

The Data Item Mapping view is displayed:




2 For pre-activity mapping, click in the *Source Expression* field for the item that you want to map, then specify an expression. For post-activity mapping, click in the *Target Expression* field for the item that you want to map, then specify an expression.

Pre-activity maps can be used for:

- Initializing form control values
- Setting default values for form controls
- Populating complex form controls with data lists derived from LDAP Queries
- Passing data from form controls of a previous activity to a form control in the current activity
- Calling external Java* classes to process data

Post-activity maps can be used for:

- Creating new data items in flowdata
- Moving form control data from an activity into flowdata
- Calling external Java* classes to process data

For information about data item mapping, see Section 8.1, “Configuring Activities,” on page 119.

The Start Activity can have hard coded strings, system variables like process locale and recipient, and Identity Vault expressions (created using the ECMA expression builder VDX Expr Panel) in pre-activity maps.

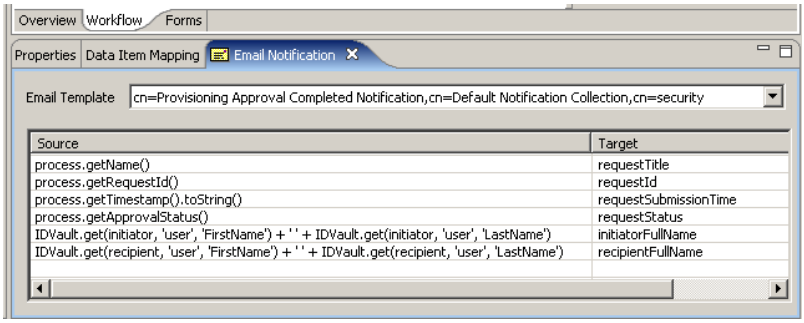
Leave the Source Expression blank in pre-activity maps for form fields that the user is expected to fill in. Alternatively, create a source expression to supply a default value for form fields that the user is expected to fill in. In either case the form field needs to be defined as editable. See Section 6.5.2, “General Properties,” on page 86 for information about setting the properties of form fields.

## 7.2.3 Defining the Email Notification Settings

You use the *Email Notification* view to select an e-mail template, and to specify expressions to provide values for named parameters included in the e-mail template. E-mails are sent when a new Approval activity starts (to notify the approver that they have work to do) and when the Finish activity completes (to notify the initiator that the workflow is done).

1 Right-click the activity icon for which you want to set properties and select Show Email Notification from the menu.

The *Email Notification* view is displayed:




2 Click in the *Email Template* field, and select an e-mail template from the list of defined templates.

3 Click in the *Source* field for a *Target* token and specify an ECMAScript expression that assigns a value to the token.

See Section 8.1, “Configuring Activities,” on page 119 for information about e-mail notification settings).

# 7.3 Working With Entity Activities

You use Entity activities to update entities in the Identity Vault. The procedures for working with Entity activities differ slightly from the procedures for working with other activity types.

The section includes the following topics:



## 7.3.1 Adding or Modifying an Entity

1 From the Workflow page, click the Entity activity icon in the palette, then click on the canvas to insert the Entity activity into the workflow.

2 Click the *Properties* tab.

3 Click in the *Value* column of the *Entity Type* field, and select the *Entity Type* (for example, User, Group) that you want to create or modify. If the target object that you specify in Step 6 already exists, the target object is modified; if the target object doesn't exist, it is created.

4 Click in the *Value* column of the *Operation* field, and select *Create/Modify*.

5 Click the Data Item Mapping tab.

6 Click the button next to the *Entity dn* field to display the ECMA expression builder, then specify an expression that identifies the target of the operation (for example, “recipient”).

7 Click *OK* to return to the Data Item Mapping view.

8 Specify expressions for other attributes as required to create the Entity.

See Section 3.2, “Working with Entities and Attributes,” on page 24 for information about adding entities. If you are adding an entity, you must enter expressions for all required attributes.

### 7.3.2 Using an Entity Activity to Delete an Entity

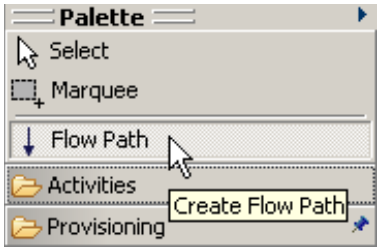
1. From the Workflow page, click the Entity activity icon in the palette, then click the canvas to insert the Entity Activity into the workflow.
2. Click the *Properties* tab.
3. Click in the *Value* column of the *Entity Type* field, and select the Entity Type (for example, User, Group) to which the entity that you want to delete belongs.
4. Click in the *Value* column of the *Operation* field, and select *Delete entity*.
5. Click the Data Item Mapping tab.
6. Click the button next to the *Entity dn* field to display the ECMA expression builder, then specify an expression that identifies the Entity that you want to delete.
7. Click *OK* to return to the Data Item Mapping view.

### 7.3.3 Using an Entity Activity to Delete an Attribute or Value

1. From the Workflow page, click the Entity activity icon in the palette, then click the canvas to insert the Entity Activity into the workflow.
2. Click the *Properties* tab.
3. Click in the *Value* column of the *Entity Type* field, and select the Entity Type (for example, User, Group) of the entity to which the attribute or value that you want to delete belongs.
4. Click in the *Value* column of the *Operation* field, and select *Delete attribute/value*.
5. Click the Data Item Mapping tab.
6. Click on the button next to the *Entity dn* field to display the ECMA expression builder, then specify an expression that identifies the entity that contains the attribute or value that you want to delete.
7. Click OK to return to the Data Item Mapping view.
8. Click in the *Delete Type* field for the attribute to which you want the operation to apply, then select the operation from the list:
   - Select *Delete Attribute* for single value attributes
   - Select either *Delete Attribute* or *Delete Value* for multi-value attributes. Selecting *Delete Value* for multi-value attributes also requires that you to enter an expression to identify the value that you want to delete.
9. To delete a value, click in the *Delete Value Expression* field for the attribute to which you want the operation to apply and specify an expression that resolves to the value of the attribute that you want to delete.

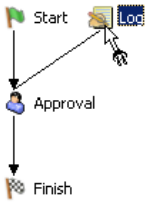
# 7.4 Adding the Flow Paths

1 Click the Flow Path tool in the palette:




The mouse pointer turns into a flow path pointer:

2 Click the activity from which you want the flow path to begin, then click the activity on which you want the flow path to end:




The activities are connected.

3 To configure the flow path, click the Select tool in the palette, right-click the flow path, then select *Show Properties*.

For information about configuring flow paths, see Section 8.2, “Configuring Flow Paths,” on page 135.

# Configuring the Workflow Activities and Flow Paths

# 8

This section provides details on configuring the workflow activities and flow paths. Topics include:

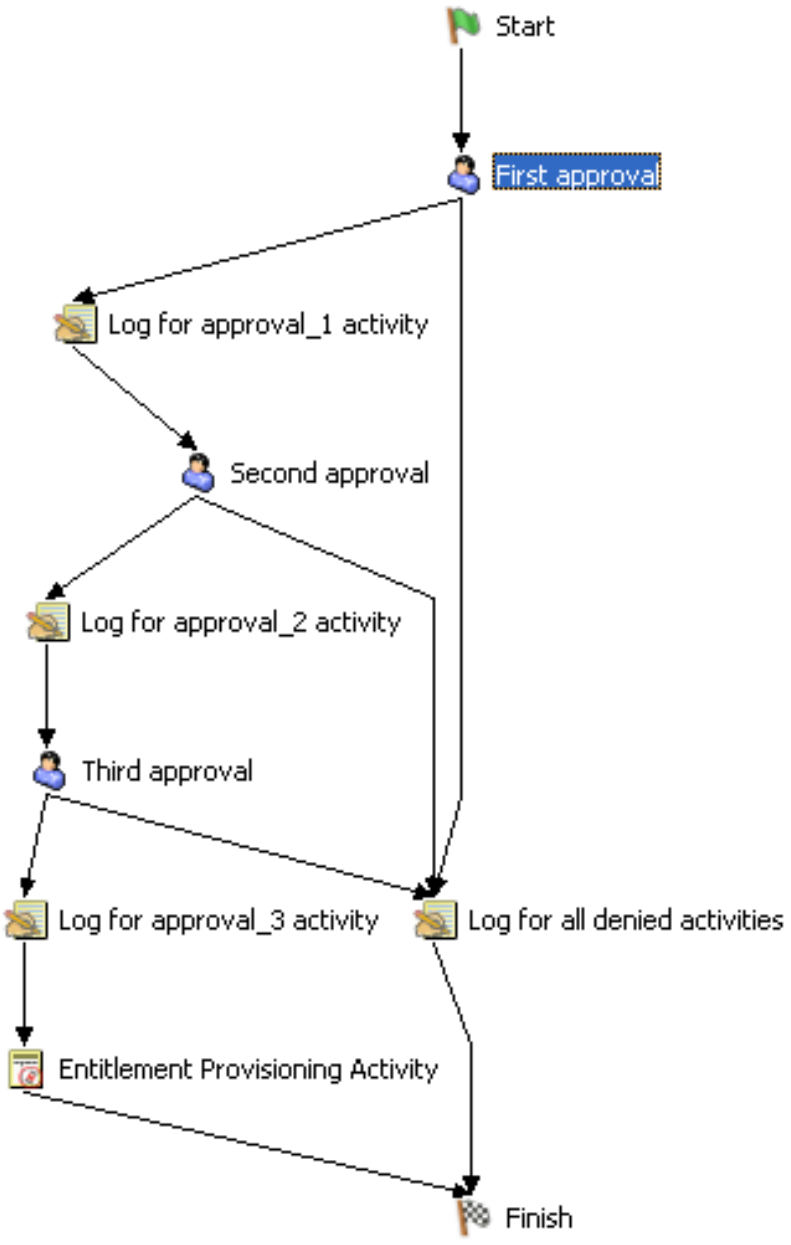


## 8.1 Configuring Activities

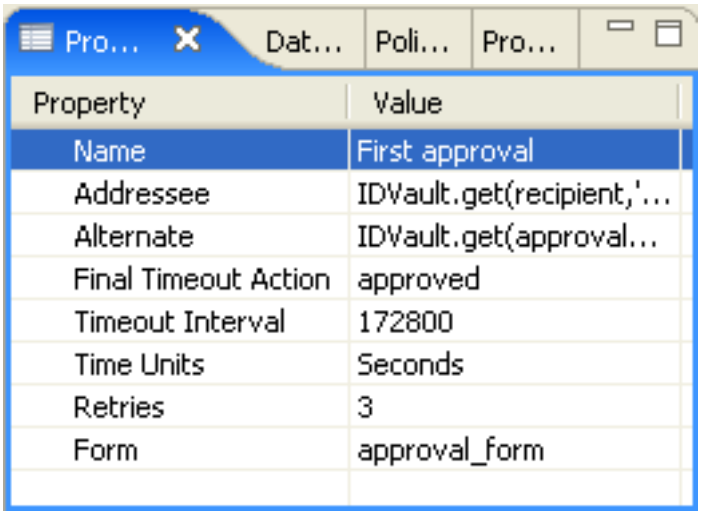
After you’ve added an activity to the workflow diagram, you can specify the properties, data mappings, and e-mail notification settings associated with the activity. For details on adding activities to a workflow, see Section 7.2, “Adding Activities to a Workflow,” on page 112.

To configure a workflow activity:

1 Click the activity in the workflow diagram.



2 Set the property values for the activity on the *Properties* tab.



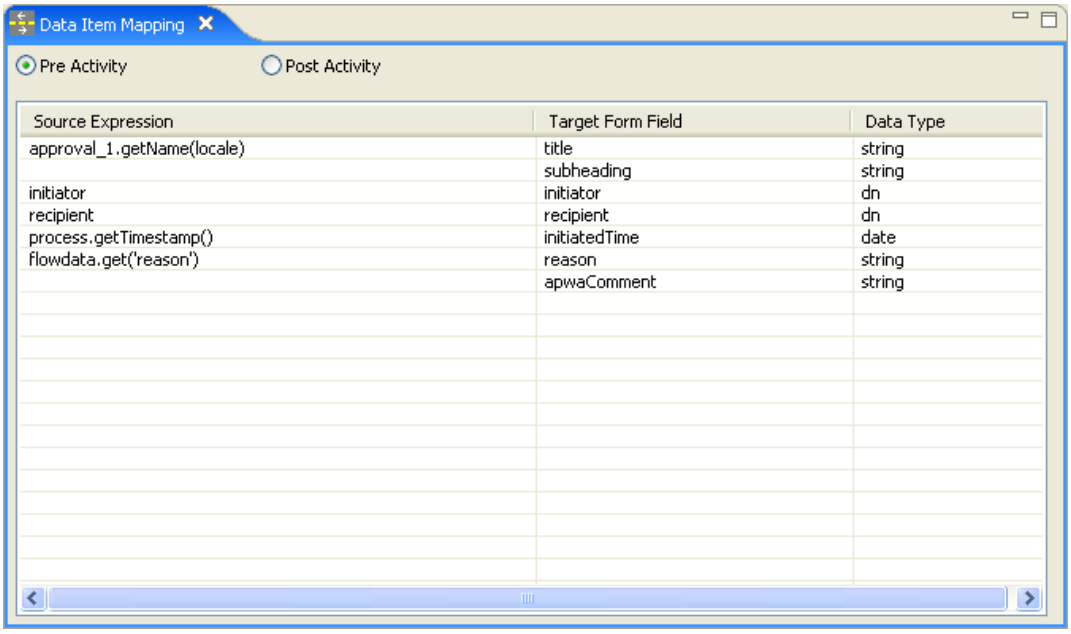

If the *Properties* tab is not displayed, right-click the activity in the workflow diagram and select *Show Properties*.

---

**TIP:** You can also display the *Properties* tab by selecting *Show Properties* from the *PRD* menu.

---

3 Set the data mappings on the *Data Item Mappings* tab.



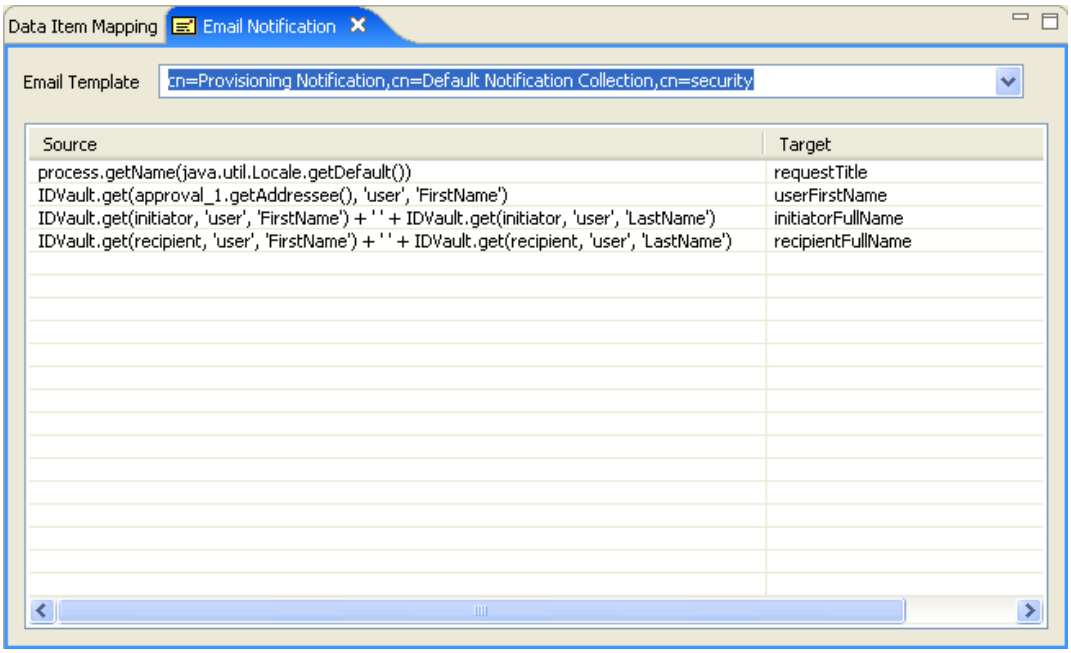

If the *Data Item Mappings* tab is not displayed, right-click the activity in the workflow diagram and select *Show Data Item Mapping*.

---

**TIP:** You can also display the *Data Item Mappings* tab by selecting *Show Data Item Mapping* from the *PRD* menu.

---

4 Specify the e-mail notification settings on the *Email Notification* tab.




If the *Email Notification* tab is not displayed, right-click the activity in the workflow diagram and select *Show Email Notification*.

---

**TIP:** You can also display the *Email Notification* tab by selecting *Show Email Notification* from the *PRD* menu.

---

**Editing an e-mail template** You can edit an e-mail template in Designer. To do this, select an Identity Vault in the *Modeler*, then scroll to *Default Notification Collection* in the *Outline View*. Right-click a template, then select *Edit Template*.

## 8.1.1 Start Activity

The Start activity is the first activity to execute in a workflow. It begins execution when the user makes a request to provision a resource. After the user makes the request, the Start activity displays the initial request form to the user. On the initial request form, the user may be asked to specify a comment that indicates the reason for the request.

You can customize the initial request form to suit your application requirements. For details on customizing forms, see Chapter 6, “Creating Forms for a Provisioning Request Definition,” on page 77.

Before displaying the form to the user, the Start activity performs any pre-activity data mappings specified for the activity.

After the user submits the form, the Start activity performs any post-activity data mappings specified for the activity. These mappings typically include copying data from form fields into the flowdata object.

### Properties

The Start activity has the following property:

***Table 8-1*** *Start Activity Property*

| Property Name | Description |
|---|---|
| Name | Provides a name for the activity. |

### Data Item Mapping

To bind the data items associated with the Start activity, you define pre-activity and post-activity mappings. The pre-activity mappings initialize data in the request form with constants or values retrieved from the flowdata object. The post-activity mappings move form data back into the flowdata object.

***Table 8-2*** *Start Activity Data Item Mappings*

| Setting | Description |
|---|---|
| Pre-Activity | Allows you to specify one or more pre-activity mappings. When this option is selected, you can double-click a cell in the *Source Expression* column to specify where the initial request form gets data for a particular target form field.<br><br>**NOTE:** When the *Pre-Activity* option is selected, the cells in the *Target Form Field* column are not editable. |
| Post-Activity | Allows you to specify one or more post-activity mappings. When this radio button is selected, you can double-click on a cell in the Target Expression column to specify where data from a form field should be copied after the form has been processed.<br><br>**NOTE:** When the *Post-Activity* option is selected, the cells in the *Source Form Field* column are not editable. |
| Source Expression | Specifies a source expression for a pre-activity mapping. When you click a cell in the *Source Expression* column, the ECMA expression builder displays to help you define your expression. |
| Target Expression | Specifies a target expression for a post-activity mapping. When you click a cell in the *Target Expression* column, the ECMA expression builder displays to help you define your expression. |

For details on building ECMA expressions, see Chapter 9, “Working with ECMA Expressions,” on page 139.

### Email Notification

Not supported with this activity.

## 8.1.2 Approval Activity

The Approval activity is a user-facing activity that displays an approval form to the user. On the approval form, the user can approve, deny, or refuse a provisioning request. The Approval activity can have multiple outgoing flow paths, but only one of the paths is executed at runtime.

You can customize the approval form to suit your application requirements. For details on customizing forms, see Chapter 6, “Creating Forms for a Provisioning Request Definition,” on page 77.

Before displaying the form to the user, the Approval activity performs any pre-activity data mappings specified for the activity.

After the user submits the form, the Approval activity performs any post-activity mappings specified for the activity. These mappings typically include copying data from form fields into the flowdata object.

### Properties

The Approval activity has the following properties:

***Table 8-3*** *Approval Activity Properties*

| Property Name | Description |
|---|---|
| Name | Provides a name for the activity. |
| Addressee | Specifies a dynamic expression that identifies the addressee for the activity. The addressee is determined at runtime, based on how the expression is evaluated.<br><br>**TIP:** To simplify the process of testing a new workflow, you can set the addressee to be the recipient. This removes the need to log out of the user application and log in again as a manager each time you want to test your forms. This technique is particularly useful when the workflow involves multiple levels of approval. Once the testing phase is complete, you can change the addressee to the correct value.<br><br>For details on building ECMA expressions, see Chapter 9, “Working with ECMA Expressions,” on page 139. For descriptions of the system variables available in a workflow, see Section 4.3.3, “Understanding Workflow Data,” on page 60. |

| Property Name | Description |
|---|---|
| Alternate | Specifies a dynamic expression that identifies the user who should get this task if the timeout limit has been reached.<br><br>The alternate addressee is determined at runtime, based on how the expression is evaulated.<br><br>For details on building ECMA expressions, see Chapter 9, “Working with ECMA Expressions,” on page 139. For descriptions of the system variables available in a workflow, see Section 4.3.3, “Understanding Workflow Data,” on page 60. |
| Final Timeout Action | Determines the final state of the request in the event that the workflow times out. The choices are:<br><br>• approved<br>• denied<br>• refused<br>• timedout<br>• error |
| Timeout Interval | Specifies a dynamic expression that defines the period of time allotted for the addressee to complete the task. The timeout interval applies each time the activity is executed by the addressee.<br><br>For details on building ECMA expressions, see Chapter 9, “Working with ECMA Expressions,” on page 139. For descriptions of the system variables available in a workflow, see Section 4.3.3, “Understanding Workflow Data,” on page 60. |
| Time Units | Determines the unit of measure used for the timeout interval. The choices are:<br><br>• Days<br>• Hours<br>• Minutes<br>• Seconds |
| Retries | Specifies the number of times to retry the activity in the event of a timeout.<br><br>When an activity times out, the workflow process can try to complete the activity again, depending on the retry count specified for the activity. With each retry, the workflow process can escalate the activity to another user. In this case, the activity is reassigned to another user (the user’s manager, for example) to give this user an opportunity to finish the work of the activity. If the last retry times out, the activity can be marked as approved, denied, refused, timedout, or in error, depending on the final timeout action specified for the activity. |

| Property Name | Description |
|---|---|
| Form | Specifies the name of the approval form to display to the user.<br><br>An Approval activity must have a form associated with it. If no form is specified, an error message is displayed at runtime. |

### Data Item Mapping

To bind the data items associated with the Approval activity, you define pre-activity and post-activity mappings. The pre-activity mappings initialize data in the approval form with constants, values retrieved from the flowdata object, system process variables, system activity variables, and/or data retrieved via expression calls to the directory abstraction layer. The post-activity mappings move form data back into the flowdata object.

***Table 8-4*** *Approval Activity Data Item Mappings*

| Setting | Description |
|---|---|
| Pre-Activity | Allows you to specify one or more pre-activity mappings. When this option is selected, you can double-click on a cell in the *Source Expression* column to specify where the approval form gets data for a particular target form field.<br><br>**NOTE:** When the *Pre-Activity* choice is selected, the cells in the *Target Form Field* column are not editable. |
| Post-Activity | Allows you to specify one or more post-activity mappings. When this option is selected, you can double-click on a cell in the *Target Expression* column to specify where data from a form field should be copied after the form has been processed.<br><br>The form for an Approval activity includes a special internal control called `apwaComment`. This control causes user comments to be written to the workflow database. It should not have a post-activity mapping. For more information on this control, see Section 6.5.6, “DNMaker,” on page 93.<br><br>**NOTE:** When the *Post-Activity* option is selected, the cells in the *Source Form Field* column are not editable. |
| Source Expression | Specifies a source expression for a pre-activity mapping. When you click a cell in the *Source Expression* column, the ECMA expression builder displays to help you define your expression. |
| Target Expression | Specifies a target expression for a post-activity mapping. When you click a cell in the *Target Expression* column, the ECMA expression builder displays to help you define your expression. |

For details on building ECMA expressions, see Chapter 9, “Working with ECMA Expressions,” on page 139.

### E-mail Notification

To enable e-mail notification for the Approval activity, you need to specify the e-mail template to use, as well as source expressions for target tokens in the e-mail body.

***Table 8-5*** *E-mail Notification Settings for the Approval Activity*

| Setting | Description |
|---|---|
| Email Template | Specifies the name of the e-mail template to use. By default, the Approval activity uses the Provisioning Notification template.<br><br>You can edit an e-mail template in Designer. For more information, see “Editing an e-mail template” on page 122. |
| Source/Target | Specifies the source expressions for target tokens in the e-mail body.<br><br>The list of target tokens is determined by the selected e-mail template. You cannot add new tokens, but you can assign values to the predefined tokens by building your own source expressions. At runtime, the source expressions are evaluated to determine the value of each token.<br><br>The available target tokens for the Provisioning Notification e-mail template are listed below:<br><br>• CC<br>• BCC<br>• recipientFullName<br>• initiatorFullName<br>• requestTitle<br>• userFirstName<br><br>If you use a provisioning request definition template to create your workflow, each token has a default source expression. The default expressions retrieve values from the workflow process (the process object) or from the data expression layer (IDVault object). You can modify these expressions to suit your application requirements.<br><br>For details on building ECMA expressions, see Chapter 9, “Working with ECMA Expressions,” on page 139. |

**NOTE:** E-mail notification is only supported when the *Notify participants by email* check box is selected on the *Overview* tab.

## 8.1.3 Log Activity

The Log activity is a system activity that writes messages to a log.

To log information about the state of a workflow process, the Workflow System interacts with Novell® Audit. During the course of its processing, a workflow can log information about various events that have occurred. Users can then use the Novell Audit reporting tools to look at logged data.

**NOTE:** During the course of workflow execution, many system events are logged that are not controlled by the Log Activity. For example, the Workflow System writes a message to the log whenever a workflow is started or stopped, or when it is approved, denied, or refused. For a complete list of the system events logged during workflow execution, see the chapter on setting up logging in the *Identity Manager User Application: Administration Guide*.

### Properties

The Log activity has the following properties:

***Table 8-6*** *Log Activity Properties*

| Property Name | Description |
|---|---|
| Name | Provides a name for the activity. |
| Audit | Specifies whether log messages should be sent. When this property is set to true, messages are sent to all log4j channels, including Novell Audit. When this property is set to false, no log messages are sent.<br><br>For details on logging setup and configuration, see the *Identity Manager User Application: Administration Guide*. |
| Author | Defines the author for the message. By default, the author is the initiator of the provisioning request. |
| Message | Specifies an ECMA expression that defines text for the log message. Typically, this text indicates where this Log activity is being executed within the process and provides other information that makes the log easy to understand.<br><br>For details on building ECMA expressions, see Chapter 9, “Working with ECMA Expressions,” on page 139. For descriptions of the system variables available in a workflow, see Section 4.3.3, “Understanding Workflow Data,” on page 60. |

### Data Item Mapping

Not supported with this activity.

### E-mail Notification

Not supported with this activity.

## 8.1.4 Branch Activity

In a workflow that supports parallel processing, the Branch activity allows multiple users to act on different areas of the work item in parallel. After the users have completed their work, the Merge activity synchronizes the incoming branches in the flow.

A workflow can have multiple Branch activities, but each Branch activity must have an associated Merge activity. All flow paths leading out of a Branch activity will execute.

### Properties

The Branch activity has the following property:

***Table 8-7*** *Branch Activity Properties*

| Property Name | Description |
|---|---|
| Name | Provides a name for the activity. |

### Data Item Mapping

Not supported with this activity.

### E-mail Notification

Not supported with this activity.

## 8.1.5 Merge Activity

In a workflow that supports parallel processing, the Merge activity synchronizes the incoming branches in the flow. The Merge activity is used in conjunction with the Branch activity, which allows two users to act on different areas of the work item in parallel. After the users have completed their work, the Merge activity synchronizes the incoming branches.

A workflow can have multiple Branch activities, but each Branch activity must have an associated Merge activity.

### Properties

The Merge activity has the following property:

***Table 8-8*** *Merge Activity Properties*

| Property Name | Description |
|---|---|
| Name | Provides a name for the activity. |

### Data item mapping

Not supported with this activity.

### Email notification

Not supported with this activity.

## 8.1.6 Condition Activity

The Condition activity lets you add conditional logic to a workflow. This logic can be used to control what happens when the workflow executes. In the Condition activity, you define logic as an ECMA expression that evaluates to a boolean value.

Each Condition activity must have two outgoing flow paths, one that handles conditions that evaluate to true and another that handles conditions that evaluate to false. Optionally, a third flow path can be added to handle error conditions that occur if the ECMA expression evaluation fails.

### Properties

The Condition activity has the following properties:

***Table 8-9*** *Condition Activity Properties*

| Property Name | Description |
|---|---|
| Name | Provides a name for the activity. |
| Condition Expression | Specifies an ECMA expression that returns true or false. The value returned determines which flow path is followed after the activity has finished executing.<br><br>**TIP:** If you need to test whether two objects are equal in a conditional expression, you should normally use the == operator, rather than the equals() method, unless you are certain that the objects being compared are Java objects of the same type. For instance, use this expression:<br><br>`(approval_A.getAction() == "DENIED")`<br><br>instead of this one:<br><br>`(approval_A.getAction()).equals("DENIED")`<br><br>For details on building ECMA expressions, see Chapter 9, “Working with ECMA Expressions,” on page 139. For descriptions of the system variables available in a workflow, see Section 4.3.3, “Understanding Workflow Data,” on page 60. |

### Data Item Mapping

Not supported with this activity.

### Email Notification

Not supported with this activity.

## 8.1.7 Finish Activity

The Finish activity marks the completion of a workflow. When the Finish activity executes, an e-mail message is sent to notify participants that the workflow has finished.

### Properties

The Finish activity has the following property:

***Table 8-10*** *Finish activity properties*

| Property | Description |
|---|---|
| Name | Provides a name for the activity. |

### Data Item Mapping

Not supported with this activity.

### E-mail Notification

To enable e-mail notification for the Finish activity, you need to specify the e-mail template to use, as well as source expressions for target tokens in the e-mail body.

***Table 8-11*** *Email notification settings for the Finish activity*

| Setting | Description |
|---|---|
| Email Template | Specifies the name of the e-mail template to use. By default, the Finish activity uses the Provisioning Approval Completed Notification template.<br><br>You can edit an e-mail template in Designer. For more information, see “Editing an e-mail template” on page 122. |

| Setting | Description |
|---|---|
| Source/Target | Specifies the source expressions for target tokens in the e-mail body.<br><br>The list of target tokens is determined by the selected e-mail template. You cannot add new tokens, but you can assign values to the predefined tokens by building your own source expressions. At runtime, the source expressions are evaluated to determine the value of each token.<br><br>The available target tokens for the Provisioning Approval Completed Notification e-mail template are listed below:<br><br>• CC<br>• BCC<br>• requestStatus<br>• requestSubmissionTime<br>• requestID<br>• recipientFullName<br>• initiatorFullName<br>• requestTitle<br><br>If you use a provisioning request definition template to create your workflow, each token has a default source expression. The default expressions retrieve values from the workflow process (the process object) or from the data expression layer (IDVault object). You can modify these expressions to suit your application requirements.<br><br>For details on building ECMA expressions, see Chapter 9, “Working with ECMA Expressions,” on page 139. |

**NOTE:** E-mail notification is only supported when the *Notify participants by email* check box is selected on the *Overview* tab.

## 8.1.8 Entitlement Activity

The Entitlement activity grants or revokes an entitlement for a user or other entity type.

A workflow must have at least one Entitlement or Entity activity.

### Properties

The Entitlement activity has the following properties:

*Table 8-12 Entitlement Activity Properties*

| Property Name | Description |
|---|---|
| Name | Provides a name for the activity. |
| Entity Type | Specifies the target entity type for the entitlement. |

## Data Item Mapping

To bind the data items associated with the Entitlement activity, you define mappings for several DirXML® attributes.

*Table 8-13 Entitlement Activity Data Item Mappings*

| Setting | Description |
|---|---|
| Source Expression | Specifies a source expression for a DirXML mapping. When you click a cell in the *Source Expression* column, the ECMA expression builder displays to help you define your expression.<br><br>The DirXML mappings for the Entitlement are described below:<br><br>• dn is the distinguished name for the recipient of the entitlement.<br>• DirXML-Entitlement-DN is the distinguished name of the entitlement to execute. For example, the entitlement might be identified as follows:<br><br>`'CN=Groups,CN=GroupEntitlementLoopback,CN=TestDrivers,O=novell'`<br><br>You can use the ECMA expression builder's ECMAScript Variable panel to see a list of all the entitlements in the driver. To select an entitlement, double-click the full distinguished name of the entitlement.<br>• DirXML-Entitlement-Action indicates whether the entitlement is granted or revoked. If the operation grants the entitlement, the value must be 1; if it revokes the entitlement, the value must be 0.<br>• DirXML-Entitlement-Parameter specifies a parameter required by the entitlement driver. For example, if the entitlement operation grants access to the Sales group, the parameter might specify the group as follows:<br><br>`'\\MYTREE\\novell\\idmsample-doc\\groups\\Sales'`<br><br>• DirXML-Entitlement-MultiValueAllowed indicates whether the entitlement supports multiple values. If it supports multiple values, the value must be true; otherwise, it must be false. |

For details on building ECMA expressions, see Chapter 9, "Working with ECMA Expressions," on page 139.

## E-mail Notification

Not supported with this activity.

## 8.1.9 Entity Activity

The Entity activity updates an entity in the Identity Vault. You can use this activity to create, modify, or delete attributes on an entity. You can also use this activity to create or delete an entity.

A workflow must have at least one Entitlement or Entity activity.

### Properties

The Entity activity has the following properties:

***Table 8-14*** *Entity Activity Properties*

| Property Name | Description |
|---|---|
| Name | Provides a name for the activity. |
| Entity Type | Specifies the target entity type. |
| Operation | Indicates what kind of operation will be performed on the target entity:<br>• Create/Modify<br>• Delete attributes/values<br>• Delete entity<br>To create or modify attributes of an entity or to create a new entity, select *create/modify*. To delete attributes of an entity, select *delete*.<br>To delete an entity, select *delete object*. |

### Data Item Mapping

To bind the data items associated with the Entity activity, you define mappings for the attributes associated with the target entity type.

***Table 8-15*** *Entity Activity Data Item Mappings*

| Setting | Description |
|---|---|
| Entity dn | Identifies the entity that is the target of the operation. The default value is recipient.<br><br>To create a new object, specify a distinguished name that does not yet exist.<br><br>**TIP:** The output of the DNMaker control can be used as input for the Entity dn value. The DNMaker control constructs the DN by allowing the user to enter the naming attribute in a text field and presenting an interface for picking a container. Once this data has been captured in a request form, the output can be mapped to a variable in the flowdata object. In the definition for the Entity activity, this flowdata variable can be accessed in the Entity dn setting with an expression such as:<br><br>`flowdata.get('groupdn');`<br><br>For details on using the DNMaker control, see Section 6.5.6, "DNMaker," on page 93. |
| Modify Type | Indicates how the mapping should be performed for an attribute. The choices are:<br><br>• Append Value<br>• Replace Value<br>• Replace All Values<br><br>For many attributes, *Replace Value* is the only option that makes sense; therefore, this option is selected automatically and cannot be changed.<br><br>You must specify the *Modify Type* setting before specifying the *Modify Value Expression* setting. |
| Modify Value Expression | Specifies a source expression for an attribute. When you click a cell in the *Modify Value Expression* column, the ECMA expression builder displays to help you define your expression. The list of attributes available varies depending on which entity type was selected on the Properties tab.<br><br>**NOTE:** The cells in the *Target Attribute* column are not editable. |

### Email notification

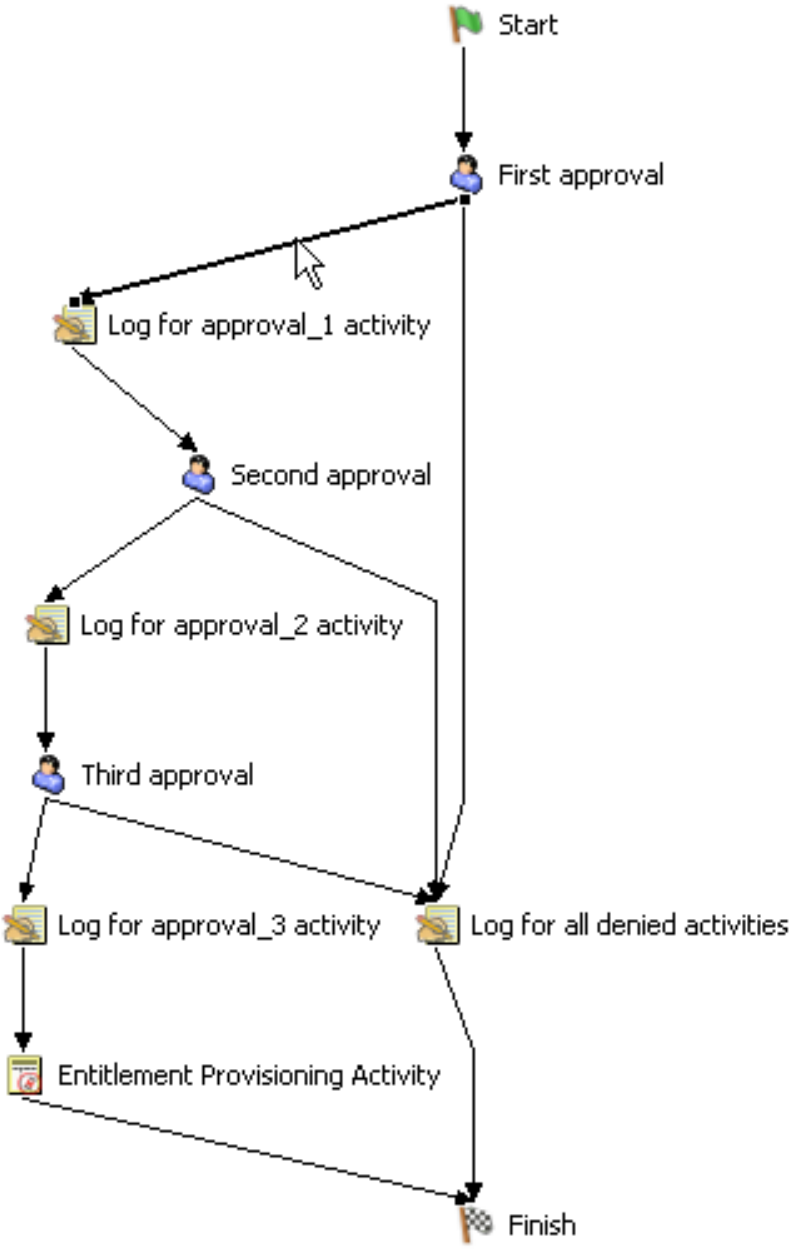
Not supported with this activity.

# 8.2 Configuring Flow Paths

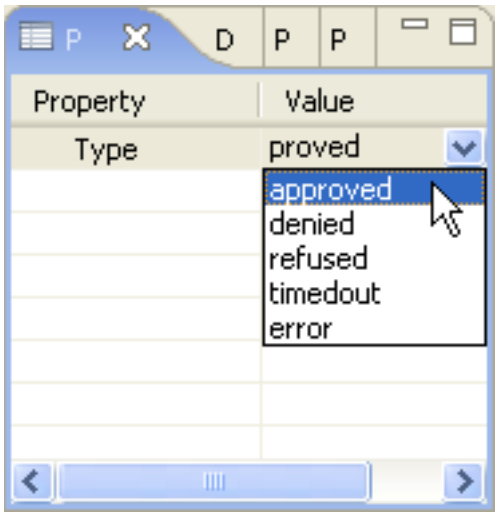
After you've added a flow path to a workflow diagram, you can specify the path type. For details on adding flow paths to a workflow, see Section 7.4, "Adding the Flow Paths," on page 117.

To configure a flow path:

1 Click the flow path in the workflow diagram.



2 Set the flow type on the *Properties* tab by selecting one of the options in the *Type* drop-down list.




The flow path types are described below:

| Flow Type | Description |
|---|---|
| forward | Forwards control to the next activity in a workflow.<br>The forward flow path is available after all activities except:<br>• Approval<br>• Condition<br>• Finish |
| approved | Determines what happens when a user approves a request.<br>The approved flow path is valid only after the Approval activity. |
| denied | Determines what happens when a user denies a request.<br>The denied flow path is valid only after the Approval activity. |
| refused | Determines what happens when a user refuses a request.<br>The refused flow path is valid only after the Approval activity. |
| timedout | Determines what happens when an Approval activity times out because the user did not respond.<br>The timedout flow path is valid only after the Approval activity. |
| error | Determines what happens when an Approval or Condition activity terminates with an error.<br>The error flow path is valid only after the Approval and Condition activities. |
| true | Determines what happens when a conditional expression evaluates to true.<br>The true flow path is valid only after the Condition activity. |

| Flow Type | Description |
| --- | --- |
| false | Determines what happens when a conditional expression evaluates to false.<br><br>The false flow path is valid only after the Condition activity. |

If the *Properties* tab is not displayed, right-click the flow path in the workflow diagram and select *Show Properties*.

# Working with ECMA Expressions

# 9

This section provides details on using the ECMA expression builder. Topics include:



## 9.1 About the ECMA Expression Builder

Designer incorporates an ECMAScript interpreter and expression editor, which allows you create script expressions that refer to and modify workflow data. For example, you can use scripting to:

- Create new data items needed in a workflow under the flowdata element.
- Perform basic string, date, math, relational, concatenation, and logical operations on data.
- Call standard or custom Java classes for more sophisticated data operations.
- Use expressions for runtime control to:
  - Modify or override form field labels.
  - Initialize form field data.
  - Customize e-mail addresses and content.
  - Set entitlement grant/revoke rights and parameters.
  - Evaluate any past activity data to conditionally follow a work flow path using the Condition activity.
  - Write different log messages that are conditionally triggered using a single Log activity.

This section discusses some of the techniques and capabilities applicable to the use of scripting.

---

**NOTE:** To define expressions for a workflow, you need to understand how workflow activities are configured. In addition, you need to understand the various types of data that are available within a workflow. For details on configuring workflow activities, see Section 8.1, “Configuring Activities,” on page 119. For descriptions of the system variables available in a workflow, see Section 4.3.3, “Understanding Workflow Data,” on page 60.

---

### 9.1.1 About ECMAScript

ECMAScript is an object-oriented scripting language for manipulating objects in a host environment (in this case, Designer). ECMAScript (ECMA-262 and ISO/IEC 16262) is the standards-based scripting language underpinning both JavaScript* (Netscape*) and JScript* (Microsoft*). It is designed to complement and extend existing functionality in a host environment such as Designer’s graphical user interface. As a host environment, Designer provides ECMAScript access to various objects for processing. ECMAScript in turn provides a Java-like language that can operate on those objects.

The extensibility of ECMAScript, and its powerful string-handling tools (including regular expressions) make it an ideal language to extend the functionality of Designer.

**NOTE:** You can find detailed information about ECMAScript at the European Computer Manufacturers Association (ECMA) Web site (http://www.ecma-international.org).

## 9.1.2 ECMAScript Capabilities

In addition to enabling you to incorporate finely-tuned custom logic into your workflow, scripting gives you a great deal of flexibility in manipulating data, because of the various DOM- and XPath-related objects and methods available in the ECMAScript extensions incorporated into the Expression Builder.

The usefulness of ECMAScript is especially apparent when dealing with in-memory DOMs. The ECMA expression builder constructs XML documents as in-memory objects according to the W3C DOM Level 2 specification. The DOM-2 specification, in turn, defines an ECMAScript binding (see the W3C Recommendation ECMAScript Language Binding (http://www.w3.org/TR/DOM-Level-2-Core/ecma-script-binding.html), with numerous methods and properties that provide ready access to DOM-tree contents. The flowdata DOM is recognized by the ECMA expression builder, and any of the W3C-defined ECMAScript extensions that apply to DOMs can be used.

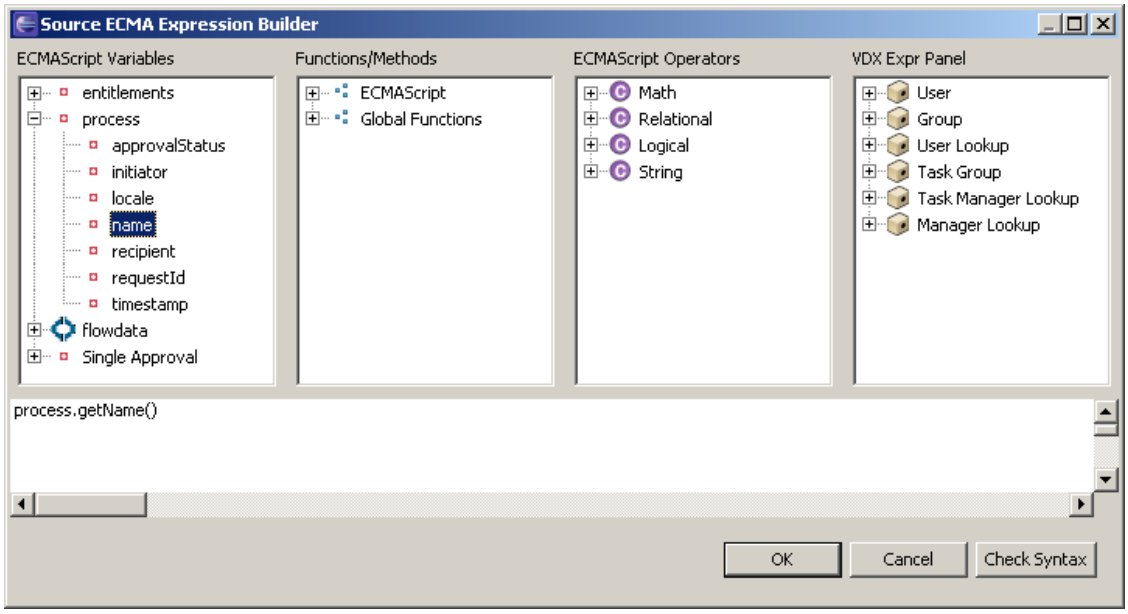
ECMAScript also provides bridges to other expression languages such as XPath. This allows you to use XPath syntax on a DOM to address various elements within its document structure.

## 9.1.3 Using the ECMA Expression Builder

Designer provides access to ECMAScript in various places in the User Application design tools. The most common form of access is through the Expression Builder, which can be displayed whenever you see this button:

The button can be found in Designer displays, such as the Properties for a Condition activity, or the Data Item Mapping view for an Entitlement Provisioning activity. Click the button to display the ECMA expression builder.

***Figure 9-1*** *ECMA Expression Builder*




The ECMA expression builder provides pick lists of available objects, methods, and properties in the top panes (all of which are resizable), with rollover tool tips to help you build ECMAScript statements. Double-clicking any item in any pick list causes a corresponding ECMAScript statement to appear in the edit pane in the lower portion of the window. In the figure, the *process* pick list has been selected in the *ECMAScript Variables* pane, and the *name* variable has been double-clicked. The ECMAScript expression that can access the contents of this workflow variable is inserted automatically in the edit pane.

This section includes the following topics:



### Checking Syntax

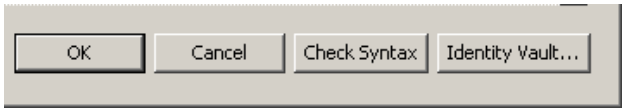
The ECMA expression builder includes a *Check Syntax* button. Clicking the button causes the ECMAScript interpreter to check the syntax of the expression. If there are problems involving ECMAScript syntax, an error message is displayed. You can then edit the expression and validate again as needed. Validation is optional.

**NOTE:** The syntax checking process does not execute your expression. It just checks syntax. Because ECMAScript is an interpreted language, syntax checking doesn't check any runtime-dependent expressions, other than to see if they conform to valid ECMAScript syntax.

### Selecting a DN

The ECMA expression builder also includes an *Identity Vault* button that is displayed when you are working with activities (for example, Start, Approval, and Entitlement activities) that may require selecting a dn from the Identity Vault.

***Figure 9-2*** *Identity Vault Button*




The *Identity Vault* button displays a dialog box that you use to navigate the Identity Vault to select a dn. The Identity Vault button is grayed (to indicate that it is unavailable) if you are not connected to the Identity Vault.

### ECMAScript Variables

This pane displays the names of variables that are relevant in the current context. For example, if you are working in the provisioning request definition editor, you see system variables for the current workflow process, system variables for the current activity, and flowdata variables created in the current workflow. Double-click the name of a variable to insert that variable into your script. For descriptions of the system variables available in a workflow, see Section 4.3.3, "Understanding Workflow Data," on page 60.

The ECMA expression builder provides two methods for reading flowdata variables.

***Table 9-1*** *Methods for Reading Flowdata Variables*

| Method | Description |
|---|---|
| flowdata.get(variable-name) | Returns a string as the node value for a variable (representing an XPath expression) in the workflow document. |
| flowdata.getObject(variable-name) | Returns an object as a node value for a variable (representing an XPath expression) in the workflow document. Use this method to retrieve the values of multivalued controls. |

### Functions/Methods

For a description of the functions and methods available in the ECMA expression builder, see Section 9.3, "ECMAScript API," on page 149.

### ECMAScript Operators

The following tables provide descriptions of the operators supported by the ECMA expression builder.

***Table 9-2*** *Math*

| Operator | Description |
|---|---|
| + Add | Returns the sum of two numerical values (either literals or variables). |
| - Subtract | Subtracts one number from another |
| * Multiply | Returns the product of two numerical values (either literals or variables). |
| / Divide | Divides one number by another. |

***Table 9-3*** *Assignment*

| Operator | Description |
|---|---|
| = Assignment | Assigns value of right operand to the left operand. |
| += Add to this | Adds the left and right operands and assigns the result to the left operand. For example, a += b is the same as a = a + b. |
| -= Subtract from this | Subtracts the right operand from the left operand and assigns the result to the left operand. For example, a -= b is the same as a = a - b. |
| *= Multiply to this | Multiplies the two operands and assigns the result to the left operand. For example, a *= b is the same as a = a * b. |
| /= Divide this to | Divides the left operand by the right operand and assigns the result to the left operand. For example, a /= b is the same as a = a / b. |
| %= Modulus | Divides the left operand by the right operand and assigns the remainder to the left operand. For example, a %= b is the same as a = a % b. |
| &= Apply bitwise AND to this | Performs bitwise AND on operands and assigns the result to the left operand. For example, a &= b is the same as a = a & b. |
| \|= Apply bitwise OR to this | Performs bitwise OR on operands and assigns the result to the left operand. For example, a \|= b is the same as a = a \| b. |
| <<= Apply bitwise left shift to this | Performs bitwise left shift on operands and assigns the result to the left operand. For example, a <<= b is the same as a = a << b. |
| >>= Apply bitwise signed right shift to this | Performs bitwise right shift on operands and assigns the result to the left operand. For example, a >>= b is the same as a = a >> b. |

| Operator | Description |
|---|---|
| >>>= Apply bitwise unsigned right shift to this | Performs bitwise unsigned right shift on operands and assigns the result to the left operand. For example, a >>>= b is the same as a = a> >> b. |

***Table 9-4*** *Other*

| Operator | Description |
|---|---|
| % Modulus | Divides the left operand by the right operand and returns the integer remainder. |
| ++ Autoincrement | Increments the operand by one (can be used before or after the operand). |
| -- Autodecrement | Decrements the operand by one (can be used before or after the operand). |
| ~ Bitwise NOT | Inverts the bits of its operand. |
| & Bitwise AND | Returns a 1 in each bit position for which the corresponding bits of both operands are ones. |
| \| Bitwise OR | Returns a 1 in each bit position for which the corresponding bits of either or both operands are ones. |
| ^ Bitwise XOR | Returns a 1 in each bit position for which the corresponding bits of either but not both operands are ones. |
| << Bitwise left shift | Shifts the digits of the binary representation of the first operand to the left by the number of places specified by the second operand. The spaces created to the right are filled in by zeros, and any digits shifted to the left are discarded. |
| >> Signed bitwise right shift | Shifts the digits of the binary representation of the first operand to the right by the number of places specified by the second operand, discarding any digits shifted to the right. The copies of the leftmost bit are added on from the left, preserving the sign of the number. |
| >>> Unsigned bitwise right shift | Shifts the binary representation of the first operand to the right by the number of places specified by the second operand. Bits shifted to the right are discarded and zeroes are added to the left. |

***Table 9-5*** *Relational*

| Operator | Description |
|---|---|
| == Equal | Assigns the value of the right operand to the left operand. |
| != Not Equal | Returns a Boolean true if both the operands are not equal. |
| < Less than | Returns true if the left operand is less than the right operand. |
| > Greater than | Returns true if the left operand is greater than the right operand. |

| Operator | Description |
|---|---|
| <= Less than or equal | Returns true if the left operand is less than or equal to the right operand. |
| >= Greater than or equal | Returns true if the left operand is greater than or equal to the right operand. |

***Table 9-6*** *Logical*

| Operator | Description |
|---|---|
| && AND | Returns a Boolean true if both the operands are true; otherwise, returns false. |
| \|\| OR | Returns true if either operand is true. Returns false when both operands are false. |
| ! NOT | Returns false if its single operand can be converted to true (or if it is a non-Boolean value). Returns true if its operand can be converted to false. |

***Table 9-7*** *String*

| Operator | Description |
|---|---|
| + Concatenate | Concatenates two string operands, returning a string that is the union of the two operand strings. |

### VDX Expr

This pane allows you to insert Entity definitions (see Section 3.2, “Working with Entities and Attributes,” on page 24) that are defined in the Identity Vault into your scripts. Both system and user-defined entities are available. The format of an expression to retrieve data from the Identity Vault is:

```
IDVault.get(dn, object-type, attribute)
```

For example if you want the recipient's manager for a data item, you would open the User node in the VDX Expr Panel and double-click the Manager item, which inserts `IDVault.get({ enter dn expression here }, 'user', 'manager')`. This expression evaluates to the string for the manager's dn (LDAP distinguished name).

## 9.1.4 About Java Integration

Java is integrated into the workflow process through the ECMA expression builder, which provides a bridge to external Java objects. To access a Java class through the ECMA expression builder, the class must be in the classpath of the workflow engine. To accomplish this, you must add the Java class to the `WEB-INF\lib` directory in the user application WAR file (`IDM.war`).

### Adding the Java Class to the User Application WAR

1. Use a WAR file utility to open the `IDM.war` file. The `IDM.war` file is located in the application server `\server\IDM\deploy` directory.
2. Copy the Java class into the `WEB-INF\lib` directory.

### Accessing Java from ECMAScript

To access a Java class, create a function inline in the ECMA expression builder. Instantiate the function, then within the function, using ECMAScript syntax, call your Java methods. The following example creates a vector:

```
function list() { v=new java.util.Vector(); v.add('{Enter Item 1}');
v.add('{Enter Item 2}'); return v; };  list();
```

To access a custom Java class, you must preface the class name with “Packages”. For example:

```
v = new Packages.com.novell.myClass("value");
```

## 9.1.5 About XPath Integration

A provisioning request definition workflow supports a special object called *flowdata* (see Section 4.3.3, “Understanding Workflow Data,” on page 60). The flowdata object is a DOM (an XML document constructed as an object in memory). You can use XPath syntax to navigate the structure of the flowdata DOM, and add, modify or delete elements and contents.

To add an object to flowdata:

| Syntax | Examples |
|---|---|
| `flowdata.{arguments}`<br>ECMAScript XPath | `flowdata.parent/child[1]`<br>`flowdata.reason` |

To get an object from flowdata:

| Syntax | Examples |
|---|---|
| `flowdata.get{arguments}`<br>ECMAScript XPath | `flowdata.getObject('parent/child[1]')`<br>`flowdata.get('reason')` |

For information about the flowdata.get() and flowdata.getObject() methods, see Table 9-1 on page 142.

## 9.1.6 Performance Considerations

ECMAScript is an interpreted language, which means that every line of script in an expression must be parsed and translated to the Java equivalent before it can be executed. This adds considerable overhead to the code and results in overall slower execution of scripts than pure Java. Before using ECMAScript, you should think about the possible performance ramifications.

The following guidelines will help you to achieve optimal performance in your components and services:

- Consider whether a task can be accomplished using a custom Java class (which you can call from ECMAScript).
- When you need the fine control offered by scripting, use ECMAScript.

Bear in mind that the key to good performance is always a good implementation (for example, choosing the correct algorithm, attention to reuse of variables). Good code written in a slow language often outperforms bad code written in a fast language. Writing something in Java does not guarantee that it will be faster than the equivalent logic written in ECMAScript, because Java has its own overhead constraints. For example, constructor call-chains (when you call a constructor for a Java object that inherits from other objects, the constructors for all ancestral objects are also called).

ECMAScript's core objects (String, Array, Date, etc.) have many built-in convenience methods for data manipulation, formatting, parsing, sorting, and conversion of strings and arrays. These methods are implemented in highly optimized Java code inside the interpreter. It is to your advantage to use these methods whenever possible, rather than create customized data-parsing or formatting functions. For example, suppose you want to break a long string into substrings, based on the occurrence of a delimiter. You could create a loop that uses the String methods indexOf() and substring() to parse out the substrings and assign them to slots in an array. But this would be a very inefficient technique when you could simply do:

```
var myArrayOfSubstrings = bigString.split( delimiter );
```

The ECMAScript String method split() breaks a string into an array of substrings based on whatever delimiter value you supply. It executes in native Java and requires the interpreter to interpret only one line of script. Trying to do the same thing with a loop that iteratively calls indexOf() and substring() would involve a great deal of needless interpreter and function-call overhead, with the accompanying performance hit.

Skillful use of built-in ECMAScript methods pays worthwhile performance dividends. If you use scripts extensively, take time to learn about the fine points of the ECMAScript language, because this can help you eliminate performance bottlenecks.

# 9.2 ECMAScript Examples

This section provides examples of common operations that you can perform using ECMAScript.

## 9.2.1 General Examples

This section presents examples that illustrate basic scripting techniques.

#### Using a Function

To create a function in the ECMA expression builder, create the function inline:

```
function abc() { var v1 = "" ; for ( i = 0; i < 9 ; i++) v1 += "$";
return v1; } ; abc();
```

## 9.2.2 Flowdata Examples

This section presents scripting examples that show the use of the flowdata object.

### Getting the Value of a Flowdata Variable

In the previous example, you entered information about an approval status into the flowdata by creating an XML element named start_reason with a child element named approval_reason and an attribute named ApprovalStatus. Use the following expression, in a pre-activity map, to retrieve the value of the ApprovalStatus attribute:

```
flowdata.get('start_reason/approval_reason/@ApprovalStatus')
```

You can enter this expression by expanding the *flowdata* nodes in the ECMAScript Variables pane of the ECMA expression builder, and double-clicking the ApprovalStatus attribute.

***Figure 9-3*** *Selecting an Attribute*


### Creating an XML Element With Child Element and Add it to the Flowdata

In the previous example, you retrieved user input to the form field ApprovalStatus. Now we want to add this information to the flowdata so that it can be used by a downstream activity. Use the following expression in a post-activity map:

```
flowdata.start_reason/approval_reason/@ApprovalStatus
```

## 9.2.3 Form Control Examples

This section presents several examples of scripting with form controls.

### Retrieving the Value of a Form Field

For example, create a form field named ApprovalStatus. To get the value of this field, use the following expression in a pre-activity map:

```
process.get('ApprovalStatus')
```

You can enter this expression by opening the Process node in the ECMAScript Variables pane of the ECMA expression builder, and double-clicking ApprovalStatus.

### Getting an Individual Value From a Multivalued Control

To get an individual value from a multivalued control (for example, a check box named *colors*), you first need to get the control into the flowdata. In the post-activity mapping for an upstream activity, use:

```
flowdata.colors
```

To get a value from colors (for example, the first value), use the following expression on a downstream activity:

```
flowdata.getObject('colors[1]')
```

### Populating a List or Checkbox Item

To populate list controls (for example, PickList or MVEditor) or the MVCheckbox control using script, use an expression like this in the pre-activity mapping:

```
function list() {var l=new
java.util.Vector();l.add('Blue');l.add('Red'); l.add('Green'); return
l;} list();
```

## 9.2.4 Error Handling Examples

This section presents scripting examples that show how to deal with errors during runtime execution.

### Handling Errors

The approach to handling errors differs between pre-activity and post-activity maps. For post-activity maps, you can use an error flow path from an Approval or Condition activity to catch errors that occur during post-activity mapping. This approach doesn't work for pre-activity maps, because any errors that occur in the process of getting data happen before the form is displayed to the user. When this occurs, an error message similar to the following appears in place of form controls in the bottom portion of forms displayed to the user:

```
XXXX FAILED to generate form due to: No data items are available!
```

In this scenario, you can use a try-catch statement in a source expression for a field in a pre-activity map:

```
function getTheData()

{

   var theData;

   try {

      theData = IDVault.get( 'cn=jsmith,ou=users,ou=idmsample1,o=acme'
, 'user', 'FirstName') + ' ' + IDVault.get (
'cn=jsmith,ou=users,ou=idmsample1,o=acme'  , 'user', 'LastName');

    }

   catch (error) { theData = 'Error retrieving data.'; }

   return theData;

};

getTheData();
```

# 9.3 ECMAScript API

This section includes the following topics:

1. Section 9.3.1, "Array Object," on page 150

## 9.3.1 Array Object

Lets you work with arrays.

#### Array(item0, item1, . . .)

```
Array()
```

Constructor

#### join(separator)

```
Array join(separator)
```

The elements of the array are converted to strings, and these strings are then concatenated, separated by occurrences of the separator. If no separator is provided, a single comma is used as the separator.

#### length

Array length The length property of this Array object

#### reverse()

```
reverse()
```

The elements of the array are rearranged so as to reverse their order. The operation is done in-place, meaning that the original array is modified.

#### sort(comparefn)

```
Array sort()
```

The elements of this array are sorted. The sort is not necessarily stable. If comparefn is supplied, it should be a function that accepts two arguments x and y and returns a negative value if x < y, zero if x = y, or a positive value if x > y.

#### toString()

```
Array toString()
```

The elements of this object are converted to strings, and these strings are then concatenated, separated by comma characters. The result is the same as if the built-in join method were invoked for this object with no argument.

## 9.3.2 Boolean Object

There is seldom a need to use the object version of Boolean in place of true/false literal values. This object is provided for completeness. It is specified in ECMA-262.

### Boolean()

```
Boolean( [true/false] )
```

Constructor. Optionally takes one of true or false as an argument.

### toString()

```
Boolean toString()
```

If this Boolean value is true, then the string “true” is returned. Otherwise, this Boolean value must be false, and the string “false” is returned.

### valueOf()

```
Boolean valueOf()
```

Returns this Boolean value.

## 9.3.3 Date Object

Lets you work with dates and times.

### Date()

```
Date()
```

The constructor of the Date can have various signatures. The date constructor format can accept up to 7 parameters. Here is the format: new Date(year,month,date,hrs,mins,secs,ms)

### getDate()

```
getDate()
```

Returns DateFromTime(LocalTime(t)).

### getDay()

```
getDay()
```

Returns WeekDay(LocalTime(t)). The days of week are numbered from 0 -6. The number 0 represents Sunday and 6 represents Saturday.

### getFullYear()

```
getFullYear()
```

Returns YearFromTime(LocalTime(t)).

### getHours()

```
getHours()
```

Returns HourFromTime(LocalTime(t)).

### getMilliseconds()

```
getMilliseconds()
```

Returns msFromTime(LocalTime(t)).

### getMinutes()

```
getMinutes()
```

Returns MinFromTime(LocalTime(t)).

### getMonth()

```
getMonth()
```

Returns MonthFromTime(LocalTime(t)). The months are returned as an integer value from 0-11. The number 0 represents January and 11 represents December.

### getSeconds()

```
getSeconds()
```

Returns SecFromTime(LocalTime(t)).

### getTime()

```
getTime()
```

Returns a number, which is this time value. The number value is a millisecond representation of the specified Date object.

### getTimezoneOffset()

```
getTimezoneOffset()
```

Returns (t * LocalTime(t)) / msPerMinute. The difference is in minutes between (GMT) and local time.

### getUTCDate()

```
getUTCDate()
```

Returns DateFromTime(t).

### getUTCDay()

```
getUTCDay()
```

Returns WeekDay(t). The days of week are numbered from 0 -6. The number 0 represents Sunday and 6 represents Saturday.

### getUTCFullYear()

```
getUTCFullYear()
```

Returns YearFromTime(t). There is no getYearUTC method, so this method must be used to obtain a year from an UTC Date object.

### getUTCHours()

```
getUTCHours()
```

Returns HourFromTime(t).

### getUTCMilliseconds()

```
getUTCMilliseconds()
```

Returns msFromTime(t).

### getUTCMinutes()

```
getUTCMinutes()
```

Returns MinFromTime(t).

### getUTCSeconds()

```
getUTCSeconds()
```

Returns SecFromTime(t).

### getYear()

```
getYear()
```

Returns YearFromTime(LocalTime(t))—1900. \The function getFullYear() is preferred for nearly all purposes, because it avoids the year 2000 problem.

### parse(string)

```
parse(string)
```

Applies the ToString operator to its argument and interprets the resulting string as a date; it returns a number, the UTC time value corresponding to the date. The string is interpreted as a local time, a UTC time, or a time in some other time zone, depending on the contents of the string.

### setDate(date)

```
setDate(date)
```

Sets the day of the month, using an integer from 1 to 31, for the supplied date according to local time.

### setFullYear(year[,mon[,date]])

```
setFullYear(year[,mon[,date]])
```

Sets the [Value] property of this value to UTC ECMAScript.Date. Returns the value of the [Value] property of this value.

### setHours(hour[,min[,sec[,ms]]])

```
setHours(hour[,min[,sec[,ms]]])
```

Sets the [Value] property of this value to UTC time. Returns the value of the [Value] property of this value. When entering a value for hours, an hour value greater than 23 is added to the exisitng hour value, not set.

### setMilliseconds(ms)

```
setMilliseconds(ms)
```

Computes UTC from argument and sets the [Value] property of this value to TimeClip(calculatedUTCtime). Returns the value of the [Value] property of this value.

### setMinutes(min[,sec[,ms]])

```
setMinutes(min[,sec[,ms]])
```

Sets the [Value] property of this value to UTC time. Returns the value of the [Value] property of this value.

### setMonth(mon[,date])

```
setMonth(mon[,date])
```

Sets the [Value] property of this value to UTC ECMAScript.Date. Returns the value of the [Value] property of this value. If the [Value] property of this exceeds 11, the [Value] property for this is added to the existing month, not set.

### setSeconds(sec [, ms ] )

```
setSeconds(sec [, ms ] )
```

Sets the [Value] property of this value to UTC time. Returns the value of the [Value] property of this value.

### setTime(time)

```
setTime(time)
```

Sets the [Value] property of this value to TimeClip(time). Returns the value of the [Value] property of this value. The [Value] property of this is a millisecond value that is converted by the TimeClip(time) method.

### setUTCDate(date)

```
setUTCDate(date)
```

Sets the [Value] property of this value to ECMAScript.Date. Returns the value of the [Value] property of this value. If the [Value] property of this exceeds 30 or 31, the [Value] of this is added to the existing date value, not set.

### setUTCFullYear(year[,mon[,date]])

```
setUTCFullYear(year[,mon[,date]])
```

Sets the [Value] property of this value to ECMAScript.Date. Returns the value of the [Value] property of this value.

### setUTCHours(min[,sec[,ms]])

```
setUTCHours(min[,sec[,ms]])
```

Sets the [Value] property of this value to time. Returns the value of the [Value] property of this value. When entering a value for hours, an hour value greater than 23 is added to the exisitng hour value, not set.

### setUTCMilliseconds(ms)

```
setUTCMilliseconds(ms)
```

Sets the [Value] property of this value to time and returns the value of the [Value] property of this value.

### setUTCMinutes(min[,sec[,ms]])

```
setUTCMinutes(min[,sec[,ms]])
```

Sets the [Value] property of this value to time. Returns the value of the [Value] property of this value.

### setUTCMonth(mon[,date])

```
setUTCMonth(mon[,date])
```

Sets the [Value] property of this value to ECMAScript.Date. Returns the value of the [Value] property of this value. If the [Value] property of this exceeds 11, the [Value] property for this is added to the existing month, not set

### setUTCSeconds(sec [, ms ] )

```
setUTCSeconds(sec [, ms ] )
```

Sets the [Value] property of this value to time. Returns the value of the [Value] property of this value.

### setYear(year)

```
setYear(year)
```

Sets the [Value] property of this value to UTC ECMAScript.Date. Returns the value of the [Value] property of this value.

#### toLocaleString()

```
toLocaleString()
```

Returns a string value. The contents of the string are implementation-dependent, but are intended to represent the Date in a convenient, human-readable form appropriate to the geographic or cultural locale.

#### toString()

```
toString()
```

Returns this string value. The contents of the string are implementation-dependent, but are intended to represent the Date in a convenient, human-readable form in the current time zone.

#### toUTCString()

```
toUTCString()
```

Returns a string value. The contents of the string are implementation-dependent, but are intended to represent the Date in a convenient, human-readable form in UTC.

#### UTC()

```
UTC()
```

This method can accept number of different arguments. The UTC function differs from the Date constructor in two ways: it returns a time value as a number, rather than creating a Date object, and it interprets the arguments in UTC rather than as local time.

#### valueOf()

```
valueOf()
```

Returns a number, which is this time value. The valueOf() function is not generic, so it generates a runtime error if the object is not a Date object.

### 9.3.4 Function Object

Used to work with the Function Object.

#### Function(p1, p2, . . . , pn, body)

Function Constructor. The last argument specifies the body (executable code) of a function; any preceding arguments specify formal parameter.

#### length

The value of the length property is usually an integer that indicates the “typical” number of arguments expected by the function. However, the language permits the function to be invoked with

some other number of arguments. The behavior of a function when invoked on a number of arguments other than the number specified by its length property depends on the function.

#### toString()

```
String toString()
```

An implementation-dependent representation of the function is returned. This representation has the syntax of a FunctionDeclaration. The use and placement of whitespace, line terminators, and semicolons within the representation string is implementation-dependent.

### 9.3.5 Global

ECMAScript provides certain “top-level” methods and properties, so-called because they are available from any context: They are not parented by any particular object.

#### escape(string)

```
String escape()
```

The escape function computes a new, URL-legal version of a string in which certain URL-illegal characters have been replaced by hexadecimal escape sequences.

#### eval(x)

```
eval()
```

When the eval function is called with one argument x, the following steps are taken:

1. If x is not a string value, return x.
2. Parse x as an ECMAScript Program. If the parse fails, generate a runtime error.
3. Evaluate the program from Step 2.
4. If Result(3) is “normal” completion after value “V”, return the value V.
5. Return undefined.

#### Infinity

A special primitive value representing positive infinity.

#### isFinite(number)

```
isFinite()
```

Applies Number( ) to its argument, then returns false if the result is NaN, +*, or **; otherwise, returns true.

#### isNaN( value )

```
isNan()
```

Returns true if the argument evaluates to NaN (“not a number”); otherwise, returns false.

**NOTE:** Any form of logical comparison of NaN against anything else, including itself, returns false. Use isNaN( ) to determine whether a variable (or a return value, etc.) is equal to NaN.

#### NaN

The primitive value NaN represents the set of IEEE standard “Not-a-Number” values.

#### parseFloat(string)

```
number parseFloat()
```

Produces a floating-point number by interpretation of the contents of the string argument. If the string cannot be converted to a number, the special value NaN (see “NaN” on page 158) is returned.

#### parseInt(string, radix)

```
number parseInt()
```

Produces an integer value dictated by interpretation of the contents of the string argument, according to the specified radix.

#### unescape(string)

```
String unescape()
```

Computes a new version of a string value in which escape sequences that might be introduced by the escape function are replaced with the character they represent.

### 9.3.6 Math Object

All of the Math object’s properties and methods are static, which means you should prepend “Math” to the property or method name in your code. For example, use “Math.PI,” not simply “PI.”

#### E

The number value for e, the base of the natural logarithms, which is approximately 2.7182818284590452354.

#### LN10

The number value for the natural logarithm of 10, which is approximately 2.302585092994046.

#### LN2

The number value for the natural logarithm of 2, which is approximately 0.6931471805599453.

#### LOG2E

The number value for the base-2 logarithm of e, the base of the natural logarithms; this value is approximately 1.4426950408889634. The value of Math.LOG2E is approximately the reciprocal of the value of Math.LN2.

### LOG10E

The number value for the base-10 logarithm of e, the base of the natural logarithms; this value is approximately 0.4342944819032518. The value of Math.LOG10E is approximately the reciprocal of the value of Math.LN10.

### PI

The number value for *, the ratio of the circumference of a circle to its diameter, which is approximately 3.14159265358979323846.

### SQRT1.2

The number value for the square root of 1/2, which is approximately 0.7071067811865476. The value of Math.SQRT1_2 is approximately the reciprocal of the value of Math.SQRT2.

### SQRT2

The number value for the square root of 2, which is approximately 1.4142135623730951.

### abs(x)

```
Number abs(x)
```

Returns the absolute value of the argument x; in general, the result has the same magnitude as the argument but has positive sign. The input value x can be any number value.

Example:

```
Math.abs(-123.23940) = 123.23940
```

### acos(x)

```
Number acos(x)
```

This function returns an implementation-dependent approximation to the arc cosine of the argument. The result is expressed in radians and ranges from +0 to +PI(3.14159...)radians. The input value x must be a number between -1.0 and 1.0:

Example:

```
 PI/4 = 0.785 Math.acos(0.785) = 0.6681001997570769
```

### asin(x)

```
Number asin(x)
```

This function returns an implementation-dependent approximation to the arc sine of the argument. The result is expressed in radians and ranges from -PI/2 to +PI/2. The input value x must be a number between -1.0 and 1.0.

Example:

```
PI/4 = 0.785 Math.asin(0.785) = 0.9026961270378197
```

### atan(x)

```
Number atan(x)
```

This function returns an implementation-dependent approximation to the arc tangent of the argument. The result is expressed in radians and ranges from -PI/2 to +PI/2. The input value x can be any number.

Example:

```
3PI/4 = 2.355 Math.atan(2.355) = 1.169240427545485
```

### atan2(x,y)

```
Number atan2(x,y)
```

This function returns an implementation-dependent approximation to the arc tangent of the quotient y/x of the arguments y and x, where the signs of the arguments are used to determine the quadrant of the result. It is intentional and traditional for the two-argument arc tangent function that the argument named y be first and the argument named x be second. The result is expressed in radians and ranges from -PI to +PI. The input value x is the x-coordinate of the point. The input value y is the y-coordinate of the point.

Example:

```
PI/2 = 1.57 Math.atan2(1.57,-1.57) = 2.356194490192345
```

### ceil(x)

```
Number ceil(x)
```

This function returns the smallest (closest to -infinity) number value that is not less than the argument and is equal to a mathematical integer. If the argument is already an integer, the result is the argument itself. The input value x can be any numeric value or expression. The Math.ceil(x) function property is the same as -Math.floor(-x). Example:

Example:

```
Math.ceil(123.78457) = 123
```

### cos(x)

```
Number cos(x)
```

This function returns an implementation-dependent approximation to the cosine of the argument. The argument must be expressed in radians.

### exp(x)

```
Number exp(x)
```

This function returns an implementation-dependent approximation to the exponential function of the argument (e raised to the power of the argument, where e is the base of the natural logarithms). The input value x can be any numeric value or expression greater than 0.

Example:

```
Math.exp(10) = 22026.465794806718
```

### floor(x)

```
Number floor(x)
```

This function returns the greatest (closest to +infinity) number value that is not greater than the argument and is equal to a mathematical integer. If the argument is already an integer, the result is the argument itself. The input value x can be any numeric value or expression.

Example:

```
Math.floor(654.895869)=654
```

### log(x)

```
Number log(x)
```

This function returns an implementation-dependent approximation to natural logarithm of the argument. The input value x can be any numeric value or expression greater than 0.

Example:

```
Math.log(2) = 0.6931471805599453
```

### max(x,y)

```
Number max(x,y)
```

This function returns the larger of the two arguments. The input values x and y can be any numeric values or expressions.

Example:

```
Math.max(12.345,12.3456)= 12.3456
```

### min(x,y)

```
Number min(x,y)
```

This function returns the smaller of the two arguments. The input values x and y can be any numeric values or expressions.

Example:

```
Math.min(-12.457,-12.567)= -12.567
```

### pow(x,y)

```
Number pow(x,y)
```

This function returns an implementation-dependent approximation to the result of raising x to the power of y. The input value x must be the number raised to a power. The input value y must be the power to which x is raised.

Example:

```
Math.pow(2,4) = 16
```

#### random()

```
Number random()
```

This method takes no arguments and returns a pseudo-random number between 0 and 1. The number value has approximately uniform distribution over that range, using an implementation-dependent algorithm or strategy. This function takes no arguments.

Example:

```
Math.random()=0.9545176397178535
```

#### round(x)

```
Number round(x)
```

This function returns the number value that is closest to the argument and is equal to a mathematical integer. If two integer number values are equally close to the argument, then the result is the number value that is closer to +infinity. If the argument is already an integer, the result is the argument itself. The input value x can be any number.

Example:

```
Math.round(13.53) = 14
```

#### sin(x)

```
Number sin(x)
```

This function returns an implementation-dependent approximation to the sine of the argument. The argument is expressed in radians. The input value x must be an angle measured in radians.

#### sqrt(x)

```
Number sqrt(x)
```

This function returns an implementation-dependent approximation to the square root of the argument. The input value x must be any numeric value or expression greater than or equal to 0. If the input value x is less than zero, the string “NaN” is returned. (NaN stands for “Not a Number”.)

Example:

```
Math.sqrt(25) = 5
```

#### tan(x)

```
Number tan(x)
```

This function returns an implementation-dependent approximation to the tangent of the argument. The argument is expressed in radians. The input value x must be an angle measured in radians.

### 9.3.7 Number Object

Lets you work with numeric values. The Number object is an object wrapper for primitive numeric values.

### MAX_VALUE

The largest positive finite value of the number type (approximately 1.7976931348623157e308).

Example:

```
Number.MAX_VALUE
```

### MIN_VALUE

The smallest positive nonzero value of the number type, (approximately 5e-324).

Example:

```
Number.MIN_VALUE
```

### NaN

The primitive value NaN represents the set of IEEE Standard “Not-a-Number” values.

Example:

```
Number.NaN
```

### NEGATIVE_INFINITY

The value of negative infinity.

Example:

```
Number.NEGATIVE_INFINITY
```

### Number()

```
Number()
```

The constructor of Number has two forms: Number(value) and Number().

### POSITIVE_INFINITY

The value of positive infinity.

Example:

```
Number.POSITIVE_INFINITY
```

### toString(radix)

```
toString()
```

If the radix is the number 10 or is not supplied, then this number value is given as an argument to the ToString operator; the resulting string value is returned. If the radix is supplied and is an integer from 2 to 36, but not 10, the result is a string, the choice of which is implementation-dependent. The toString function is not generic; it generates a runtime error if its this value is not a Number object. Therefore, it cannot be transferred to other kinds of objects for use as a method.

#### valueOf()

```
valueOf()
```

Returns this number value. The valueOf function is not generic; it generates a runtime error if its value is not a Number object. Therefore, it cannot be transferred to other kinds of objects for use as a method.

## 9.3.8 Object

Used to work with objects. Object is the primitive JavaScript object type. All ECMAScript objects are descended from object. That is, all ECMAScript objects have the methods defined for object.

#### Object( )

Constructor for object.

#### toString()

```
Object toString()
```

When the toString method is called on an arbitrary object, the following steps are taken:

1. Get the [[Class]] property of this object.
2. Compute a string value by concatenating the three strings “[object “, Result(1), and “]”.
3. Return Result(2).

#### valueOf()

```
Object valueOf()
```

The valueOf method for an object usually returns the object; however, if the object is a “wrapper”

for a host object, as might be created by the Object constructor, the contained host object should be returned.

## 9.3.9 String Object

Used to work with String Objects.

#### String(x)

```
String(x)
```

The constructor of the string.

#### charAt(pos)

```
charAt(pos)
```

Returns a string containing the character at position pos in the string resulting from converting this object to a string. If there is no character at that position, the result is the empty string. The result is a string value, not a string object.

### charCodeAt(pos)

```
charCodeAt(pos)
```

Returns a number (a nonnegative integer less than 2^16) representing the Unicode* code point encoding of the character at position pos in the string resulting from converting this object to a string. If there is no character at that position, the result is NaN.

### fromCharCode(char0, char1, . . .)

```
fromCharCode(char0, char1, . . .)
```

Returns a string value containing as many characters as the number of arguments. Each argument specifies one character of the resulting string, with the first argument specifying the first character, and so on, from left to right. An argument is converted to a character by applying the operation ToUint16 and regarding the resulting 16-bit integer as the Unicode code point encoding of a character. If no arguments are supplied, the result is the empty string.

### indexOf(searchString, pos)

```
indexOf(searchString, pos)
```

If the given searchString appears as a substring of the result of converting this object to a string, at one or more positions that are at or to the right of the specified position, then the index of the leftmost such position is returned; otherwise, -1 is returned. If position is undefined or not supplied, 0 is assumed, in order to search all of the string.

### lastIndexOf(searchString, pos)

```
lastIndexOf(searchString, pos)
```

If the given searchString appears as a substring of the result of converting this object to a string, at one or more positions that are at or to the left of the specified position, then the index of the rightmost such position is returned; otherwise, -1 is returned. If position is undefined or not supplied, the length of the string value is assumed, in order to search all of the string.

### length

Returns the length of the String.

### match(RegExp)

```
String match(RegExp)
```

Takes a regular expression object as argument. It returns an array of matches; otherwise, returns null.

### replace(RegExp, String)

```
String replace(RegExp, String)
```

Takes a regular expression and a replacement string. It returns the original string with replacements accomplished.

### search(RegExp)

```
String search(RegExp)
```

Takes a regular expression as the sole arg and returns the offset of the first substring that matches, or -1 on no match.

### split(separator)

```
split(separator)
```

Returns an Array object, into which substrings of the result of converting this object to a string have been stored. The substrings are determined by searching from left to right for occurrences of the given separator; these occurrences are not part of any substring in the returned array, but serve to divide the string value. The separator may be a string of any length.

### substring(start, end)

```
substring(start, end)
```

Returns a substring of the result of converting this object to a string, starting from character position start and running to the position end of the string. If the second parameter is not present, the end position is considered the end of the string. The result is a string value, not a string object.

### toLowerCase()

```
toLowerCase()
```

Returns a string equal in length to the length of the result of converting this object to a string. The result is a string value, not a string object. Every character of the result is equal to the corresponding character of the string, unless that character has a Unicode 2.0 lowercase equivalent, in which case the lowercase equivalent is used instead. The canonical Unicode 2.0 case mapping must be used, which does not depend on implementation or locale.

### toString()

```
toString()
```

Returns this string value. When concerned with the placement and use of whitespace line terminators and semicolons within the representation, the string value is implementation-dependent.

### toUpperCase()

```
toUpperCase()
```

Returns a string equal in length to the length of the result of converting this object to a string. The result is a string value, not a string object. Every character of the result is equal to the corresponding character of the string, unless that character has a Unicode 2.0 uppercase equivalent, in which case the uppercase equivalent is used instead. The canonical Unicode 2.0 case mapping must be used, which does not depend on implementation or locale.

### valueOf()

```
valueOf()
```

Returns this string value. The valueOf() function is not generic, so it generates a runtime error if the object is not a String object.

## 9.3.10 Global Functions

Global functions in ECMAScript are functions that are independent of any particular object. The Expression Builder supports the following Global functions:

### getEnvironmentCountry

```
getEnvironmentCountry()
```

Function returns a 2-character string (for example, US) that represents the location that is currently selected on the user's computer.

### getEnvironmentLanguage

```
getEnvironmentLanguage()
```

Function returns a 2-character string (for example, EN) that represents the input language that is currently selected on the user's computer.