# NetIQ® Identity Manager

## Administrator's Guide to Designing the Identity Applications

**February 2017**

## Legal Notice

# Contents

**About this Book and the Library** **11**

**About NetIQ Corporation** **13**

**1 Introduction to the Identity Applications Design Tools** **15**

About the Provisioning View . . . 15
About the Directory Abstraction Layer Editor . . . 15
About the Provisioning Request Definition Editor . . . 16
About the ECMA Expression Builder . . . 16
About the Provisioning Team Editor . . . 16
About the Role Catalog . . . 16
Documenting a Project . . . 17
- Provisioning Locales . . . 17
- Directory Abstraction Layer . . . 17
- Provisioning Request Definitions . . . 17
- Provisioning Teams . . . 17
- Role Catalog . . . 18

**2 Working with the Provisioning View** **19**

Setting Up a Provisioning Project . . . 19
- Adding a User Application Driver to the Project . . . 19
- Adding a Role Service Driver to the Project . . . 20
- Modifying the Role Service Driver Properties . . . 21
- About Email Notification Templates . . . 22
- Email Based Approval . . . 22

Accessing the Provisioning View . . . 26
Setting Provisioning View Preferences . . . 27
Importing Provisioning Objects . . . 27
- Importing from a Driver Configuration File . . . 27
- Importing from an Identity Vault . . . 27

Exporting Provisioning Objects . . . 28
Validating Provisioning Objects . . . 28
- Directory Abstraction Layer Objects . . . 29
- Provisioning Request Definitions . . . 29
- Provisioning Teams . . . 29
- Role Configuration Objects . . . 29
- Roles . . . 30
- Resources . . . 30
- User Application Driver Locales . . . 30

Deploying Provisioning Objects . . . 31
- Deploying Provisioning Objects . . . 31
- Testing the Deployed Changes . . . 32

Comparing Provisioning Objects . . . 33
Specifying Locales and Localization Resource Groups . . . 33
- Specifying the Default Locale . . . 34
- Defining the User Application's Supported Locales . . . 34
- Creating a Custom Localization Resource Group . . . 37

Localizing Provisioning Objects . . . 39
- Using Designer to Localize . . . 40
- Supported Languages . . . 41

Exporting and Importing Data to Localize . . . . . . . . . . 41

## 3 Configuring the Directory Abstraction Layer 45

About the Directory Abstraction Layer . . . . . . . . . . 45
Analyzing the User Application’s Data Needs . . . . . . . . . . 45
About the Directory Abstraction Layer Editor . . . . . . . . . . 46
About Directory Abstraction Layer Editor Files . . . . . . . . . . 47
Working with Entities and Attributes . . . . . . . . . . 48
About Entities and Attributes . . . . . . . . . . 48
Adding Entities . . . . . . . . . . 49
Adding Attributes . . . . . . . . . . 53
Updating the Schema Elements List . . . . . . . . . . 54
Working with Lists . . . . . . . . . . 54
Working with Queries . . . . . . . . . . 56
Working with Relationships . . . . . . . . . . 58
Working with Configuration Settings . . . . . . . . . . 59
Directory Abstraction Layer Property Reference . . . . . . . . . . 59
Entity Properties . . . . . . . . . . 60
Attribute Properties . . . . . . . . . . 63
Queries Properties . . . . . . . . . . 72
Relationship Properties . . . . . . . . . . 73

## 4 Configuring Provisioning Request Definitions 77

About Provisioning Request Definitions . . . . . . . . . . 77
Using the Provisioning Request Definition Editor . . . . . . . . . . 78
Creating a Provisioning Request Definition . . . . . . . . . . 78
Starting the Provisioning Request Definition Editor . . . . . . . . . . 79
Creating a Provisioning Request Definition By Using a Template . . . . . . . . . . 80
Creating a Custom Provisioning Request Definition . . . . . . . . . . 84
Creating a Roles Based Provisioning Request Definition . . . . . . . . . . 84
Modifying Settings of a Provisioning Request Definition . . . . . . . . . . 86
Provisioning and Workflow Example . . . . . . . . . . 90
Step 1: Initiating the Request . . . . . . . . . . 90
Step 2: Approving the Request . . . . . . . . . . 90
Step 3: Fulfilling the Request . . . . . . . . . . 94
Step 4: Completing the Workflow . . . . . . . . . . 94

## 5 Creating Forms for a Provisioning Request Definition 95

About Forms . . . . . . . . . . 95
About Form Control Data Binding . . . . . . . . . . 95
About Forms and Events . . . . . . . . . . 96
About the Forms Tab . . . . . . . . . . 96
About Form Selection . . . . . . . . . . 96
About Form Controls . . . . . . . . . . 97
Creating Forms . . . . . . . . . . 97
Creating New Forms . . . . . . . . . . 97
Adding Form Controls and Actions . . . . . . . . . . 97
Defining Events . . . . . . . . . . 99
Using the Scripts Tab . . . . . . . . . . 100
Action Reference . . . . . . . . . . 102
Form Control Reference . . . . . . . . . . 103
Data Type for Roles Based Request Forms . . . . . . . . . . 104
Data Type for Resource Based Request Forms . . . . . . . . . . 104
Controls for User-Entered Comments . . . . . . . . . . 105

General Form Control Properties . . . 105
CheckBoxPickList . . . 106
DatePicker . . . 107
DateTimePicker . . . 109
DNContainer . . . 109
DNDisplay . . . 111
DNLookup . . . 113
DNMaker . . . 115
DNQuery . . . 117
Global List . . . 119
Localized Label . . . 120
Html . . . 121
MVCheckbox . . . 122
MVEditor . . . 124
Password . . . 129
PickList . . . 129
RadioButtons . . . 132
Static List . . . 134
Text . . . 136
Text Area . . . 137
Title . . . 138
TrueFalseCheckBox . . . 138
TrueFalseRadioButtons . . . 139
TrueFalseSelectBox . . . 140
Working with Distinguished Names . . . 140
Formatting DNs . . . 140
Working with Object Selectors . . . 141
Using DAL Queries in Forms . . . 143
Printing Forms . . . 148

**6 Creating the Workflow for a Provisioning Request Definition 149**

About the Workflow Tab . . . 149
Canvas . . . 149
Palette . . . 151
Views . . . 152
Adding Activities to a Workflow . . . 153
Setting the General Properties of an Activity . . . 154
Defining the Data Item Mappings . . . 156
Defining the E-Mail Notification Settings . . . 157
Adding Flow Paths . . . 158
Configuring Flow Paths . . . 158
Guidelines for Creating Workflows . . . 160
Rules for Activities . . . 160
Rules for Flow Paths . . . 161
Understanding Workflow Data . . . 163
Guidelines for Creating Roles Based Workflows . . . 168
About Role Approval Workflows . . . 168
Writing Custom Role Workflows . . . 170
About Separation of Duties Approval Workflows . . . 174
Customizing the Standard Separation of Duties Workflow . . . 177
Guidelines for Creating Resource Based Workflows . . . 177
About Resource Approval Workflows . . . 178
Writing Custom Resource Workflows . . . 180
Debugging a Workflow . . . 183
Using the Log Activity . . . 183
Using the Workflow Database . . . 183
Changing Log Levels . . . 183

Provisioning Multiple Individuals with One Workflow Instance . . . . . 184
Basic Steps for Using the Workflow . . . . . 184
Making Distinguished Name References Portable . . . . . 184

7 Workflow Activity Reference 187

Start Activity . . . . . 187
Properties . . . . . 187
Data Item Mapping . . . . . 188
Email Notification . . . . . 189
Approval Activity . . . . . 189
Properties . . . . . 189
Data Item Mapping . . . . . 194
Available ECMAScript Methods . . . . . 195
Email Notification . . . . . 196
Addressing an Approval Activity . . . . . 198
Log Activity . . . . . 204
Properties . . . . . 204
Data Item Mapping . . . . . 204
Email Notification . . . . . 205
Branch Activity . . . . . 205
Properties . . . . . 205
Data Item Mapping . . . . . 205
Email Notification . . . . . 205
Merge Activity . . . . . 205
Properties . . . . . 206
Data Item Mapping . . . . . 206
Email Notification . . . . . 206
Condition Activity . . . . . 206
Properties . . . . . 206
Data Item Mapping . . . . . 207
Email Notification . . . . . 207
Mapping Activity . . . . . 207
Properties . . . . . 208
Data Item Mapping . . . . . 208
Email Notification . . . . . 208
Workflow Status . . . . . 208
Properties . . . . . 209
Data Item Mapping . . . . . 209
Email Notification . . . . . 209
Email Activity . . . . . 209
Properties . . . . . 209
Data Item Mapping . . . . . 209
Email Notification . . . . . 210
Role Request Binding Activity . . . . . 211
Properties . . . . . 211
Data Item Mapping . . . . . 211
Email Notification . . . . . 211
Role Request Activity . . . . . 212
Properties . . . . . 212
Data Item Mapping . . . . . 216
Email Notification . . . . . 216
Resource Request Binding Activity . . . . . 216
Properties . . . . . 217
Data Item Mapping . . . . . 217
Email Notification . . . . . 217
Resource Request Activity . . . . . 217
Properties . . . . . 217

Data Item Mapping . . . 219
Email Notification . . . 219
Start Workflow Activity . . . 219
Properties . . . 219
Data Item Mapping . . . 220
Email Notification . . . 220
Finish Activity . . . 220
Properties . . . 221
Data Item Mapping . . . 221
Email Notification . . . 221
Rest Activity . . . 223
Properties . . . 223
Data Item Mapping . . . 225
Email Notification . . . 226
Integration Activity . . . 227
Properties . . . 227
Data Item Mapping . . . 228
Email Notification . . . 229
Entitlement Activity . . . 229
Properties . . . 229
Data Item Mapping . . . 230
Email Notification . . . 230
Entity Activity . . . 230
Properties . . . 231
Data Item Mapping . . . 232
Email Notification . . . 233
Working with Entity Activities . . . 233

**8 Working with Integration Activities 235**

About the Integration Activity . . . 235
Adding an Integration Activity . . . 235
Moving Data to and from the Integration Activity . . . 237
Using the Integration Activity Editor Interface . . . 238
XML Views . . . 239
Action Model . . . 244
WSDL Editor . . . 250
Messages . . . 250
Regenerating Code for the Action Model . . . 251
Adding Actions to the Action Model . . . 251
Actions . . . 251
Advanced . . . 252
Data Exchange . . . 256
Repeat . . . 259
Comment . . . 263
Decision . . . 264
Function . . . 264
Log . . . 265
Map . . . 266

**9 Working with ECMA Expressions 271**

About the ECMA Expression Builder . . . 271
About ECMAScript . . . 271
ECMAScript Capabilities . . . 272
Using the ECMA Expression Builder . . . 272
About Java Integration . . . 274
About XPath Integration . . . 275

About Global Configuration Values Integration . . . 277
About Global ECMAScripts Integration . . . 277
Performance Considerations . . . 277
ECMAScript Examples . . . 278
General Examples . . . 278
Flowdata Examples . . . 278
Form Control Examples . . . 279
Error Handling . . . 280
User Application API . . . 280
Form Action Script Methods . . . 280
IDVault Functions . . . 288
nrfRequest Properties and Methods . . . 298
Role Vault API . . . 302
About the Role Vault API . . . 303
Role Script API Reference . . . 306
Role Vault Bean API Reference . . . 312

10 Configuring Provisioning Teams 321

About Teams . . . 321
About Team Requests . . . 322
Using a Team to Manage Direct Reports . . . 323
Managing Teams . . . 323
Creating a Team . . . 323
Deleting a Provisioning Team . . . 324
Creating a Team to Manage Direct Reports . . . 324

11 Configuring Roles 325

About the Roles Based Provisioning Module . . . 325
About the Role Catalog . . . 325
About the Role Editor . . . 326
Understanding Role Hierarchy . . . 326
Using the Role Editor . . . 327
Role Properties Reference . . . 333
About the Separation of Duties Editor . . . 335
Using the Separation of Duties Editor . . . 335
Separation of Duties Constraints Properties . . . 337
About the Role Configuration Editor . . . 339
Role Configuration Editor Properties . . . 340
Importing Roles Defined in CSV Files . . . 343
Setting Up the File to Import . . . 343
Required CSV File Format . . . 346
Using the Wizard to Import Roles . . . 346
Error Handling . . . 348

12 Configuring Resources 349

About Resources . . . 349
About the Resource Editor . . . 349
About Resource Containers . . . 349
Using the Resource Editor . . . 349
Resource Property Reference . . . 352
Importing Resources Defined in CSV Files . . . 353
Setting Up the File to Import . . . 353
Required CSV File Format . . . 356
Using the Wizard to Import Roles from a CSV File . . . 356
Error Handling . . . 357

## A ECMAScript Core Reference 359

ECMAScript Operators . . . . . 359

Functions/Methods . . . . . 361

DOM Methods . . . . . 362

- Node . . . . . 362
- Document . . . . . 366
- Element . . . . . 371
- Attribute . . . . . 376
- CharacterData . . . . . 377
- NodeList . . . . . 378
- NamedNodeMap . . . . . 380
- Text . . . . . 382
- DocumentType . . . . . 382
- DOMImplementation . . . . . 383
- Notation . . . . . 384
- Entity . . . . . 384
- ProcessingInstruction . . . . . 384

ECMAScript Core . . . . . 385

- Array Object . . . . . 385
- Boolean Object . . . . . 386
- Date Object . . . . . 386
- Function Object . . . . . 391
- Global . . . . . 392
- Math Object . . . . . 393
- Number Object . . . . . 398
- Object . . . . . 399
- String Object . . . . . 399

# About this Book and the Library

The *User Application: Design 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, the provisioning request definition editor, the provisioning team editor, and the role catalog.

## Intended Audience

This book provides information for individuals responsible for understanding administration concepts and implementing a secure, distributed administration model.

## Other Information in the Library

The library provides the following information resources:

**Identity Manager Setup Guide**

Provides overview of Identity Manager and its components. This book also provides detailed planning and installation information for Identity Manager.

**Designer Administration Guide**

Provides information about designing, testing, documenting, and deploying Identity Manager solutions in a highly productive environment.

**User Application: Administration Guide**

Describes how to administer the Identity Manager User Application.

**User Application: User Guide**

Describes the user interface of the Identity Manager User Application and how you can use the features it offers, including identity self-service, the Work Dashboard, role and resource management, and compliance management.

**Identity Reporting Module Guide**

Describes the Identity Reporting Module for Identity Manager and how you can use the features it offers, including the Reporting Module user interface and custom report definitions, as well as providing installation instructions.

**Analyzer Administration Guide**

Describes how to administer Analyzer for Identity Manager.

**Identity Manager Common Driver Administration Guide**

Provides information about administration tasks that are common to all Identity Manager drivers.

**Identity Manager Driver Guides**

Provides implementation information about Identity Manager drivers.

# About NetIQ Corporation

We are a global, enterprise software company, with a focus on the three persistent challenges in your environment: Change, complexity and risk—and how we can help you control them.

## Our Viewpoint

**Adapting to change and managing complexity and risk are nothing new**

In fact, of all the challenges you face, these are perhaps the most prominent variables that deny you the control you need to securely measure, monitor, and manage your physical, virtual, and cloud computing environments.

**Enabling critical business services, better and faster**

We believe that providing as much control as possible to IT organizations is the only way to enable timelier and cost effective delivery of services. Persistent pressures like change and complexity will only continue to increase as organizations continue to change and the technologies needed to manage them become inherently more complex.

## Our Philosophy

**Selling intelligent solutions, not just software**

In order to provide reliable control, we first make sure we understand the real-world scenarios in which IT organizations like yours operate — day in and day out. That's the only way we can develop practical, intelligent IT solutions that successfully yield proven, measurable results. And that's so much more rewarding than simply selling software.

**Driving your success is our passion**

We place your success at the heart of how we do business. From product inception to deployment, we understand that you need IT solutions that work well and integrate seamlessly with your existing investments; you need ongoing support and training post-deployment; and you need someone that is truly easy to work with — for a change. Ultimately, when you succeed, we all succeed.

## Our Solutions

- Identity & Access Governance
- Access Management
- Security Management
- Systems & Application Management
- Workload Management
- Service Management

# Contacting Sales Support

For questions about products, pricing, and capabilities, contact your local partner. If you cannot contact your partner, contact our Sales Support team.

| | |
|---|---|
| **Worldwide:** | www.netiq.com/about_netiq/officelocations.asp |
| **United States and Canada:** | 1-888-323-6768 |
| **Email:** | info@netiq.com |
| **Web Site:** | www.netiq.com |

# Contacting Technical Support

For specific product issues, contact our Technical Support team.

| | |
|---|---|
| **Worldwide:** | www.netiq.com/support/contactinfo.asp |
| **North and South America:** | 1-713-418-5555 |
| **Europe, Middle East, and Africa:** | +353 (0) 91-782 677 |
| **Email:** | support@netiq.com |
| **Web Site:** | www.netiq.com/support |

# Contacting Documentation Support

Our goal is to provide documentation that meets your needs. The documentation for this product is available on the NetIQ Web site in HTML and PDF formats on a page that does not require you to log in. If you have suggestions for documentation improvements, click **comment on this topic** at the bottom of any page in the HTML version of the documentation posted at www.netiq.com/documentation. You can also email Documentation-Feedback@netiq.com. We value your input and look forward to hearing from you.

# Contacting the Online User Community

NetIQ Communities, the NetIQ online community, is a collaborative network connecting you to your peers and NetIQ experts. By providing more immediate information, useful links to helpful resources, and access to NetIQ experts, NetIQ Communities helps ensure you are mastering the knowledge you need to realize the full potential of IT investments upon which you rely. For more information, visit http://community.netiq.com.

# 1 Introduction to the Identity Applications Design Tools

This section provides an overview of the tools available for designing and configuring the identity applications, particularly the User Application.

**IMPORTANT:** The identity applications, including the User Application, are a set of applications and not a framework. The areas within the identity applications that you should modify are outlined within the product documentation. Modifications to areas not outlined within the product documentation are not supported.

## About the Provisioning View

In the User Application, the Provisioning view provides persistent access to Designer's provisioning, roles, and compliance features. Use the Provisioning view to perform the following actions on provisioning and roles objects:

- Access the editors that allow you to create and manipulate User Application components, such as:
  - The directory abstraction layer editor
  - The provisioning request definitions editor
  - The teams editor
  - The role catalog

  Double-clicking an item from the Provisioning view opens the editor for that item.
- Manipulate object definitions, such as:
  - Importing and exporting object definitions from the Identity Vault or the local file system
  - Validating local object definitions
  - Deploying object definitions to the Identity Vault
  - Comparing the objects on the local file system with those in the Identity Vault
- Define the User Application driver's supported and default locales, including:
  - Importing and exporting display labels and other User Application strings for localization
  - Defining custom localization resource groups (used only for field localization)

## 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 entities.
- Defining LDAP searches as Queries that you can run from request and approval forms.

## 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.

## 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 email addresses and content.
  - Set entitlement grant/revoke rights and parameters.
  - Evaluate any past activity data to conditionally follow a workflow path by using the Condition Activity.
  - Write different log messages that are conditionally triggered by using a single Log Activity.

## About the Provisioning Team Editor

The provisioning team editor allows you to define a set of users who can act as a team within the User Application. The team definition determines who can manage provisioning requests and approval tasks associated with this team. The team definition consists of a list of team managers, team members, and team options. In addition, you can define the set of provisioning request objects that the team can act on.

## About the Role Catalog

The Role Catalog includes tools that let you define the contents of the **Roles** tab of the User Application. The tools available through the Role Catalog include:

- **Resource editor:** Defines the set of available resources. Includes information about the resource’s trustees, owners, approval workflow, and entitlements.

- **Role editor:** Defines the set of available roles. Includes information about the role's trustees, owners, role containment hierarchy, and entitlements.
- **Separation of Duties editor:** Defines the separation of duties constraints and how to handle requests for exceptions to those constraints.
- **Role Configuration editor:** Lets you modify the roles subsystem administrative settings.

The Role Catalog also includes a menu option that enables you to import roles defined in a comma-separated values (CSV) file.

# 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.

## Provisioning Locales

Lists the supported locales and default locales along with the provisioning resource groups.

## Directory Abstraction Layer

Includes the following sections:

- **Entities:** Including access properties, auxiliary classes, and LDAP classes.
- **Lists:** Including key and display label.
- **Queries:** Including the query's keys, parameters, and conditions.
- **Relationships:** Including key, parent key, parent attribute, child key, and child attribute.
- **Configuration:** Including default entity key, default locale, and container classes.

## Provisioning Request Definitions

Includes:

- A table containing the definition's category, status, and email 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.

## Provisioning Teams

Includes:

- Display name
- The team members
- The request type and scope
- The manager's permissions

## Role Catalog

Includes the following section:

- **Roles:** Including display name, description, role level, categories, and approval details.
- **Separation of Duties Constraints:** Including display name, description, conflicting roles, approval type, and approvers.
- **Role Configuration:** Including role removal grace period, role level display names and descriptions, approval types, and approval definitions.

# 2 Working with the Provisioning View

To perform many of the operations available from the Provisioning view (such as compare, import, and deploy along with the wizards and editors), Designer must be able to establish a connection to the Identity Vault. Designer generates error messages when it cannot connect to the Identity Vault while performing these actions. To ensure that Designer is always able to connect to the Identity Vault, you can choose to save the password when you configure the Identity Vault credentials for your project. When you choose **Save password**, Designer saves the password to the local file system; it is not secure.

## Setting Up a Provisioning Project

The Provisioning view is only available for Designer projects that contain a User Application driver. After you set up an Identity Manager project and configure an Identity Vault and driver set for the project, you add and configure a User Application driver.

To use Designer to configure the **Roles** tab of the User Application, you must additionally add a Role Service driver to your project.

### Adding a User Application Driver to the Project

To add the User Application driver to a project, the driver must be installed and deployed in your environment. For more information, see Creating and Deploying the Drivers for the Identity Applications in the *NetIQ Identity Manager Setup Guide*.

1 In an open Designer project, create a new driver by using one of these methods:

- Click **Provisioning** in the **Palette**, then drag the **User Application** icon onto the modeler.
- Right-click the driver set for your project, then select **New > Driver**.
- Click the driver set for your project, then select **Model > Driver > New**.

2 Select **User Application Base** from the list of driver base packages in the Driver Configuration Wizard, then click **Next**.

3 Specify the properties you want to use for the driver:

**Driver Name**

Specifies the either the User Application driver created when you installed Identity Manager or a new User Application driver.

**Authentication ID**

Specifies the DN of the User Application Administrator.

**Application Password**

Specifies the password for the User Application Administrator.

**Host**

Specifies the name or IP address of the application server where you deployed the User Application.

Identity Manager uses this information to perform the following actions:

- Trigger workflows on the application server to connect to access workflows, such terminate and retract
- Update cached data definitions

**Port**

Specifies the port for the host server.

**Application Context**

Specifies context of the User Application. For example, IDMProv.

**Allow Initiator Override**

*Applies to workflows that start automatically*

Specifies whether you want to use an identity other than the User Application Administrator to start workflows. To use an alternate identity, select **Yes**.

For more information, see the *Identity Manager User Application: Administration Guide.*

4 Click **Next**, then **Finish**.

5 (Optional) Deploy the email templates to support email notifications in your workflows.

When you add a User Application driver, Identity Manager adds email templates to the Default Notification Collection. You must explicitly deploy the templates. They are not deployed by default when you deploy the driver. For more information about deploying the templates, see Setting Up E-Mail Notification Templates in the *NetIQ Analyzer for Identity Manager Administration Guide*.

## Adding a Role Service Driver to the Project

1 In the same project where you created a User Application driver, click **Provisioning** in the **Palette**, then drag and drop the **Role Service** icon onto the Modeler.

2 Select **Role and Resource Service Base** from the list of driver base packages in the Driver Configuration Wizard, then click **Next**.

3 Specify the name you want to use for the driver and click **Next**.

4 Specify the properties you want to use for connecting the driver to the User Application. If you have already configured the User Application driver, the Driver Configuration Wizard should prepopulate the fields with the correct information, but we recommend you double-check the specified properties. Use the following information to configure the driver:

| Field | Description |
|---|---|
| User-Group base container DN | Specify the DN of the root container that the Role Service driver services. |
| User Application Driver DN | Specify the DN of the User Application Driver object that hosts the role system. For example, `system\driverset1\UserApplication`. |
| User Application URL | Specify the URL used to connect to the User Application. The default URL is `http://127.0.0.1:8180/IDMProv`. |
| User Application Identity | Specify the DN of the User Application Administrator. For example, `cn=admin,ou=sa,o=data`. |
| User Application Password | Specify the Application Password you specified for the User Application driver. |

5 Click **Next**.

6 Click **Finish**.

## Modifying the Role Service Driver Properties

After creating the Role Service driver, you can optionally modify some of the driver configuration settings and modify the additional settings described in Table 2-1. To customize the additional settings:

1 In the Modeler, right-click the Role Service driver and select **Driver** > **Properties**.

2 Select **Driver Configuration** (in the left pane).

3 Click the **Driver Parameters** tab.

4 Click the **Driver Options** tab. You can modify the driver's properties that you specified when you created the driver as well as the properties described in Table 2-1.

5 Click **OK** to save the changes.

*Table 2-1 Additional Settings for Customizing the Role Service Driver*

| Field | Description |
|---|---|
| Number of days before processing removed request objects | The number of days the driver should wait before cleaning up request objects that have finished processing. This value determines how long you are able to track the status of requests that have been fulfilled. |
| Frequency of reevaluation of dynamic and nested groups (in minutes) | The number of minutes the driver should wait before reevaluating dynamic and nested groups. This value determines the timeliness of updates to dynamic and nested groups used by the User Application. In addition, this value can have an impact on performance. Therefore, before specifying a value for this option, you need to weigh the performance cost against the benefit of having up-to-date information in the User Application. |
| Generate audit events | Determines whether audit events are generated by the driver. |

## About Email Notification Templates

Identity Manager includes a standard set of email notification templates.

When you create a User Application driver, any email notification templates that are missing from the standard set are replaced. However, existing email notification templates, which might come from an earlier version of Identity Manager, are not updated. To replace existing templates with new templates:

1. Expand the Outline view.
2. In the Default Notification Collection, delete the email notification templates that you want to replace.
3. Right-click **Default Notification Collection** and select **Add Default Templates** or **Add All Templates**.

   You can also use this command at any time to update email notification templates without creating a new User Application driver.
4. To deploy the email notification templates to the Identity Vault, right-click **Default Notification Collection** and select **Live > Deploy**.

## Email Based Approval

You can allow request reviewers to approve or deny a request using e-mail. The e-mail notification can include links that can be used for approving or rejecting requests to help you easily respond to the request without logging in to the identity applications.

**IMPORTANT:** If request reviewer forwards the mail to another user, action links for approval or rejection action do not work.

An request reviewer can respond to the notification email in the following two ways:

- When the request reviewer clicks **Approve** or **Deny** link in the e-mail, the identity applications composes a new email with the required subject and content. The request reviewer can enter comments in the e-mail send it after adding the comments.
- • Before replying to a notification, a request reviewer can modify the subject line of the e-mail. However, ensure that the keyword (**Approve** or **Deny**) along with the alpha-numeric unique code is not changed.

> **IMPORTANT:** To reply to the notification e-mail, set the ‘from’ address of request reviewer mail server as same as the incoming mail server.

## Checklist to Setup Email Based Approval

Before setting up the Email Based Approval, review the following checklist:

| | **Checklist Items** |
|---|---|
| □ | 1. Ensure that the outgoing mail server is configured using Configuration Update Utility or iManager and verify whether outgoing mail is working. To configure the outgoing mail server using Configuration Update Utility, see Email Server Configuration in the NetIQ Identity Manager Setup Guide. If you are using iManager, see E-Mail Notification in the NetIQ iManager Administration Guide. |
| □ | 2. Install Email Based Approval package in Designer. For more information, see “Installing Email Based Approval Package” on page 23. |
| □ | 3. Choose the Email Based Approval Templates for workflows. For more information, see “Using Email Based Approval Templates for Workflow” on page 24. |
| □ | 4. 4. For receiving emails, set up an account on an email server that supports POP3 or IMAP protocol. If you are using a POP3 type of server, the received emails are not marked as ‘read’ in the incoming mailbox. For more information, see Help in the Identity Manager Dashboard. |
| □ | 5. Enable Email Based Approval and configure incoming Mail server properties from the User Application, see “Enabling Email Approval and Configuring Incoming Mail Server Properties from User Application” on page 24. |
| □ | 6. (Conditional) To enable Digital Signature to sign an outgoing email, see “Enabling Digital Signature to Sign an Outgoing Email” on page 24. |
| □ | 7. (Conditional) To enable Email Based Approval on Cluster Environment, see “Enabling Email Based Approval on Cluster Environment” on page 25. |

## Installing Email Based Approval Package

In Designer, open the project where you want to install Email Based Approval package and perform the following steps:

1. Right-click on the Identity Manager engine and select **Properties > Packages**.
2. Click + icon to add new package and select **Email Based Approval Templates** package.
3. Install the package and click **OK**.
4. Expand **Default Notification Collection** and select all the Email Based Approval templates.
5. Deploy the selected templates on to your Identity Vault.

## Using Email Based Approval Templates for Workflow

In Designer, select the required Email Based Approval templates to deploy the workflow:

1. Create a PRD workflow, select the **Notify participants by E-mail** check box.
2. Right-click on the Approver icon, and select **Show E-mail Notification**.
3. Select one of the following e-mail options as:
    - **Notify**,
    - **Reminder**
    - **Escalation Reminder**
4. From **E-mail Template** drop-down list, select **Email Based Approval Provisioning Notification** template.
5. Repeat Step 1 through Step 4 for the workflows that requires approval.

    **NOTE:** Email Based Approval allows request reviewers to process the requests which is not associated with complex forms.

6. Add the source ECMA script for the required field based on the selected template.
7. After creating a PRD, deploy the workflow.

## Enabling Email Approval and Configuring Incoming Mail Server Properties from User Application

To allow users to approve or reject requests from an email:

1. Log in to the Identity Manager Dashboard.
2. Select **Administration > Email-based approval**.

For more information, see the Help in the Dashboard.

## Enabling Digital Signature to Sign an Outgoing Email

To enable Digital Signature service, perform the following:

1. In **Identity Manager Dashboard**, navigate to the **Email based Approval** page.
2. In **Email Content Options** panel, select the **Include action links with digital signature** option.

In **Configuration Update Utility**, ensure that you have specified the Identity Vault certificates details and updated the **Email Server Configuration** fields to enable digital signature service on the outgoing e-mails. For more information about the User Application Parameters, see Email Server Configuration section in the NetIQ Identity Manager Setup Guide.

To setup a digital signature support, perform the following steps:

1. In Configuration Update Utility, specify the private keystore and private key certificate properties.

    **NOTE:** If private key certificate alias contains any special character, then rename the certificate alias.

2. Import the private key certificate by running the following command:

```
<JRE_BIN_PATH>/keytool -v -importkeystore -srckeystore
<privatekey_certificate_path>-srcstoretype PKCS12 -destkeystore
<keystore_path>  deststoretype JKS
```

3 (Conditional) Rename the imported certificate by running the following command:

```
<JRE_BIN_PATH>/keytool -changealias -keystore <privatekey_store_path> -alias
"<old_alias_name>" -destalias <new_alias_name>
```

## Enabling Email Based Approval on Cluster Environment

Ensure that the outgoing mail server is configured on each nodes of the cluster, you can configure this mail server using **Configuration update Utility**. For more information, see Email Server Configuration in the NetIQ Identity Manager Setup Guide.

On cluster environment, you must specify the IP address of the ActiveMQ server in `server.xml` file of the other nodes in the cluster, which is set to `localhost` by default. Look for the `brokerURL` attribute for the AcitveMQ server in the `server.xml` and replace the `localhost` with the ActiveMQ server IP address.

If you enable or disable the Email Based Approval (**Off/On**) or change the incoming mailbox properties, then restart all the cluster nodes for the change to take effect. You must also verify the connection between the mailbox host and other servers.

## Troubleshooting Tips

This section helps you to troubleshoot the general issues while configuring Email Based Approval.

- In provisioning request e-mail, if recipient’s mail address, or **Approval** or **Deny** Tokens are missing, enable the **Email Based Approval** option.
- If a request reviewer cannot find the **Approve** or **Deny** link in the e-mail, review the **Email Based Approval** settings.
- If a request reviewer cannot compose an e-mail after clicking the **Approve** or **Deny** link, verify whether the default e-mail client is configured. (For example: Microsoft Outlook)

You can verify the Email Based Approval settings using the `catlina.out` events logs.

For example, if you want to verify the status of Email Based Approval, look for the following entries in the `catlina.out` property file:

```
INFO  com.novell.soa.notification.impl.EmailReceiverEngine- [RBPM] Email based
approval feature is turned on. Starting the incoming mailbox service soon..
INFO  com.novell.soa.notification.impl.jms.JMSConnectionMediator- [RBPM] Starting
JMS notification system
INFO  com.novell.soa.notification.impl.EmailReceiverEngine- [RBPM] Successfully
started persistent JMS notification system for email based approval
INFO  com.novell.soa.notification.impl.EmailReceiverThread- [RBPM] Starting
asynchronous notification system
INFO  com.novell.soa.notification.impl.EmailReceiverEngine- [RBPM] Mailbox service
for incoming mail started successfully without any warning
INFO  com.novell.soa.notification.impl.EmailReceiverEngine- [RBPM] Email based
approval token cleanup service started successfully.
```

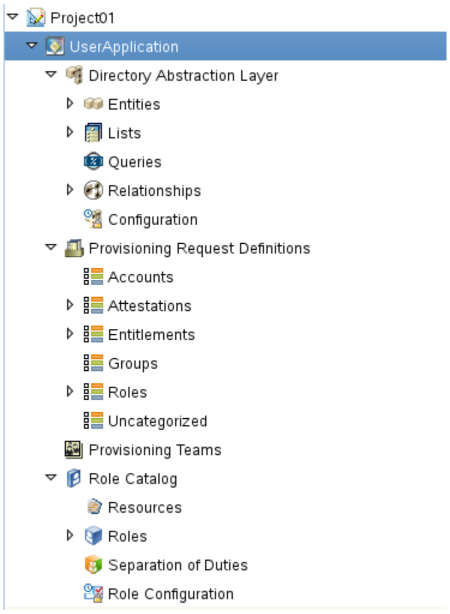
# Accessing the Provisioning View

You can access the Provisioning view in the following ways:

- Select **Window > Show View > Provisioning.**
- In the Modeler window, right-click the User Application, then select **Show Provisioning View**.
- In the Outline view, right-click the User Application, then select **Show Provisioning View**.

When it is open, the Provisioning view displays all of the provisioning projects located in the same workspace. The contents of the view depend on what version of the User Application driver youa selected when you created the project.

***Figure 2-1** Sample Provisioning View*



- Project01
  - UserApplication
    - Directory Abstraction Layer
      - Entities
      - Lists
      - Queries
      - Relationships
      - Configuration
    - Provisioning Request Definitions
      - Accounts
      - Attestations
      - Entitlements
      - Groups
      - Roles
      - Uncategorized
    - Provisioning Teams
    - Role Catalog
      - Resources
      - Roles
      - Separation of Duties
      - Role Configuration

The Provisioning view displays icons to indicate the object’s status. The icons are described in Table 2-2.

***Table 2-2** Provisioning View Status Icons*

| Icon | Description |
|---|---|
| ⟳ | Indicates that the local object has changed. |
| ⚠ | Indicates that the local object contains a validation warning. |
| ☒ | Indicates that the local object contains a validation error. |

The User Application driver icon includes a tooltip that provides the project's Identity Vault name, the DriverSet, the driver name, and the version.

**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.

# Setting Provisioning View Preferences

You can customize some Provisioning view behaviors by setting preferences. You access the preferences page through **Windows > Preferences > NetIQ > Provisioning**.

# Importing Provisioning Objects

The Provisioning view's import feature lets you import provisioning objects in different ways.

- "Importing from a Driver Configuration File" on page 27
- "Importing from an Identity Vault" on page 27

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 driver set'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.

## 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. Right-click the container 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**.

Trustee information is stored in the driver configuration file. When you import a driver configuration file using Designer, the trustee information is processed as expected. If you import the driver configuration file using iManager, the trustee information is ignored.

## 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. Right-click the container and select one of the following:

   - **Live > Import** to import the contents of the currently selected container.
   - **Live > Import Object** to browse the Identity Vault and select the object to import.
   - **Live > Import From** to browse the Identity Vault and select a container whose contents you want to import objects from.

   If prompted, provide the Identity Vault credentials and click **OK**.

   **NOTE:** For provisioning teams, **Import Object** imports only the team object. **Import Team Requests** imports any associated team request objects.

3. Navigate to the Identity Vault container or object that you want to import and click **OK.**
4. Review the Import Summary page to determine how you want to proceed. To complete the import, click **Import**, or click **Cancel**. If you click **Import**, Designer performs the operation and displays a summary of the completed operation.

# 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.

To export 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. Right-click the container or object 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.

# 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.

You can validate provisioning objects individually, by node (such as the directory abstraction layer, a provisioning team, or a separation of duty constraint), or at the User Application driver level. Each node (individual, container-level, or driver-level) has a right-click menu item called **Validate**. In addition, when you open an object in the editor, you can access the **Validate** option, for that item, from Designer's main menu and toolbar. For example, if you have a provisioning request definition open in the editor, the main menu and toolbar provides a **PRD > Validate** menu option.

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

Each object type has unique validation rules. They are described in each of the following sections:

- "Directory Abstraction Layer Objects" on page 29
- "Provisioning Request Definitions" on page 29
- "Provisioning Teams" on page 29
- "Role Configuration Objects" on page 29
- "Roles" on page 30
- "Resources" on page 30
- "User Application Driver Locales" on page 30

## 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 references exist.
- Ensures that every entity has at least one attribute defined.
- Ensures that every local and global list contains at least one item.

## 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 for 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.
- Ensures that static list keys contain the correct data for the decimal data type.

## Provisioning Teams

Designer does the following:

- Validates that managers and members have been defined for the team.
- Validates that team requests are specified for the team.
- If the request scope is Categories, it validates that the team request actually references a category.

## Role Configuration Objects

Designer does the following:

- Ensures that the **Quorum** value should be a number between 0 and 100. Validation rules take into consideration that a percentage can be entered.
- Ensures that the **Removal Grace Period** is a positive number.
- Ensures that Display Names and Descriptions use supported locales.

- Ensures that the Provisioning Request Definitions defined for the Role Approval and SoD Conflict Approvals are valid, are not templates, and whose process types match properly.
- Separation of Duties (SoD) approvers must exist and be valid.

## Roles

Before deployment, Designer validates that:

- The category exists.
- The description is provided for all supported languages.
- The Quorum is a valid expression.
- Approvers are present when the approval type is set to standard serial or parallel.

On deploy, Designer validates that the following objects exist in the Identity Vault:

- The entitlement
- The owner
- The Role Trustees
- The lower-level roles
- Groups
- Containers
- Approvers
- Provisioning request definition

## Resources

Before deployment, Designer validates that:

- The category exists.
- The description is provided for all supported languages.
- The Quorum is a valid expression.
- Approvers are present when the approval type is set to standard serial or parallel.

On deploy, Designer validates that the following objects exist in the Identity Vault:

- The owner
- The Resource Trustees
- Approvers
- Provisioning request definition

## User Application Driver Locales

For the User Application driver locales, Designer ensures that the locales contain descriptions and display names. You can turn off the validation of display names for each locale by setting a preference. For more information, see "Setting Provisioning View Preferences" on page 27.

# 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 the items 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 driver set'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.

### Deployment and Versions

If you deploy a User Application driver version 4.0 or later and the Identity Vault does not contain the necessary schema changes, the provisioning objects are not deployed and Designer displays an error message in the Deploy Results dialog box. This is to prevent you from deploying a 4.0 or later driver to a 3.0 Identity Vault.

## 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, right-click the object to deploy, then select **Live > Deploy** or **Live > 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 (if necessary), validates the objects, and writes any messages to the project checker view.

   When you deploy a *driver* that contains provisioning objects that fail validation, Designer deploys the driver but not the invalid objects (regardless of the deployment preferences). Designer displays the errors in the deployment result dialog box.

   When you deploy a *provisioning object* that contains validation errors, Designer performs the deployment based on the defined preferences and writes the errors to the Project Checker view.

- "Tips for Deploying Provisioning Request Definitions" on page 31
- "Deploying Roles" on page 32

## Tips for Deploying Provisioning Request Definitions

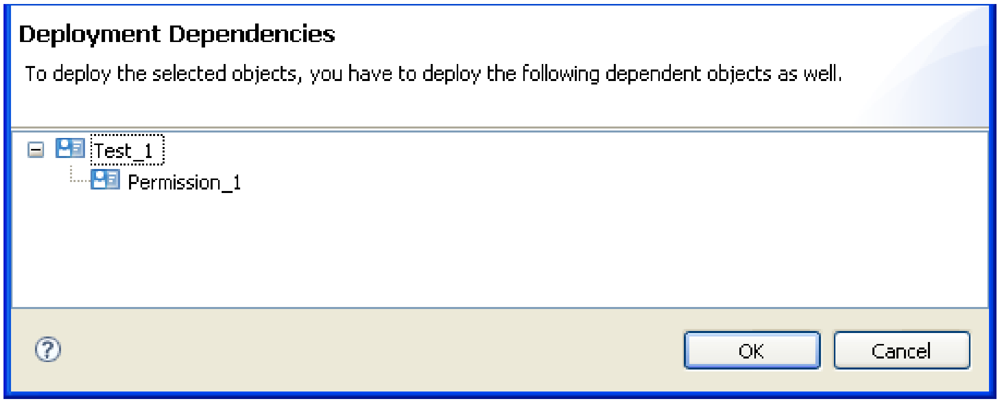
- If errors associated with activities are detected during deployment of a provisioning request definition, Designer identifies the activity in which the error occurred by activity Id. However, in the user interface, Designer by default displays activities by activity name. To make it easier to identify the activity in an error message, 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 and select **Show Activity Ids**.

- A common error occurs when you fail to replace a placeholder expression in an entitlement provisioning activity. If this is the case, correct the error, then deploy the provisioning request definition again.
- Designer cannot evaluate expressions at design time, so it might display a warning when you use an expression for an entitlement that must be resolved at runtime. This is not a fatal error and the deployment will succeed.
- Make sure that the **Status** is **Active** (in the **Overview** tab).
- If a provisioning request definition with the same CN already exists in the Identity Vault, the Deployment Summary displays the differences. You can review the differences before you decide to proceed.

## Deploying Roles

Because roles can be related through a role hierarchy, Designer notifies you, on deploy, if the role you are deploying contains any dependent roles. To ensure that roles in the Identity Vault are in a valid state, Designer requires that you deploy the role and any dependent roles at the same time by displaying them in the dialog box shown in Figure 2-2 on page 32.

***Figure 2-2** Deploying Dependent Roles*



**Deployment Dependencies**

To deploy the selected objects, you have to deploy the following dependent objects as well.

## Testing the Deployed Changes

You can access the User Application from within Designer to view or test what you deploy:

1. Select **Tools > Access User Application.**
2. Choose the project and User Application driver container associated with the User Application you want to view, then click **OK**.

   Designer uses the driver configuration information that you defined for the project to make the connection. Designer uses the browser settings specified in **Windows > Preferences > General > Web Browser**

# 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 driver set'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.

To compare provisioning objects:

1. Right-click a container or object in the Provisioning view, then select **Live > 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 deselecting **Only show differences**.

   **NOTE:** For provisioning teams, you must select the container to compare the provisioning request and provisioning team objects. If you select an individual team, it compares only the provisioning team objects.
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. |

If a provisioning request definition or role object contains trustees, the trustees for the local object are compared with the trustees defined for the object in the Identity Vault. Trustees are not compared for directory abstraction layer objects.

# Specifying Locales and Localization Resource Groups

- "Specifying the Default Locale" on page 34
- "Defining the User Application's Supported Locales" on page 34
- "Creating a Custom Localization Resource Group" on page 37

## Specifying the Default Locale

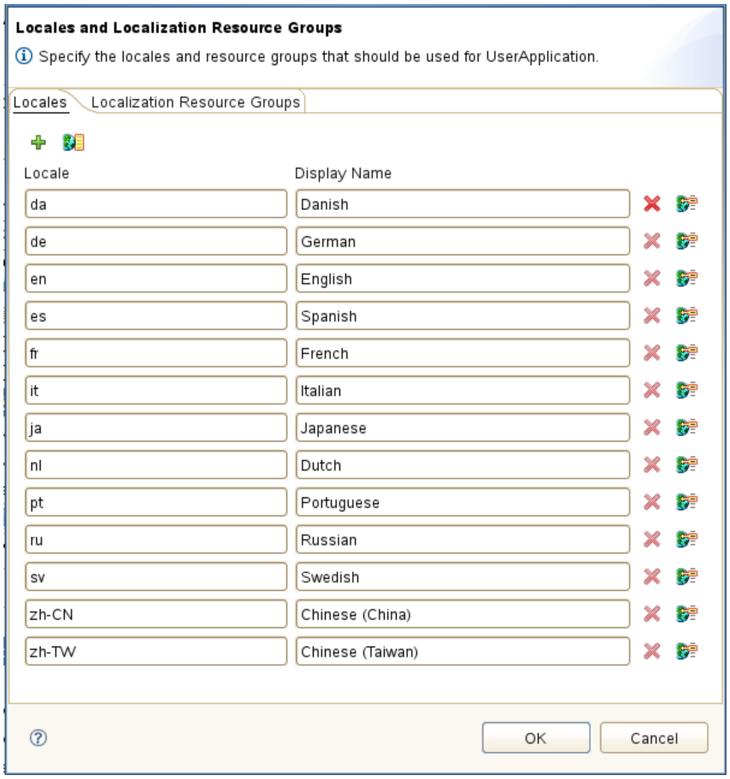
To specify the User Application driver’s default locale:

1 Right-click the User Application driver in the Provisioning view, then select **Configure > Default Locale**.

2 Select the locale from the drop-down list box, then click **OK**. If you do not see the locale in the list, you must add it through the Locales dialog box.

## Defining the User Application’s Supported Locales

1 Right-click the User Application driver in the Provisioning view, then select **Configure > Locales**. Designer displays the Locales and Localization Resource Groups dialog box.

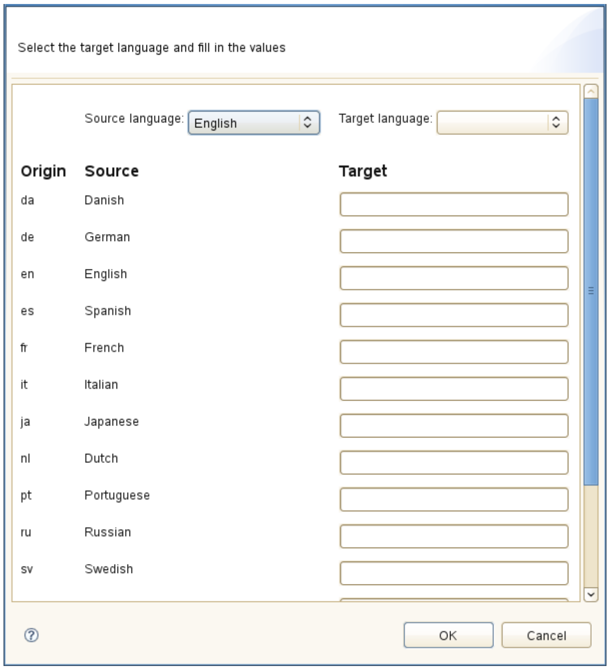


**Locales and Localization Resource Groups**

ⓘ Specify the locales and resource groups that should be used for UserApplication.

| Locale | Display Name |
|---|---|
| da | Danish |
| de | German |
| en | English |
| es | Spanish |
| fr | French |
| it | Italian |
| ja | Japanese |
| nl | Dutch |
| pt | Portuguese |
| ru | Russian |
| sv | Swedish |
| zh-CN | Chinese (China) |
| zh-TW | Chinese (Taiwan) |

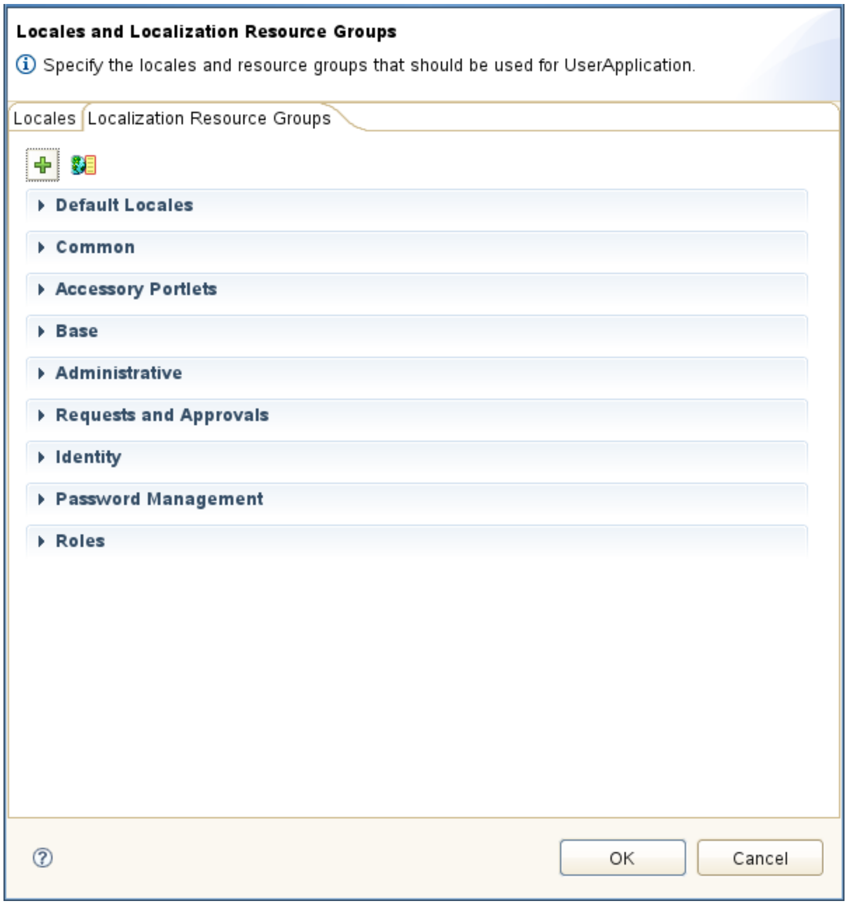
| Button | Description |
|---|---|
| + | Lets you add a new locale to the list of locales supported by the User Application driver. When you click the **Add** button, Designer prompts you for the Language and Country. You can either select them or type a value. The Language is required and displays as the locale. The country is not required. |
| ✕ | Deletes the selected Locale from the supported list. Any files for the selected locale are not deleted. |

| Button | Description |
|---|---|
|  | Lets you localize all supported locales in one dialog box. When you click the button, it launches this dialog box:<br><br>Select the target language and fill in the values<br>Source language: English<br>Target language:<br>Origin Source Target<br>da Danish<br>de German<br>en English<br>es Spanish<br>fr French<br>it Italian<br>ja Japanese<br>nl Dutch<br>pt Portuguese<br>ru Russian<br>sv Swedish<br> <br> <br>You can select the source language from which you want to translate the names of the supported locales. Select the target language into which you want to translate the names of the supported locales. Type the translation in the **Target** field. |
|  | Lets you localize the display labels for the selected locale. This is the same procedure described in "Localizing Provisioning Objects" on page 39. |

2 Click ✚, then select the Language and optionally the Country from the drop-down list, or type the value. The language is required, but the country is not. The language is displayed as the Locale in the Supported Locales dialog box.

3 Click **OK**.

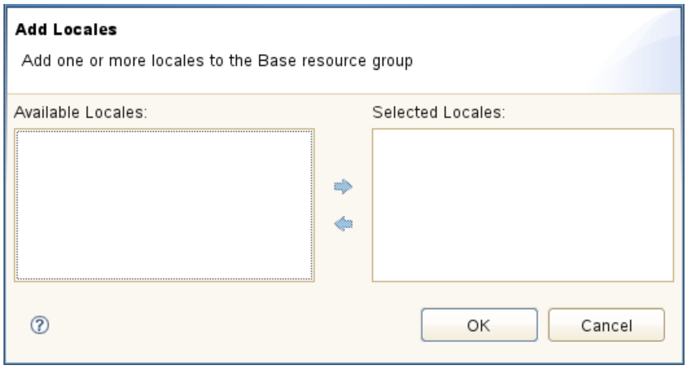
4 Click *Localization Resource Groups*, then click the localization resource group that you want to support the new locale.



Locales and Localization Resource Groups

Specify the locales and resource groups that should be used for UserApplication.

5 Click ✚ in the Locales area, then select the locale from the **Available Locales** list and move it to the Selected Locales list, then click **OK**.

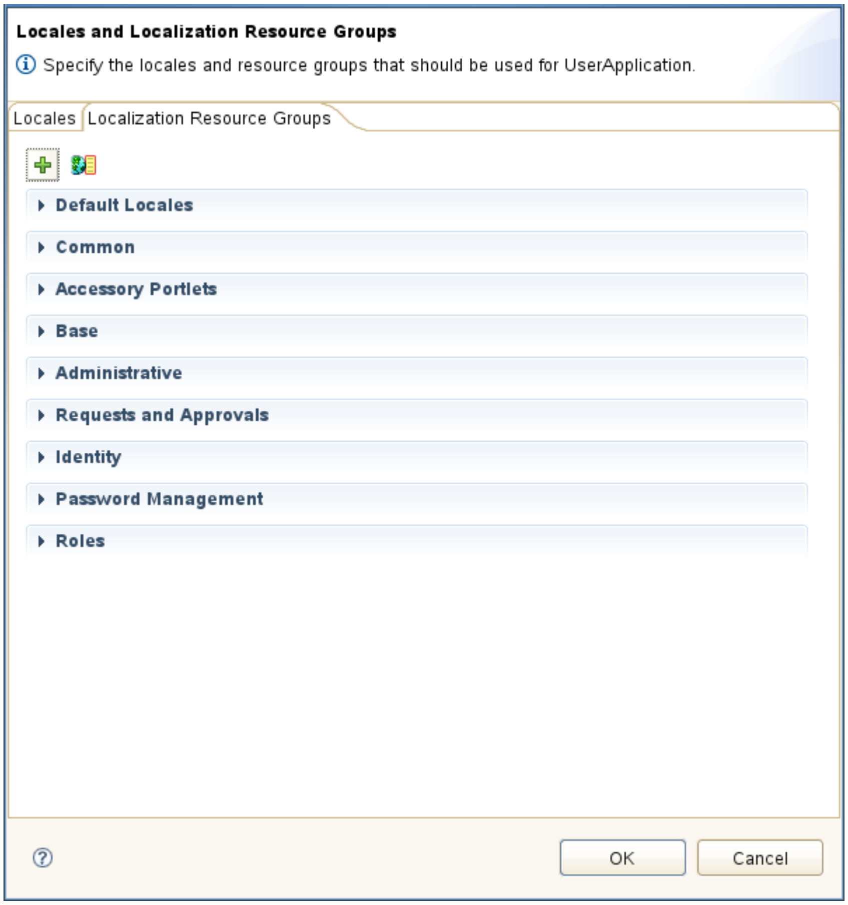


Add Locales

Add one or more locales to the Base resource group

# Creating a Custom Localization Resource Group

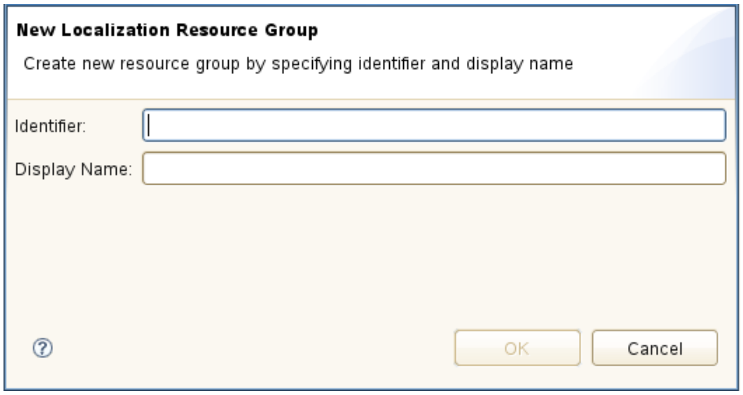
1. Right-click the User Application driver in the Provisioning view, then select **Configure > Locales**. Go to the **Localization Resource Groups** tab.



Locales and Localization Resource Groups

Specify the locales and resource groups that should be used for UserApplication.

2. Click ✚.



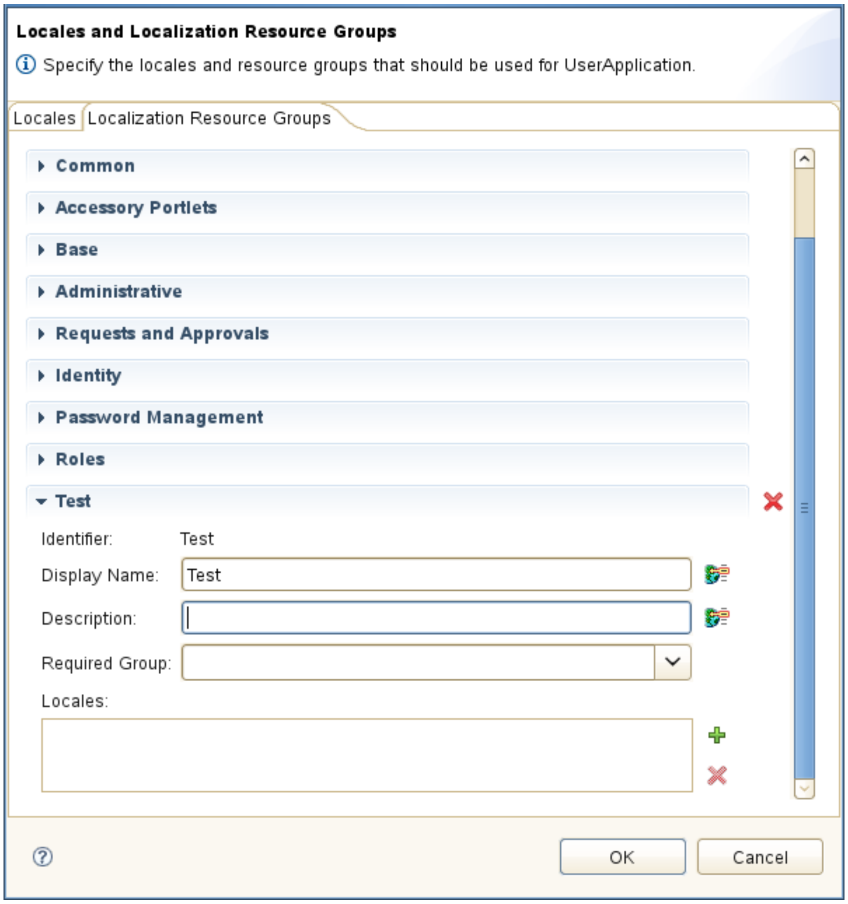
New Localization Resource Group

Create new resource group by specifying identifier and display name

**3** Complete the fields as follows:

| Field | Description |
|---|---|
| Identifier | A unique name used to identify the localization resource group. |
| Display Name | The name displayed in Designer to correspond with this group. |

**4** Click **OK**. The Localization Resource Groups property dialog box displays:



**Locales and Localization Resource Groups**

ⓘ Specify the locales and resource groups that should be used for UserApplication.

Identifier: Test

Display Name: Test

Description:

Required Group:

Locales:

**5** Complete the fields as follows:

| Button | Description |
|---|---|
| ✚ | Lets you add a new localization resource group. |
| ✖ | Deletes the current localization resource group. |
| 🌐 | Lets you localize the localization resource groups in one dialog box. |
| **Description** | Lets you add descriptive text for this entry. |

| Button | Description |
|---|---|
| **Required Group** | Lets you specify dependencies on other localization resource groups.<br>Localization resource groups are used for creating resource bundles for non-standard language. The required groups are defined in the *User Application: Localization Toolkit Guide*. Contact your Novell Sales Representative for more information about the toolkit. |
| **Locales** | Lets you add or remove the locales into which this localization resource group must be localized. If the locale is not on the list, see "To specify the User Application driver's default locale:" on page 34. |

# Localizing Provisioning Objects

- "Using Designer to Localize" on page 40
- "Supported Languages" on page 41
- "Exporting and Importing Data to Localize" on page 41

Designer allows you to translate the names and descriptions of provisioning objects into multiple languages. Table 2-3 describes the types of provisioning objects that you can translate.

***Table 2-3** Localizable Objects*

| Designer Tool | Description |
|---|---|
| Directory Abstraction Layer Editor | ◆ Entity and attribute display labels<br>◆ Relationship names<br>◆ Global and local list items<br>◆ Query display labels and parameter display labels |
| Provisioning Request Definition Editor | ◆ Activity properties that are displayed to the user<br>◆ Form properties that are displayed to the user |
| Provisioning Team Editor | ◆ Provisioning team display name and descriptions |
| Role Catalog | ◆ Resource display label and description<br>◆ Role display label and description<br>◆ Separation of Duties display label and description<br>◆ Role level display label and description |

To localize the provisioning objects listed in Table 2-3:

1. Verify that the locale (or language) is supported by the User Application driver. See "Supported Languages" on page 41 for the list of languages supported by default.
2. If necessary, add the new locale (or language) to the User Application driver and to the resource groups. For more information, see "Defining the User Application's Supported Locales" on page 34.

**3** Translate the names and descriptions in one of the following ways:

**3a** Directly within Designer.

**NOTE:** You cannot edit the provisioning request definitions in the Attestation category. For this reason, you cannot use this method for localizing them. You must use the method described in Step 3b.

For more information, see "Using Designer to Localize" on page 40.

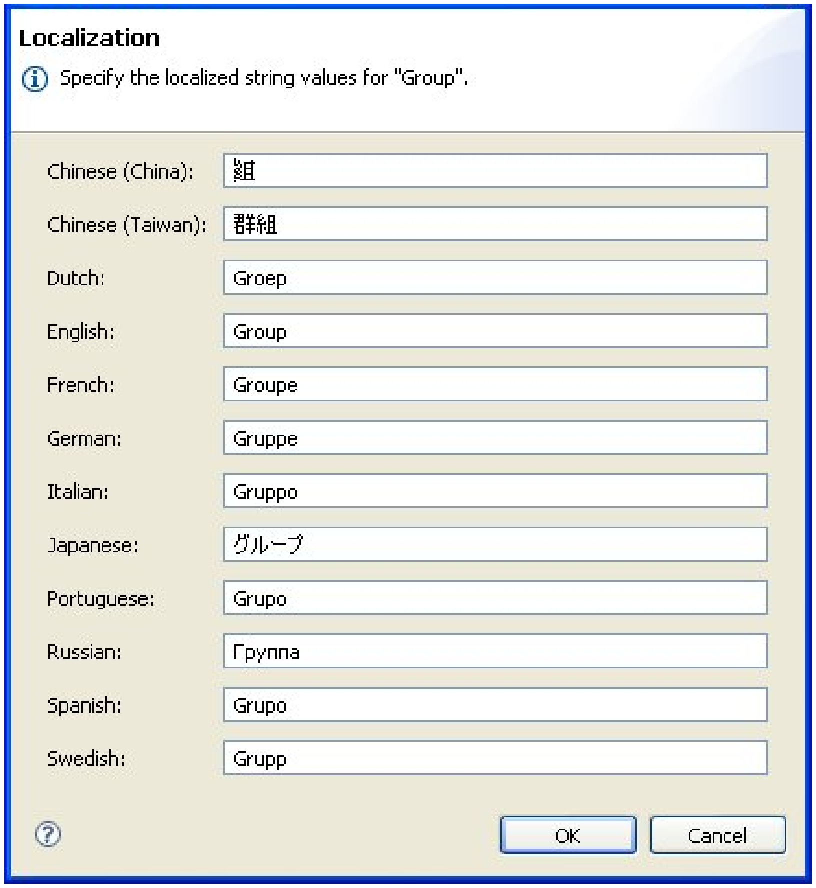
**3b** By exporting the set of localizable objects into an external properties or XML file, translating the contents of the file, then importing the data back into the project.

For more information, see "Exporting and Importing Data to Localize" on page 41).

## Using Designer to Localize

**1** Click the localize button.

When you click this button, Designer displays a dialog box that lets you add the localized text. This is an example of the Localization dialog box.



**Localization**

Specify the localized string values for "Group".

Chinese (China): 组

Chinese (Taiwan): 群組

Dutch: Groep

English: Group

French: Groupe

German: Gruppe

Italian: Gruppo

Japanese: グループ

Portuguese: Grupo

Russian: Группа

Spanish: Grupo

Swedish: Grupp

The languages displayed in this dialog box are the languages currently supported by the User Application driver. If your language is not shown in this dialog, you must add it. For more information, see "Defining the User Application's Supported Locales" on page 34.

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

***Table 2-4*** *Accessing the Localization Dialog Boxes*

| To define the localization text for... | Perform this action... |
|---|---|
| Every localizable item in the directory abstraction layer | Select **DAL > Set Global Localization**.<br>or<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, right-click the object to localize, 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). |

## Supported Languages

You can localize the display labels, display names, and descriptions into the languages listed in the localization dialog box. This list represents the languages (locales) supported by the User Application driver. For information about adding new languages to this list, see "Specifying Locales and Localization Resource Groups" on page 33.

The locale configuration is stored in the driver's `<default-locale>` element in the AppConfig.AppDefs.locale-configuration XMLData attribute.

You *must* provide a display label for the User Application driver's default language, or the User Application generates the following runtime error: `The resource resolver com.novell.soa.common.i18n.LocalizedMapResolver did not return a resource for the default locale of <locale>. It is required that a resource exist for the default local`.

## Exporting and Importing Data to Localize

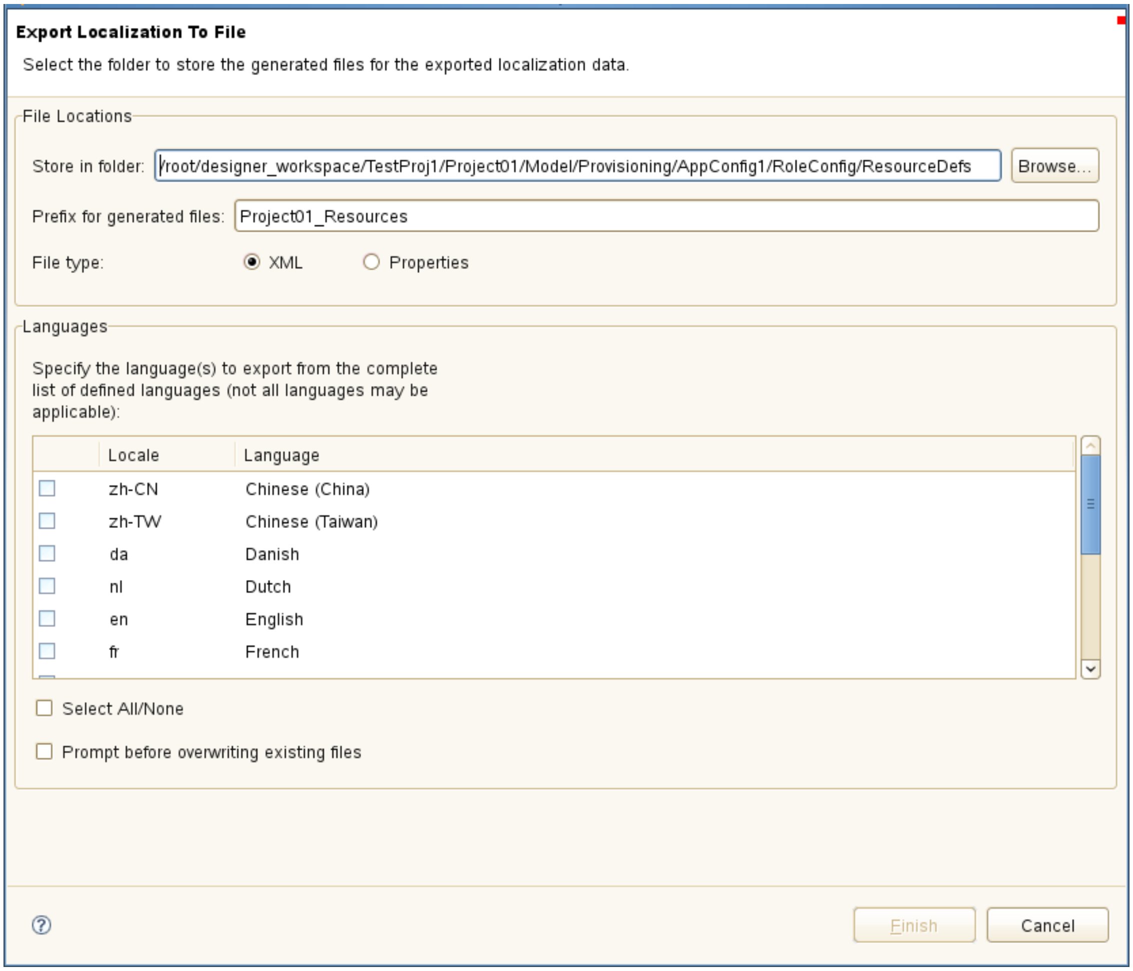
You can export the localizable data (such as display names and descriptions) in your project to an XML or properties file. After the data in that file is translated, you can import it back to the Designer project. You can export an entire driver, one object, or a subset of objects.

- "Exporting Data to Localize" on page 42
- "Importing Localized Files" on page 44

## Exporting Data to Localize

1 Right-click a container node or an object in the Provisioning view.

2 Select **Localize > Export Localization to File**.



**Export Localization To File**

Select the folder to store the generated files for the exported localization data.

File Locations

Store in folder: /root/designer_workspace/TestProj1/Project01/Model/Provisioning/AppConfig1/RoleConfig/ResourceDefs

Prefix for generated files: Project01_Resources

File type: ● XML ○ Properties

Languages

Specify the language(s) to export from the complete list of defined languages (not all languages may be applicable):

| | Locale | Language |
|---|---|---|
| □ | zh-CN | Chinese (China) |
| □ | zh-TW | Chinese (Taiwan) |
| □ | da | Danish |
| □ | nl | Dutch |
| □ | en | English |
| □ | fr | French |

□ Select All/None

□ Prompt before overwriting existing files

3 Fill in the fields as follows:

| Field | Description |
|---|---|
| **Store in folder** | Specify the name of a local folder where the exported files should be written. |
| **Prefix for generated files** | Specify a prefix for the generated files. Determine a naming strategy so you are able to identity the files for projects. |
| **File Type** | Select **XML** or **Properties** depending on the encoding or format you prefer. XML files are UTF-8 encoded. Properties use Unicode*. |
| **Select the languages to export** | Select the languages you want localizations for. A file containing the display label key is generated for that language. The localizations need to be added to this file in the proper format so you can import them to the proper User Application driver objects. |
| **Prompt before overwriting existing files** | If this option is selected, Designer prompts you before it overwrites any existing files of the same name in the target directory. |

4 Click **Finish**. Designer displays a message describing the result of the export operation and the location of the exported data.

## Importing Localized Files

1 Right-click a container node or an object in the Provisioning view, then select **Localize > Import Localization from File**.

2 Fill in the fields as follows:

| Field | Description |
|---|---|
| **Search in folder** | Specify the folder location where the files to import are located. |
| **File Type** | Select **XML** if the file you want to import is in XML format.<br>Select **Properties** if the file you want to import is in the properties format. |
| **Preferences** | Select **Suppress warnings about unused strings** if you want the wizard to suppress warning messages.<br>Select **Create backup of existing display label strings** if you want the wizard to create a backup of the existing strings before the import. Useful in case you need to revert. |
| **Files** | Select the files to import. This table is populated with the files from the folder location and file type specified above. If it is blank, no files of the specified type are located in the target folder. The wizard attempts to determine the language by looking at the filename. If the name cannot be determined, it defaults to English.<br>You can change the **Language** column if the wizard assumes the wrong language. The wizard changes the filename to reflect the language you specify and import the display labels to the corresponding language. |

3 Click **Finish** to complete the import. Designer displays a status dialog box that describes the results including any errors reading the files and any warnings about display label keys that are unused.

# 3 Configuring the Directory Abstraction Layer

This section provides details on configuring the directory abstraction layer.

## 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 User Application uses the directory abstraction layer definitions to determine:

- The Identity Vault objects and attributes that the User Application can display or modify.
- How the User Application displays Identity Vault data.
- The relationships the User Application can display.
- The provisioning request categories, email notification types, and delegate relationships the User Application can display.

The User Application ships with a default set of entities, relationships, and lists that it needs to function, but you can add new or modify existing directory abstraction layer objects to customize the User Application for your own business needs. You use the directory abstraction layer editor to define the contents of the directory abstraction layer.

### Analyzing the User Application’s Data Needs

Before you make changes to the directory abstraction layer objects, analyze how you want to display your Identity Vault data in the User Application. Consider:

- What parts of the Identity Vault you want to make available to the User Application.

  For example, what objects do you want your users to be allowed to search and display? Check this list against the base set of abstraction layer definitions to determine if you need to add any new objects.

- What is the structure of your Identity Vault schema? Have you added custom extensions and auxiliary classes?
- What is the structure of your data?
  - What is required and what is optional?
  - What validation rules are in place?
  - What are the relationships between objects (DN references)?
  - How are the attributes defined? (For example, an attribute that represents a phone number might be multi-valued for home, office, and cell phone numbers)
- Who sees the data? Is the User Application available as a public or private site?

Use the information about your data needs to map your Identity Vault objects to abstraction layer entities.

## 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 and run the configuration wizard, Designer creates an initial set of directory abstraction layer files. If you do not run the configuration wizard, the initial files are not created. 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 the directory abstraction layer tree containing nodes for **Entities**, **Lists**, **Queries, Relationships**, and **Configuration**.

| 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. |
| **Queries** | Lets you define LDAP search criteria that can be run from a workflow form. |
| **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 and in iManager when defining provisioning. |
| **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 properties for the selection.
3. Use the right pane to define the properties for the selection. For more information about the properties, see "Directory Abstraction Layer Property Reference" on page 59.

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 Query Wizard |
|  | Launches the New Relationship Wizard. |
|  | Launches the Set Global Access Modifiers dialog box. |
|  | Launches the Set Global Localization dialog box. |
| ⊞ ⊟ | Expands and collapses the directory abstraction layer tree. |

# 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.

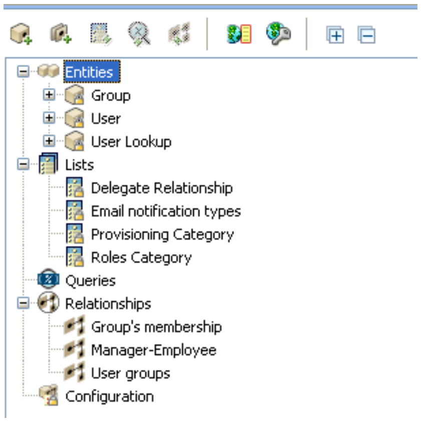
***Table 3-2*** *Local Directory Abstraction Layer Directories*

| 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. |
| `QueryDefs` | Contains the files that define queries. Files have the query extension. |
| `RelationshipDefs` | Contains the files that define the relationships available to the Org Chart portlet and iManager provisioning configuration. These files have the `relation` extension. |

Designer creates the base set of directory abstraction layer files for each provisioning project. An identical set is added 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 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 **Group**, **User** or **User Lookup** entities.

***Figure 3-1** DAL User Application Default Entities, Lists, and Relationships*



- Entities
  - Group
  - User
  - User Lookup
- Lists
  - Delegate Relationship
  - Email notification types
  - Provisioning Category
  - Roles Category
- Queries
- Relationships
  - Group's membership
  - Manager-Employee
  - User groups
- Configuration

If you define multiple User Application drivers in a single project, Designer creates multiple AppConfig folders and names them AppConfig, AppConfig1, AppConfig2, and so on.

# 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 "Adding Entities" on page 49 and "Adding Attributes" on page 53. To modify the entity files of an already ed 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 "Importing Provisioning Objects" on page 27.

## About Entities and Attributes

Any Identity Vault object that you want users to search, display, or edit in the Identity Manager User Application must be defined as an *entity* in the directory abstraction layer. For example, to use the inetOrgPerson Identity Vault object in the User Application, you must create an entity definition for it. There are two logical kinds of entities (but you create them the same way):

- **Entities that are mapped from schema:** These entities represent objects that exist in the Identity Vault that are directly exposed to users in the User Application. When defining this type of entity, expose all of the attributes that you want your users to work with. Examples of this entity type include User and Group. You can create more than one entity definition for the same object to expose different sets of attributes to different kinds of users. For more information, see "Creating Multiple Entity Definitions for a Single Object" on page 49.

- **Entities that represent LDAP relationships:** This type of entity is known as a *DNLookup* and it is used by the User Application to:
  - Populate a list with the results of a DN search among related entities
  - Maintain referential integrity across DN referenced attributes during updates and deletes

  Entities that support DNLookups are used by the Org Chart portlet to determine relationships and are also used by the Search, Create, and Detail portlets to provide pop-up selection lists and DN contexts. The User Lookup entity is an example of this type of entity. For more information, see “Attributes and DNLookup Properties” on page 68.

## Creating Multiple Entity Definitions for a Single Object

You can create more than one entity definition that represents the same Identity Vault object but provides a different view of the data. Within the entity definitions, you can define different attributes for each entity definition, or you can define the same attributes but specify different access properties that control how the attributes are searched, viewed, edited, or hidden.

**NOTE:** You can optionally define a filter to hide certain entities from the result set.

You can then use these different entity definitions in different parts of the user interface. For example, suppose that you want to create a directory of employees; one for a public site and one for an internal site. On the public site you want to supply first and last names and a phone number, but on the internal site, you want to list additional information like title, managers, and so on. Here’s how you can accomplish this:

1. Create two entity definitions (with different keys).

   Both entity definitions expose the same Identity Vault object, but one entity definition key is public-staff-information, and the other is internal-staff-information.
2. Within each entity definition, define a different set of attributes: one for public-staff-information, the other for internal-staff-information.
3. Use the **Portal Administration** tab of the Identity Manager User Application to create a portlet instance for the public page, and another one for the internal page.

   For more information about creating portlet instances, see the Portlet Reference section in the I*dentity Manager 3.5 User Application: Administration Guide*.

# Adding Entities

You add entities through the Add Entity Wizard (described in the next procedure) or by clicking **Add Entity** (from the 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:

- Right-click the **Entities** node, then choose **New**.

From the directory abstraction layer editor:

- Select **DAL > New > Entity**

  or

- Right-click the **Entities** node, then choose **New Entity-Attributes Wizard**.

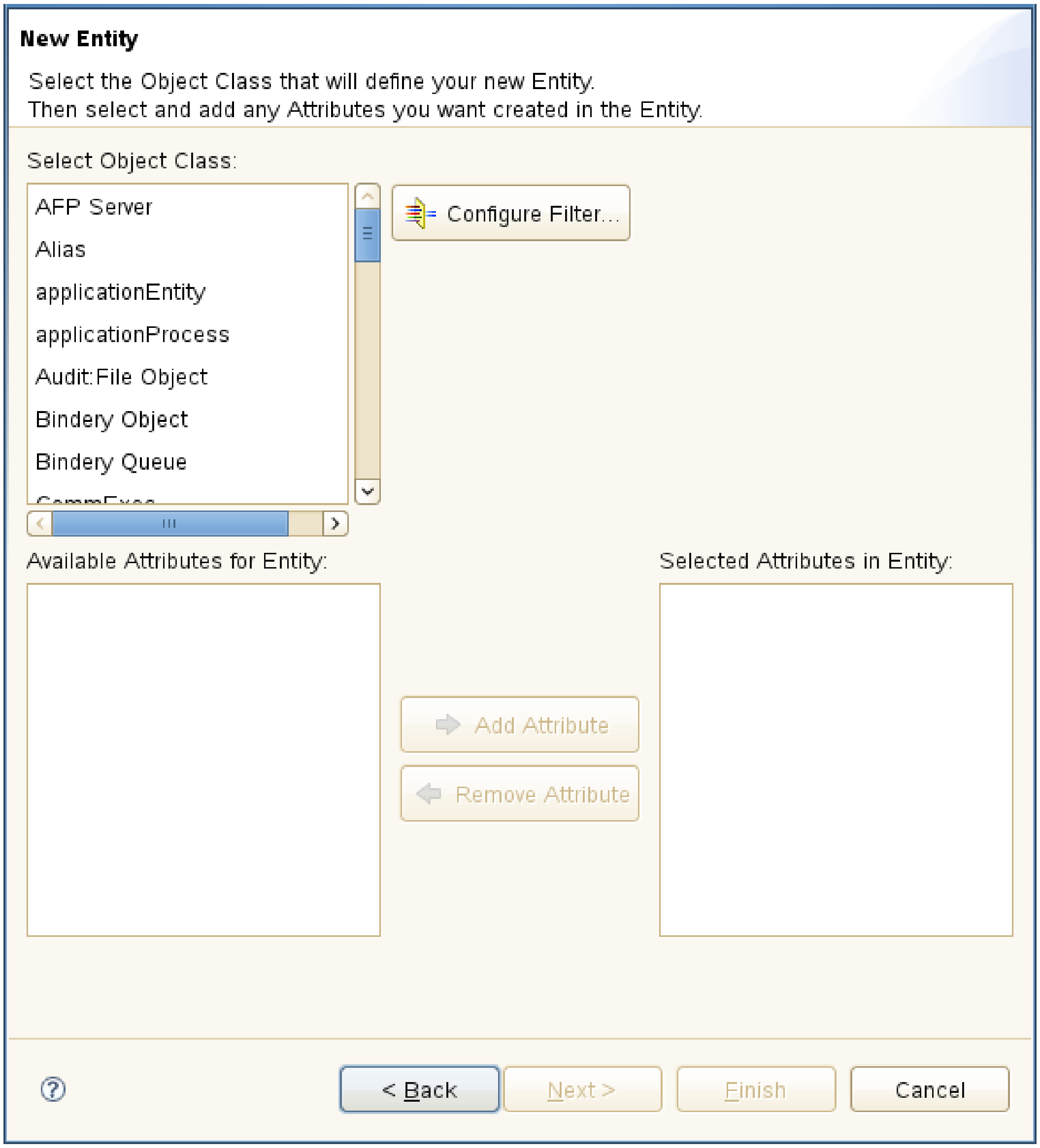
The New Entity dialog box displays.

**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>**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 “Localizing Provisioning Objects” on page 39. |

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



**New Entity**

Select the Object Class that will define your new Entity.
Then select and add any Attributes you want created in the Entity.

Select Object Class:

AFP Server
Alias
applicationEntity
applicationProcess
Audit:File Object
Bindery Object
Bindery Queue

Available Attributes for Entity:

Selected Attributes in Entity:

**4** Choose the entity's object class and add the attributes you want by double-clicking them in the **Available Attributes for Entity** list. Mandatory attributes are added when you select an **Object Class**, and you cannot remove them from the **Selected Attributes in Entity** list.

**TIP:** If the entity's object class is not shown in the **Select Object Class** list, you should update Designer's local schema file by following the steps described in "Updating the Schema Elements List" on page 54.

**5** Click **Finish**.

The property page displays for editing. For more information, see "Entity Properties" on page 60. You must deploy the entity before it is available to the User Application.

## Filter the Object Class List

You can limit the object classes shown in the New Entity dialog box by adding a filter. To add a filter:

1. Click **Configure Filter** to launch the Class List Filters dialog box.

   By default, Designer does not apply any class filters. The Class Filter dialog box contains two predefined filters (**starts-with "DirXML"** and **starts-with "srvprv"**). To activate them, click **Select All**, then click **OK**. The filters are immediately applied to the object class list. Filters are applied until you deselect them.
2. Use the buttons as follows:

| Button | Description |
|---|---|
| | Choose one of the string comparison operators, such as contains, starts-with, ends-with, then type the string to compare against. |
| ✚ | Adds a filter. Enabled when you define the filter comparison value. |
| ✖ | Removes the selected filter. |
| Select All | Click this option when you want to use all of the filters. It selects all of the defined filters. |
| Deselect All | Click this option when you want to deselect all of the defined filters. If you apply this change, no filters are used. |

## Adding Entity Filters

In the Directory Abstraction Layer Editor, you can define an entity filter to limit the entries returned for the specified entity. You define the filter based on attributes and their comparison to another value that you specify. For example, you can create a filter so that the User entity includes only those entries whose Region attribute contains Northeast.

1. Click **Add Condition Grouping**.
2. Use the drop-down list on the left to select an attribute.
3. Use the drop-down list in the middle to select a comparison operation.
4. Use the entry on the right to specify a value for comparison.
5. To specify multiple condition groupings, repeat this procedure. Within a condition grouping, you specify each criterion that you want and connect them by using the logical operations: and, or.

The conditions are evaluated in the order in which you define them.

## Adding Attributes

1. Select an entity.
2. Do any of the following to add an attribute:
   - Right-click an entity, then select **Add Attribute.**
   - Click the **Add Attribute** button.
   - Click **DAL > New > Attribute**.

   You are prompted to choose the entity class that contains the attributes that you want to add to the entity. You can also add (and remove) auxiliary classes if you need to add a class that contains the attribute you are looking for.
3. Add attributes by double-clicking them in the **Available Attributes for Entity Class** list.

   LDAP operational attributes are supported by the directory abstraction layer editor and User Application; however, when you add an operational attribute, the Edit, Required, and Hidden properties are set to false and are disabled so you cannot change these property values.

   **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 54.
4. Click **OK**. The property page displays for editing.

   For more information, see "Attribute Properties" on page 63. To make an attribute available to the User Application, you must deploy it.

## Adding DAL Calculated Attributes

You can create an attribute that is derived from an expression. For example, you can concatenate two or more attributes to produce a single calculated value. The expressions are ECMAScript compatible and conform to the ECMA 262 Language specification.

**Restrictions:** Because this attribute type does not map to a specific attribute in the Identity Vault, these attributes cannot be updated, removed, multivalued, required, or searched.

To create a calculated attribute:

1. Add an attribute as instructed in "Adding Attributes" on page 53 and make sure to select **DAL Calculated Attribute** from the **Available Attributes for Entity Class** list.

Designer adds the Attribute with the following restrictions:

*Table 3-3 Calculated Attribute Properties*

| Property Name | Description |
|---|---|
| Expression | Click **Build ECMAScript Expression** to launch the ECMA Expression Builder. To learn more about how to use the ECMA Expression Builder, see Chapter 9, "Working with ECMA Expressions," on page 271. |

## Updating the Schema Elements List

1. With the Identity Manager project open, right-click your Identity Vault, then 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.**

# Working with Lists

The lists node lets you define the contents of global lists. You can then define an attribute control type as a global list. When the User Application displays the attribute for editing, the contents of the global list are displayed in a drop-down list for the user to make a selection. By default, the directory abstraction layer includes the global lists described in Table 3-4.

***Table 3-4*** *Directory Abstraction Layer Default Global Lists*

| List Name | Description |
|---|---|
| Delegate Relationship | Defines the relationships that can be selected when making a Delegate Assignment by relationship. The contents of this list display in a drop-down list box. The values can only be DN attributes from the User entity. |
| Email Notification Types | Represents the type of email notification that a user wants to receive when involved in proxy/delegate processing of resource requests. Types are locked.<br>**WARNING:** Do not edit these values.<br>This is used by the Preferred Notification attribute of the user entity. |
| Provisioning Category | Defines the set of categories that organize provisioned resources (entitlements) and provisioning requests. The categories in this list display in:<br>◆ Designer: Provisioning request definition editor plug-in<br>◆ iManager: Provisioning Request Configuration plug-in<br>◆ User Application: Make a Process Request page |
| Resources Category | Defines the set of categories that organize resources. The categories are displayed in:<br>◆ Designer: Role Plug-in<br>◆ User Application: Roles and Resources tab |
| Roles Category | Defines the set of categories that organize roles. The categories are displayed in:<br>◆ Designer: Role plug-in<br>◆ User Application: **Roles** tab |

**NOTE:** You cannot delete these lists or change the key values for the lists. Except for the Email Notification types, you can add and remove items and change existing values and labels.

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**, select **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:

- Right-click the Lists node, then select **New**.

From the directory abstraction layer editor:

- Click **New List.**
- Right-click the Lists node, then select **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>**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 “Localizing Provisioning Objects” on page 39. |

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. This is the name displayed 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 |
| ▲<br>▼ | Moves the row up or down in the list. This order specifies how the labels are displayed in the User Application. |
| ☰ | Displays the localization dialog box. For more information on using the dialog box, see “Localizing Provisioning Objects” on page 39. |
| ✖ | Deletes the row. |

5 Save the project to make it available to the User Application.

# Working with Queries

The Queries node allows you to define commonly used LDAP searches that you can execute from a request or approval form by using the DNQuery control or by calling the globalQuery() method. To define the query, you specify the directory abstraction layer entity, the search root, the number of rows to retrieve, and the conditions for retrieving the source entity. You can hard-code the conditions (for example, Where LastName contains s) or specify one or more parameters that are supplied by the user on the request or approval form.

To create a query:

1 Launch the New Query Wizard in any of these ways:

From Designer’s menus:

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

From the Provisioning view:

- Right-click **Query**, then select **Add**.

From the directory abstraction layer editor:

- Click the **Add Query** button.
- Right-click **Query**, then select **Add Query**.

The New Query 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>**NOTE:** This field displays when you create queries from the **File** menu. |
| **Query Key** | Type a unique value for the query key. This value is used in the Expression Builder to identify the query. |
| **Display Label** | Type a string to display in the directory abstraction layer editor and Provisioning view. This value is not visible in the Expression Builder. |

**3** Click **Finish**.

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

**4** Select a **Query Entity**. If the entity you want to use is not displayed, make sure it is defined in the Entities node.

**5** In the **Parameters** section, define one or more parameters for the query. To add parameters:

**5a** Click **Add Row**.

**5b** Specify a unique key and a display label for the parameter. You pass this key when calling the globalQuery() method on a form. For more information on globalQuery(), see "globalQuery(fieldname, key, param)" on page 297.

**5c** Add additional parameters by repeating these steps.

**6** To further refine the query, add **Query Conditions**.

**6a** Click **Add Condition Grouping** (a **Query Entity** must be selected to enable **Add Condition Grouping**).

**6b** Use the drop-down list on the left to select an attribute. The attributes in this drop-down are the attributes on the selected **Query Entity**.

**6c** Use the drop-down in the middle to select a comparison operation to perform against your chosen attribute.

**6d** Use the entry on the right to specify a value to compare against your chosen attribute. You can select a variable name by clicking **Predefined Parameters** to launch the Predefined Parameters dialog box.

If the query needs to filter on more than one attribute or condition and you want to control the order in which the conditions are evaluated, you can define multiple conditions or condition groups. Within a condition grouping, you specify each criterion that you want and connect them by using the logical operations: and, or.

**7** To specify multiple condition groupings, click **Add Condition Groupings** and make your selections from the drop-down list boxes.

**8** Define the query's LDAP Search properties if you want to narrow the search further than already defined for the selected entity. The query's search root, unlike the entity search root, does not support the use of predefined parameters. For more information, see "Queries Properties" on page 72.

**9** Click **Save**.

**10** the query to make it available to the User Application.

# Working with Relationships

The Relationships node allows you to define relationships between entities defined in the directory abstraction layer. The relationships you define are used in the User Application by the Organization Chart and in iManager for defining the members within a group.

The relationship you define can be between like entities (such as user/user) or unlike entities (such as user/device). You can define conditions for the relationship to further refine it. For example, you might want to create a condition that shows all Manager-Employee relationships and then refine it to show only employees in one particular region, or show all the subordinates of a vice president located in the eastern region.

The following relationships are defined, by default, for the User Application:

- Group’s membership (Org Chart only)
- Manager-Employee (Org Chart and Management)
- User groups (Org Chart only)

A relationship can only be used by Management when the Source and Target entities are both related to the InetOrgPerson object.

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

1 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:

   - Right-click **Relationships**, then select **Add**.

   From the directory abstraction layer editor:

   - Click the **Add Relationship** button.
   - Right-click **Relationships**, then select **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>**NOTE:** This field displays when you create relationships from the **File** menu. |
| **Relationship Key** | Type a unique value for the relationship key. |
| **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 “Relationship Properties” on page 73.

To delete a relationship:

1 Right-click the relationship you want to delete, then click **Delete**.

To add a relationship condition:

1 Click **Add Row**.
2 Use the drop-down list box (on the left) to select an attribute. The attributes in this drop-down are attributes on the **Target Object** entity.
3 Select an operator from the middle drop-down list box.
4 Use the text field on the right to specify the comparison value to complete the condition.

You can create a condition that filters on more than one attribute or condition and connect the attributes by using the logical operations: and, or. The conditions are evaluated in the order in which you define them.

# Working with Configuration Settings

The Configuration node allows you to set general configuration properties for the User Application.

***Table 3-5*** *Configuration Settings*

| Field | Description |
|---|---|
| **Default ‘My Profile’ Entity** | Defines the entity to display when the user clicks **My Profile** in the user interface.<br>This field is restricted to show only entities whose object class is user (or LDAP inetOrgPerson). |
| **Default LDAP Naming Attribute** | Defines the default LDAP naming attribute if the entity’s Create Naming Attribute is not defined. |
| **Default Management Attributes** | The attributes used to look up members. In the User Application, the user is able to search for members by clicking the search icon. The search displays the attributes specified.<br>These settings only affect the lookups performed by Managers. The User Application administrator sees only First Name and Last Name. |
| **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. |

# Directory Abstraction Layer Property Reference

This section provides definitions for the properties for the abstraction layer nodes.

# Entity Properties

This section describes the properties you can set for 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. |
| Edit | When deselected, this object cannot be changed by the User Application regardless of the underlying ACLs.<br>When selected, this object is editable, 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. It is defined when the entity is created and cannot be modified after the entity is created. |
| 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). |

## Entity Auxiliary Properties

***Table 3-8** Entity Auxiliary Properties*

| Property Name | Description |
|---|---|
| Auxiliary Classes | A list of zero or more auxiliary classes for this entity. If you are adding auxiliary classes, you are prompted to define:<br>◆ The auxiliary class by selecting from the list of those available<br>◆ Whether it is searchable. Setting searchable to True applies a filter to LDAP searches that involve directory abstraction layer relationships. For example, if you added an aux class to the user entity and specified that the aux class was searchable, the Org Chart (using the manager-employee relationship) would display only the employees that have the aux class.<br>◆ Whether to **Add Always**. When True (selected), the object class is automatically added when the entity is modified in the User Application. Modification includes create or update operations. When False, the object class is only added if an attribute associated with the auxiliary class is modified. |

## Entity Search Properties

***Table 3-9** 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>`ou=sample,o=ourOrg`<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 63. |
| Search Scope | Specifies where the search occurs in relation to the search root. Values are:<br>**<Default>**: This search scope is the same as selecting **Containers and subcontainers**.<br>**Container**: The search occurs in the search root DN and all entries at the search root level.<br>**Container and subcontainers**: The search occurs in the search root DN and all subcontainers. This is the same as selecting **<Default>.**<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. |

| Property Name | Description |
|---|---|
| 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 it between 100 and 200 for greatest efficiency. Do not set it over 1000. |
| Perform Automatic Query | When selected, performs an automatic query of the entity and presents the results in a selectable list. Do not choose this option if the data returned will be a large number because it forces the user to scroll through a large result set.<br>When not selected, allows the user to specify the search criteria for the entity query, then presents the results in a selectable list. |

## Entity Create Properties

***Table 3-10** Entity Create Properties*

| Property Name | Definition |
|---|---|
| Create Container | The name of the container where a new entity of this type is created.<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 63.<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. |
| Create 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. |
| LDAP attribute | The LDAP attribute for the Create Naming Attribute. |
| Create Naming Label | Display label displayed in the User Application for the Create Naming Attribute. |

## Entity Password Management Properties

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

| Property Name | Definition |
|---|---|
| Password required when entity is created | If the password attribute is required, set this value to True (selected) to ensure that one is required by the Create portlet. If a password is required, then you cannot create this entity in a workflow. |

## 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. |

# Attribute Properties

This section describes the properties you can set for attributes.

## Attribute Access Properties

**NOTE:** You can set attribute access for all of an entity's attributes by selecting **DAL > Set Attribute Access,** right-clicking an entity, and selecting **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. |

| Name | Description |
|---|---|
| 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 a photo) is displayed by the application.<br>When deselected, the **Hide** check box is disabled for this attribute, so the user cannot choose to hide this attribute.<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.<br>◆ He or she is either the owner of the attribute or a User Application Administrator.<br>◆ He or she has Trustee rights to update the srvprvHideAttributes attribute on the Identity Vault.<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>**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.<br>The Search and Hide properties are mutually exclusive. If Hide is selected (True), Search cannot also be selected (True). If Search is selected (True), Hide cannot be selected (True). |
| Multivalue | Specifies whether this attribute can be multivalued, for example, a phone number.<br>When selected, the attribute can be multivalued. |
| Read | When this option is 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.<br>You must set this value to `True` for attributes that you want to include in the **Create User** form. For more information about creating users, see Creating Users or Groups in the *NetIQ Identity Manager - User’s Guide to the Identity Applications*. |
| Require | When this option is selected, the attribute must be supplied. |
| Search | When this option is 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>**TIP:** If an attribute used in a search is also indexed in eDirectory, the search is faster.<br>The Search and Hide properties are mutually exclusive. If Hide is selected (True), Search cannot also be selected (True). If Search is selected (True), Hide cannot be selected (True).<br>You must set this value to `True` for attributes that you want to include in the **Create User** form. For more information about creating users, see Managing Your Permission Requests in the *NetIQ Identity Manager - User’s Guide to the Identity Applications*. |
| View | When this option is 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 General Properties

***Table 3-14** Attribute General 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 Default Value Properties

This value is used when an object is created via the Create identity portlet or through a workflow. You can express the default value as a literal or an ECMAScript expression. You cannot use a default value as part of a calculated attribute. If defined as an ECMAScript expression, it is resolved at runtime. If you define both the literal and an expression, the expression takes precedence.

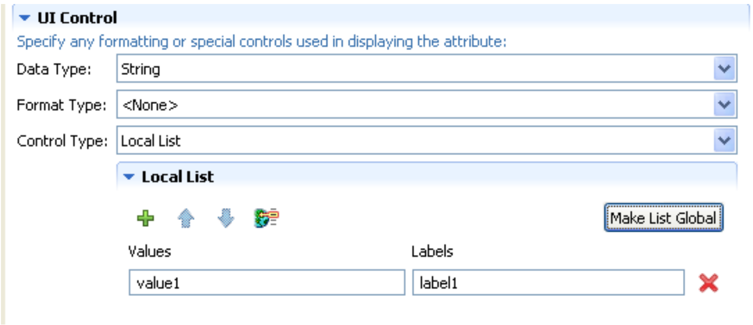
**TIP:** If you want the default value to be displayed by the Create portlet, you must define the access property viewable as True (selected). If you want the user to be able to change the value, you must set the editable property to True.

## Attribute UI Control Properties

***Table 3-15** 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>**NOTE:** Selecting the LocalizedString Data Type causes the search to be case-sensitive in the UI.<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>**DNLookup**: Defines that this attribute contains a DN reference. Use when you want to:<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 available to the form controls you can define for provisioning request and approval forms.<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>For more information on defining this property, see the "Attributes and DNLookup Properties" on page 68. For more information on the object selector dialog box for request and approval forms, see "Working with Object Selectors" on page 141.<br>**Global List**: Display this attribute as a drop-down list whose contents are defined in a file outside of this attribute definition. Click **Go to list** to access the Global List editor for the selected list.<br>For more information, see "Working with Lists" on page 54.<br>**Local List**: Display this attribute as a drop-down list whose contents are defined with this attribute. To define a local list:<br>1. With the attribute selected, set the control type to **Local List**.<br><br>UI Control<br>Specify any formatting or special controls used in displaying the attribute:<br>Data Type: String<br>Format Type: <None><br>Control Type: Local List<br>Local List<br>Make List Global<br>Values Labels<br>value1 label1<br>2. Use the buttons to add or remove list items. Use the up-arrow and down-arrow buttons to change the position of the item in the list.<br>In the **Value** column, type the value to write to the Identity Vault. It can include letters, numbers, and the underscore (_) character.<br>3. In the **Labels** column, type the text you want displayed in the user interface.<br>**Range**: Use the Range control type with Integer data types to restrict user input to a sequential range of values. Define the range's start and end values. |

## Attributes and DNLookup Properties

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

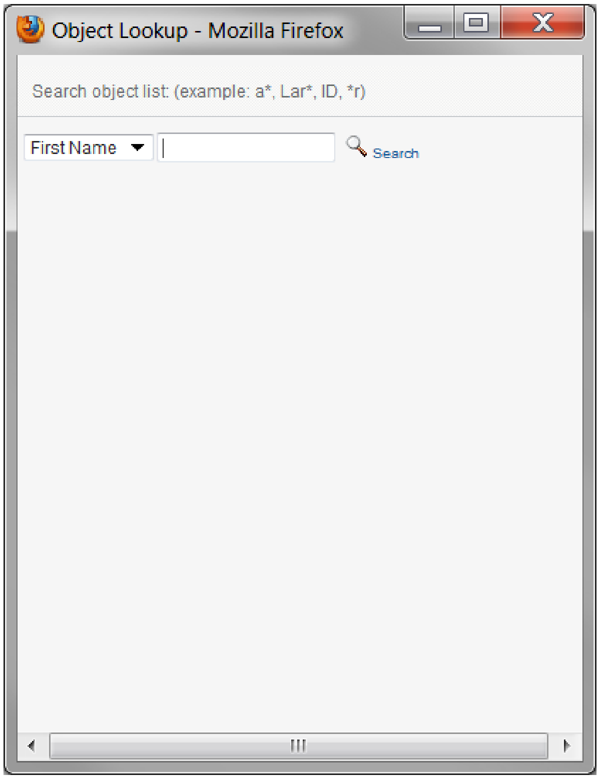
- This attribute can be used in an object selector dialog box that allows users to select from a list of possible values when searching on this attribute.
- When this attribute is created, populated, or deleted through the User Application, an attribute on a related entity is updated appropriately depending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

- "DNLookups for Object Selectors" on page 68
- "DNLookups for Referential Integrity" on page 70
- "DNLookup Property Reference" on page 71

### 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*

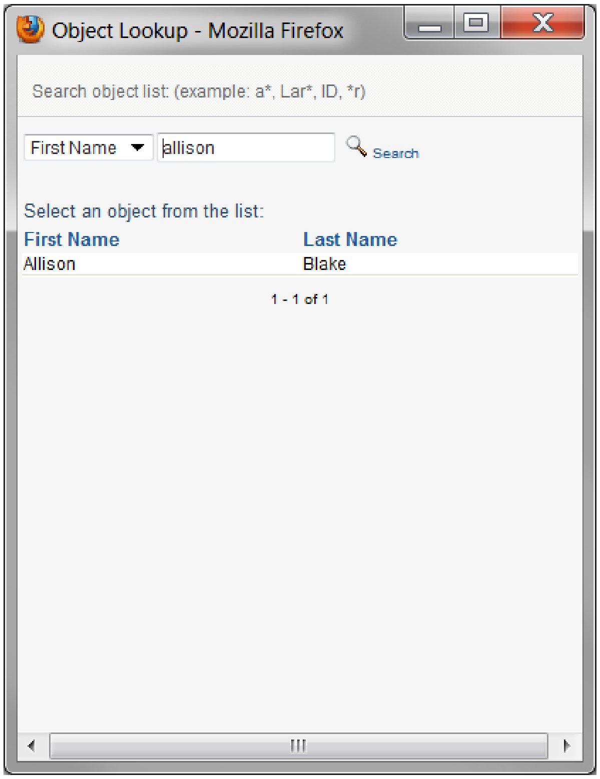


Object Lookup - Mozilla Firefox

Search object list: (example: a*, Lar*, ID, *r)

The result of the user's selection looks like this:

*Figure 3-3 Sample Object Selector Results*



Object Lookup - Mozilla Firefox

Search object list: (example: a*, Lar*, ID, *r)

First Name: allison

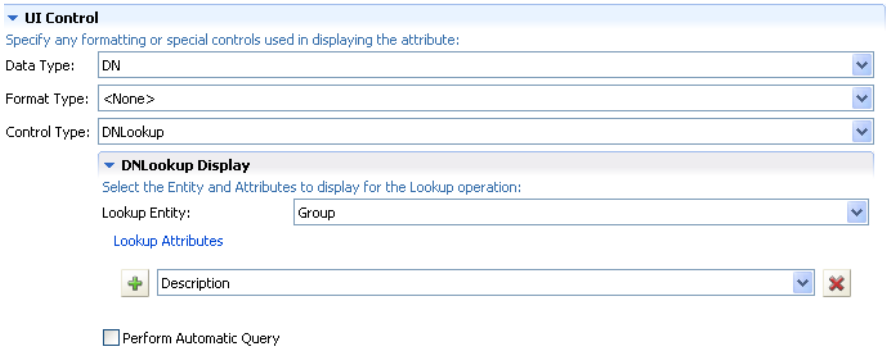
Select an object from the list:

| First Name | Last Name |
|---|---|
| Allison | Blake |

1 - 1 of 1

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*



**UI Control**

Specify any formatting or special controls used in displaying the attribute:

Data Type: DN

Format Type: <None>

Control Type: DNLookup

**DNLookup Display**

Select the Entity and Attributes to display for the Lookup operation:

Lookup Entity: Group

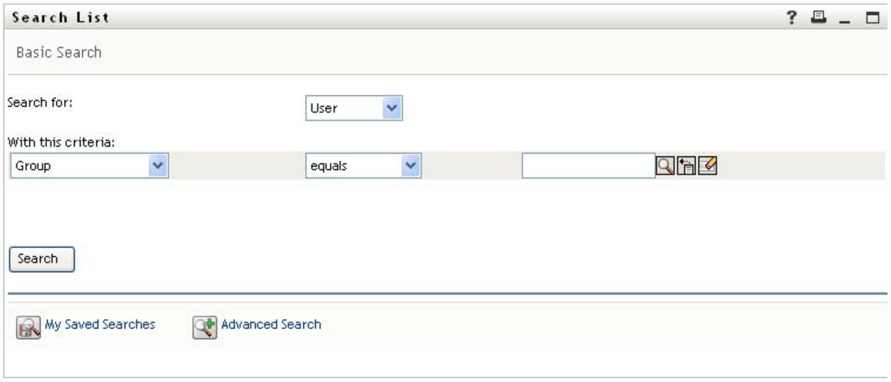
Lookup Attributes

Description

□ Perform Automatic Query

This definition also controls the way identity portlets provide a selection list 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 criteria, as follows:

***Figure 3-5*** *Search Criteria*



Search List

Basic Search

Search for: User

With this criteria: Group equals

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.

When the user picks a group, then he or she 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.

## DNLookup Property Reference

***Table 3-16** DNLookup Display Properties*

| Property Name | Description |
|---|---|
| Lookup Entity | The name of the entity to search. For example, suppose that the User entity contains an attribute for Manager. To populate that field, you'd need to know which users are 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 this option is 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 this option is deselected, allows the user to specify the search criteria for the entity query, then presents the results in a selectable list. |

***Table 3-17** 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 3-18** 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. |

# Queries Properties

This section describes the Queries properties.

- "Queries General Properties" on page 72
- "Query Parameters Properties" on page 72
- "Query Search Properties" on page 72

## Queries General Properties

*Table 3-19 Queries General Properties*

| Property Name | Description |
|---|---|
| **Key** | A unique value for the query key. This value is used in the Expression Builder to identify the query. The key is specified at the query creation time. It cannot be modified after the query is created. |
| **Query Entity** | Select an entity from the drop-down list box. The resulting LDAP search is on this entity. |
| **Display Label** | Type a string to display in the directory abstraction layer editor and Provisioning view. This value is not visible in the Expression Builder. |

## Query Parameters Properties

*Table 3-20 Queries Parameters Properties*

| Property Name | Description |
|---|---|
| **Parameter Keys** | A unique identifier for the key. You pass this key when calling the globalQuery() method on a form. |
| **Parameter Display Labels** | A label to identify the key. |

## Query Search Properties

If left blank, the query search properties default to the search properties specified for the selected entity. Specify the query search properties to further refine the search scope already defined for the entity. You cannot specify predefined parameters (for example, `%user-root%`) in the query's search properties.

*Table 3-21 Query Search Properties*

| Property Name | Description |
|---|---|
| Search Root | Specifies the location in the LDAP tree where the LDAP search defined by the query begins. |
| Search Scope | Specifies where the search occurs in relation to the search root. Values are:<br>**<Default>**: This search scope is the same as selecting **Containers and subcontainers**.<br>**Container**: The search occurs in the search root DN and all entries at the search root level.<br>**Container and subcontainers**: The search occurs in the search root DN and all subcontainers. This is the same as selecting **<Default>.**<br>**Object**: Limits the search to the object specified. This search is used to verify the existence of the specified object. |
| 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 it between 100 and 200 for greatest efficiency. Do not set it over 1000 |

## Relationship Properties

Relationship properties include:

- "Relationship Access Properties" on page 74
- "Relationship Properties" on page 74

## Relationship Access Properties

*Table 3-22 Relationship Access Properties*

| Property Name | Description |
|---|---|
| Used by Organizational Chart | When selected, this relationship can be used by the Org Chart portlet. |
| Used by Team-Management | **IMPORTANT:** This option is only available if you select **User** for both the **Source Entity** and **Target Entity** relationship properties and **This entity's key** for the **Source Attribute** relationship property. For more information about Relationship properties, see "Relationship Properties" on page 74.<br>When selected, this relationship can be used to define the provisioning members in iManager.<br>For example, if **Used by Management** is selected for the manager-employee relationship, then the provisioning application administrator can use this relationship to define the members as all users that report to the manager. |
| Enable Cascading Relationship | This option is only available if you select **Used by Team-Management**.<br>If **Enable Cascading Relationship** is selected, then the relationship can include several levels within the organization. You define the number of levels via **Maximum Levels to Cascade**. |

## Relationship Properties

*Table 3-23 Relationship Properties*

| Property Name | Description |
|---|---|
| Key | The read-only unique identifier for the relationship.<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>Click Localize to provide the translation for the display label text. |
| Source Entity | Choose an entity from the drop-down list.<br>The entity that you choose becomes the parent or source object in the organization chart hierarchy. In a Manager-Employee relationship, the Source Entity is User. For a Group-Member relationship, the source entity is Group.<br>Directory abstraction layer requirements: The entities in this list are a subset of the entities defined in the directory abstraction layer. Source entities must have the view access property selected (True). |

| Property Name | Description |
|---|---|
| Source Attribute | Choose an attribute from the drop-down list.<br>This attribute is used to find matching target entities. When the value of this attribute matches a corresponding value on an attribute of the target entity (see Target Attribute below), then a relationship can be established.<br>Directory abstraction layer requirements: This list of attributes is populated using the selected Source Entity’s attributes. It includes any attributes that are searchable and readable. |
| Target Entity | Choose an entity for the child object in the hierarchy. In a Manager-Employee relationship, it is user.<br>This entity must contain the attribute that is related to the Source attribute. |
| **Target Attribute** | Choose the attribute that matches the Source Attribute.<br>This is the target entity’s attribute used to find matching source entities. When the value of this attribute matches a corresponding value on the source entity (see Source 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 Source entity, but you can define a dynamic group as the target entity.

# 4 Configuring Provisioning Request Definitions

A provisioning request is a user or system action that initiates one or more provisioning workflows. These workflows can be used to grant or revoke resources or roles, or perform attestation processes. You use the provisioning request definition editor to create and deploy provisioning requests to the User Application driver. This section includes information about the provisioning request definitions shipped with the system and how to create new provisioning request definitions.

## About Provisioning Request Definitions

Provisioning request definitions are directory objects that encapsulate the business rules for granting or revoking a corporate resource or role, and binding the corporate resource or role to a workflow. Provisioning request definitions can also be used to launch attestation workflows. They are used in the User Application to support:

- Resource requests on the **Requests & Approvals** tab

  Resource requests allow users to request access to resources such as accounts, applications, servers, and so forth. Novell provides a read-only resource-oriented provisioning request definition named *Resource Approval*.

  For information about customizing the existing definition or writing your own resource based provisioning request definitions, see “Guidelines for Creating Resource Based Workflows” on page 177.
- Role assignment requests on the **Roles** tab.

  Role assignment requests allow users to request roles that grant them permissions to resources and not to the resources themselves. Novell provides these two read-only role-oriented provisioning request definitions:

  - *Role Approval*: Manages role requests.
  - *SoD Conflict Approval*: Manages role requests that result in Separation of Duties (SoD) conflict overrides.

  For information about customizing the existing definitions or writing your own roles based provisioning request definitions, see “Guidelines for Creating Roles Based Workflows” on page 168.
- Attestation process requests on the **Compliance** tab.

  Attestation process requests are used by Compliance Administrators and Attestation Officers to submit requests for attestation workflows. These workflows allow users to verify their own user profile information, to allow authorized users to verify the violations and exceptions to SoD constraints, or to verify role and user assignments.

  Designer provides these two attestation type provisioning request definitions:

  - *Attestation Report*: Manages the attestation process that allows users to verify the violations and exceptions for a set of SoD constraints.
  - *Attestation User Profile*: Manages the attestation process that allows users to confirm that their user profiles contain accurate information.

Attestation type provisioning request definitions are not editable within Designer. You cannot define or use custom provisioning request definitions for attestation, and they are not visible on the **Requests & Approval** tab.

You use Designer to define the trustees for the attestation process requests, to deploy the provisioning request definitions, and to localize the text users see during the approval process. For information on localizing attestation provisioning request definition text, see "Localizing Provisioning Objects" on page 39.

# Using the Provisioning Request Definition Editor

The provisioning request definition editor allows you to create provisioning request definitions that bind corporate resources or roles to a workflow. The provisioning request definition editor includes the following tools:

- **Overview tab:** Used to define the basic characteristics of the provisioning request.
- **Workflow tab:** Used to define the associated workflow by configuring activities and flow paths.
- **Form tab:** Used to define the request and approval forms that the user interacts with in the **Requests & Approvals** tab.
- **Signature Declarations tab:** Used to define the Digital Signature declarations.

For provisioning request definitions that are not based on roles, you can use a provisioning request template 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 own custom provisioning request definitions.

## Creating a Provisioning Request Definition

This section describes how to create both provisioning requests that are based on roles and provisioning requests that are not based on roles by using the following methods:

- From a template (not supported for roles based provisioning request definitions).
- From a copy of an existing provisioning request definition.
- As a custom provisioning request definition.

When possible, you should use a template or a copy of an existing definition because it saves you 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 custom provisioning request.

The following table describes the steps for defining a provisioning request.

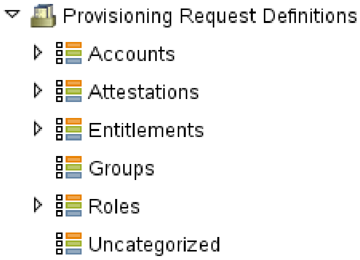
*Table 4-1 Basic Steps for Defining a Provisioning Request*

| Task | Action | For More Information |
|---|---|---|
| 1 | Create a provisioning request definition. | Depending on what you want to create, see:<br>◆ "Creating a Provisioning Request Definition By Using a Template" on page 80<br>◆ "Creating a Custom Provisioning Request Definition" on page 84.<br>◆ "Creating a Roles Based Provisioning Request Definition" on page 84<br>◆ "Modifying Settings of a Provisioning Request Definition" on page 86 |
| 2 | Create the request and approval forms.<br>**TIP:** Creating the forms before the workflow simplifies the process of mapping the form fields to the application data. | See Chapter 5, "Creating Forms for a Provisioning Request Definition," on page 95 |
| 3 | Create the workflow diagram by adding activities to the workflow diagram and connecting them with flow paths. | See Chapter 6, "Creating the Workflow for a Provisioning Request Definition," on page 149 |
| 4 | Configure the activities and flow paths by specifying the properties, data item mappings, and email notification settings for the activities. Then define the semantics for the flow paths. | See Chapter 7, "Workflow Activity Reference," on page 187 |

## Starting the Provisioning Request Definition Editor

**1** Open the Provisioning view and double-click the **Provisioning Request Definitions** node.

Designer displays the provisioning request definitions tree containing nodes for the default provisioning definition request definition categories of **Accounts, Attestations, Entitlements, Groups, Roles,** and **Uncategorized**.


- Provisioning Request Definitions
  - Accounts
  - Attestations
  - Entitlements
  - Groups
  - Roles
  - Uncategorized

These default categories are defined by the directory abstraction layer Provisioning Category list. For information on managing categories, see "Working with Lists" on page 54.

The installed templates are available in the Entitlements node, and the default role provisioning request definitions are available in the Roles node.

2 Use the Provisioning view pane to navigate the provisioning request definition categories. The right-click menu is available from the top-level node or when you select an existing provisioning request definition. You cannot create a provisioning request definition by selecting a category node.

3 Double-click a provisioning request definition to open it in the editor in the right pane.

| For details on using... | See |
|---|---|
| **Overview** tab | "Modifying Settings of a Provisioning Request Definition" on page 86. |
| **Workflow** tab | Chapter 6, "Creating the Workflow for a Provisioning Request Definition," on page 149. |
| **Forms** tab | Chapter 5, "Creating Forms for a Provisioning Request Definition," on page 95. |

Provisioning request definitions are stored locally in the `Provisioning\AppConfig\RequestDefs` directory within your workspace.

## Creating a Provisioning Request Definition By Using a Template

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 > Provisioning Request Definition**. 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 |
|---|---|
| **Identifier (CN)** | The CN (common name) identifier for the provisioning request definition. The name cannot be longer than 64 characters. |
| **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.

4 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**.

You use the next panel of the wizard to specify the provisioning request definition's category and trustees (the users, groups, or containers) who can access the provisioning request definition after it is deployed.

5 Select a category from the list.

6 Select **Notify participants by e-mail** if you want approvers to be notified by email about pending approval tasks, and also want initiators to be notified by email of workflow completion. If **Notify participants by email** is not selected, users must look at the **Task Notifications** list on the **Work Dashboard** tab of the User Application for notifications about tasks.

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

Designer displays a panel that allows you to browse the Identity Vault to select a trustee. You can select a user, group, or container. If you cannot connect to the Identity Vault, you can type trustee DNs directly in the **Trustee DN** field.

8 Select the trustee, then click **OK**.

Designer returns you to the previous panel. If desired, add additional trustees by repeating the previous step.

9 When you have finished adding trustees, click **Finish**.

Designer displays the Provisioning Request Definition Details panel on the **Overview** tab (see "Modifying Settings of a Provisioning Request Definition" on page 86).

10 Click the **Workflow** tab. The Workflow view is displayed.

The provisioning request definition template includes some default values that you must 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.

11 Click the Entitlement Provisioning activity, then click the **Data Item Mapping** tab.

12 Double-click in the **Source Expression** field to display 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 271 for information about the ECMA Expression Builder.

13 Use the ECMA Expression Builder to replace the placeholder expression with an expression that specifies the entitlement that you want to provision with this provisioning request.

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

15 Click the **Forms** tab and customize the forms for the provisioning request to meet your needs.

See Chapter 5, "Creating Forms for a Provisioning Request Definition," on page 95.The template includes predefined request and approval forms. You might want to add additional forms, or add or remove form controls.

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

See Chapter 6, "Creating the Workflow for a Provisioning Request Definition," on page 149 and Chapter 7, "Workflow Activity Reference," on page 187.

## About 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 from zero to 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 if the workflow times out during the course of processing.

The following table lists the templates included with Identity Manager.

***Table 4-2** 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 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 the next activity.<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 the next activity.<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 the next activity.<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 the next activity.<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>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>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>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>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 the next activity.<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 the next activity.<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 the next activity.<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 the next activity.<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>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>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>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>This template supports parallel processing. |

### About the Installed Templates and Flow Strategy

By default, provisioning request definition templates use the **Single Flow** flow strategy. The templates assume that the recipient is a user DN. If you change the flow strategy, you must modify the template. If you change the flow strategy to:

- **Flow per member**: Remove the recipient reference from the request form.
- **Single flow provision members**: Remove the recipient from the request form and add logic to determine the addressee for the approval activity. The templates assume the recipient is a user DN, so you must determine if the recipient is a user or group DN. You can use the IDVault isGroup(String dn) method to determine if the DN is a group. If the recipient is a group DN, you must provide logic for assigning the approval activity addressee.

## Creating a Custom Provisioning Request Definition

If no existing provisioning request definition resembles the new work that you want to do, then you need to build a custom provisioning request definition.You can still save time and effort by re-using forms from other workflows.

**NOTE:** The procedure in this section does not use roles. To create a custom roles based provisioning request definition, see "Creating a Roles Based Provisioning Request Definition" on page 84

1. Create the basic information for a new provisioning request definition (see "Creating a Provisioning Request Definition By Using a Template" on page 80). In step Step 4 on page 80, 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. Defining forms before you create the workflow topology ensures that data bindings can be set up automatically for each activity when you create activities.

   To create the forms, see "Creating Forms" on page 97.
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 6, "Creating the Workflow for a Provisioning Request Definition," on page 149.
4. Specify the details (properties, data item mappings, email notification) for each workflow activity.

   To specify workflow activity details, see Chapter 7, "Workflow Activity Reference," on page 187.
5. Configure the flowpaths between workflow activities.

   To configure flowpaths, see "Configuring Flow Paths" on page 158.

## Creating a Roles Based Provisioning Request Definition

Designer supplies two Roles Based provisioning request definitions that you should use as a basis for your custom roles based provisioning request definitions. They are:

- Role Approval
- SoD Conflict Approval

To create a copy and customize its contents:

1 From the Provisioning view, open **Roles** in the **Provisioning Request Definitions** node.

2 Select one of the roles-based provisioning request definition (depending on which type of approval you want to create), right-click, then select **Create From.**

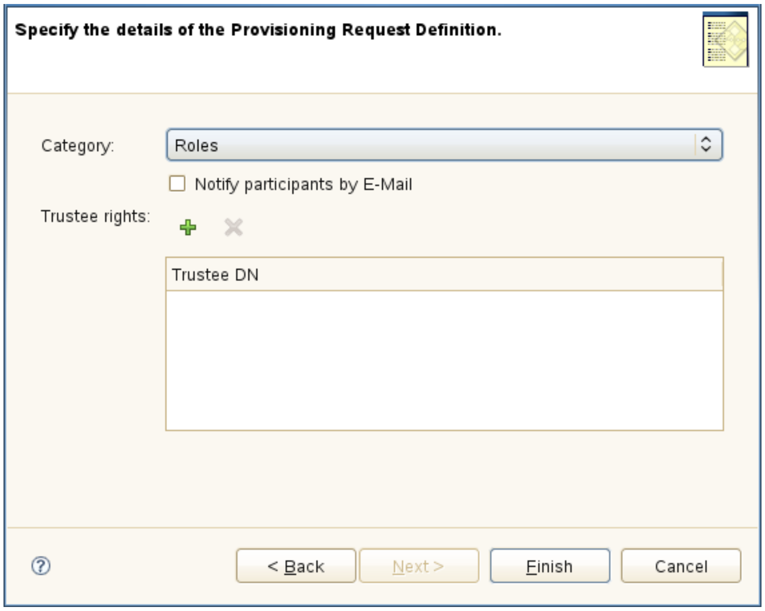
- **Role Assignment/Revocation Approval**: Choose this for role requests.
- **SoD Conflict Approval**: Choose this for SoD conflict approval requests.

Designer displays the **Create a New PRD** Wizard.

3 Fill in the fields as follows:

| **Field** | **Description** |
|---|---|
| **Identifier (CN)** | The CN (common name) identifier for the provisioning request definition. The name cannot be longer than 64 characters. |
| **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. |

4 Click **Next**. Designer displays the following dialog box.



**Specify the details of the Provisioning Request Definition.**

Category: Roles

☐ Notify participants by E-Mail

Trustee rights:

| Trustee DN |
|---|

5 Specify **Roles** for the category.

6 Select **Notify participants by e-mail** if you want approvers to be notified by email about pending approval tasks, and also want initiators to be notified by email of workflow completion. If **Notify participants by e-mail** is not selected, users must look at the **Roles** tab in the User Application for notifications about tasks.

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

Designer displays a panel that allows you to browse the Identity Vault to select a trustee. You can select an individual trustee or a group.If you cannot connect to the Identity Vault, you can type trustee DNs directly in the **Trustee DN** field.

8 (Optional) Select the trustee, then click **OK**.

Designer returns you to the previous panel. Add additional trustees by repeating the previous step.

9 Click **Finish**.

Designer displays the Provisioning Request Definition Details panel on the **Overview** tab (see "Modifying Settings of a Provisioning Request Definition" on page 86).

For more details on defining the associated workflow, see "Guidelines for Creating Roles Based Workflows" on page 168.

# Modifying Settings of a Provisioning Request Definition

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

## Modifying Basic Settings

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

*Table 4-3 Overview Properties*

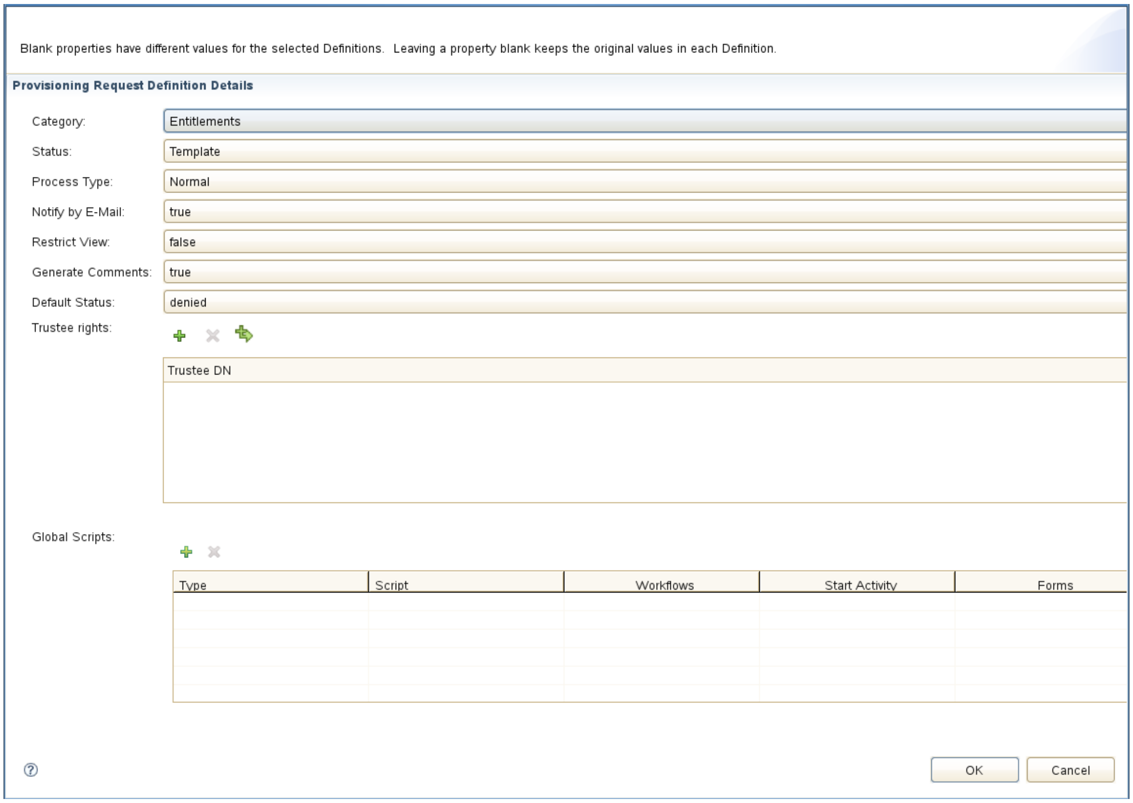
| Field | Description |
|---|---|
| **Identifier (CN)** | Displays the CN (common name) of the provisioning request definition. The CN cannot be changed. |
| **Display Name** | Specifies the display name of the provisioning request definition. This is the name that is displayed to the user in Designer 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. The Provisioning view displays the provisioning request definitions by categories. |
| **Status** | Specifies the status of the provisioning request definition:<br>**Active:** Select this option to make the provisioning request definition available for use in the User Application.<br>**Inactive:** Select this option 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 the provisioning request definition separate from the person who activates the provisioning request definition. For example, a developer could be responsible for marking the provisioning request definition as Inactive, and an administrator could be responsible for changing the status to Active.<br>**Template:** Select this option 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>**Retired:** Select this option 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. |

| Field | Description |
|---|---|
| **Flow Strategy** | Specifies the flow strategy for the provisioning request definition:<br>**Single Flow:** This strategy allows one workflow with one recipient.<br>**Flow per member:** This strategy allows the recipient to be a group DN. If you select this strategy, the User Application starts a workflow instance for each member of the group, and each workflow can be approved or denied separately. For example, assume there is a provisioning request definition for the recipient Human Resources. The Human Resource group has the members ablake and kchester. The User Application passes the Human Resources DN to the provisioning start. The provisioning interface starts two workflow instances, one for ablake and one for kchester.<br>**Single Flow Provision Members:** This strategy allows the recipient to be a group DN. If you select this strategy, the User Application starts a single workflow for the group. The workflow spawns multiple provisioning steps (one for each member) within the single workflow. The workflow is approved or denied to the group as a whole, not for each individual member. |
| **Process Type** | Specifies the type of provisioning request definition. Values are:<br>**Normal:** Used for typical workflow definitions that are not related to roles.<br>**Role Approval:** Specify this type if the workflow is used for roles approvals. When this is set, the workflow is available to the Roles Configuration editor and the SoD Editor. It ensures that the NrfRequest object data item is available in the data item mapping.<br>**Resource Approval:** Specify this type if the workflow is used for resource approvals. When this is set, the workflow is available to the Resource Editor. It ensures that the NrfResourceRequest object data item is available in the data item mapping.<br>**Attestation:** Used by Compliance Administrators and Attestation Officers to submit requests for attestation workflows.<br>**WARNING**<br>◆ If you change the process type from **Role Approval** or **Resource Approval** to **Normal**, you must also remove any ECMA expressions related to the NrfRequest object. ECMA expressions related to the NrfRequest object are only valid in role based workflows and resource based workflows. Including these expressions in non-roles-related workflows or non-resource-related workflows cause runtime errors in the User Application.<br>◆ The driver must be Roles Based Provisioning 3.7 or higher. |
| **Notify participants by E-Mail** | Specifies whether approvers are notified by email about pending approval tasks, and whether initiators are notified by email of workflow completion. If **Notify participants by email** is not selected, then users must look at the Task Notifications pane of the **Work Dashboard** tab in the User Application for notifications about tasks.<br>For information about selecting an email template and customizing email template parameters, see “Finish Activity” on page 220. |
| **Restrict View** | Specifies that the list of tasks that can be viewed by the user in My Requests in the User Application is restricted to tasks initiated by the user. The default behavior (that is, **Restrict View** is not selected) is that the user can see any requests that the user initiated or for which the user is the recipient. |

| Field | Description |
|---|---|
| **Generate Comments** | Specifies that the workflow engine should generate comments as the workflow progresses. These comments can be displayed by clicking the following:<br>◆ **View Comment and Flow History** in a **Request Detail** form in the User Application<br>◆ **View Comment History** in a **Task Detail** form in the User Application |
| **Set Default Completion Status to Approved** | Specifies that the default completion status of the provisioning request is approved if selected, or denied if not selected. This feature can be useful for provisioning requests that do not contain a provisioning activity (an Entitlement or Entity). The value of this parameter can be overridden by explicitly setting the completion status by using a Provisioning activity or Workflow Status activity. |
| **Trustee rights** | Specifies the users and groups that can use the provisioning request definition. |
| **Global Scripts** | Specifies the global ECMA scripts that provisioning request definition imports while working with workflows and forms. Inline or extenal scripts can be added by specifying the type of the script. You can also specify when these scripts should be available to you. These scripts can be available on workflows, start activity, or forms. |

Designer allows you to modify the properties of more than one provisioning request definition at a time. The provisioning request definitions must be of the same category. You will see a different set of properties for provisioning request definitions whose categories are Attestations or Roles. See "Modifying Properties of Attestation or Roles Based Provisioning Request Definitions" on page 89.

1 With the Provisioning view open, select one or more provisioning request definitions, right-click, then select **Properties**. A dialog box like the following displays (the properties you can modify depend on the type of provisioning request definitions you have selected):



Blank properties have different values for the selected Definitions. Leaving a property blank keeps the original values in each Definition.

**Provisioning Request Definition Details**

Category: Entitlements
Status: Template
Process Type: Normal
Notify by E-Mail: true
Restrict View: false
Generate Comments: true
Default Status: denied
Trustee rights:

| Trustee DN |
|---|

Global Scripts:

| Type | Script | Workflows | Start Activity | Forms |
|---|---|---|---|---|

2 Modify the selected provisioning request definitions as desired by changing the value for that property.

If you have selected multiple provisioning request definitions and one of the properties does not contain a value, the corresponding field is blank. If you modify that field, then the value you specify is applied to all of the selected provisioning request definitions.

3 For Trustees, delete, add, or merge the values for the selected provisioning request definitions.

A merge adds all of the trustees in the selected provisioning request definitions to all of the selected provisioning request definitions.

4 For Global Scripts, delete, add, or modify the values for the selected provisioning request definitions.

- **External:** The script is incorporated into the provisioning request definition by reference using the supplied ECMA Script DN.
- **Inline:** The script is inserted directly into the provisioning request definition in a `<script>` block. You add your JavaScript using this ECMAScript Editor. To learn more about using the editor, click the editor's help button. For inline scripts, the following is inserted in the page:

```
<script>whatever you type</script>
```

5 Click **OK** to save your changes.

## Modifying Properties of Attestation or Roles Based Provisioning Request Definitions

The provisioning request definitions that support Attestations and Roles cannot be opened in the Provisioning Request Definition editor so you cannot set their properties by using the **Overview** tab. You can set several important properties, such as Trustee Rights, before deploying them.

1 With the Provisioning view open, select one (or more) of the Roles or Attestation provisioning request definitions, right-click, then select **Properties**.

2 Fill in the fields as follows:

| Field Name | Description |
|---|---|
| **Notify by E-Mail** | Specifies whether approvers are notified by email about pending approval tasks, and whether initiators are notified by email of workflow completion. If **Notify participants by email** is not selected, then users must look at the Task Notifications pane of the **Work Dashboard** tab in the User Application for notifications about tasks.<br>For information about selecting an email template and customizing email template parameters, see "Finish Activity" on page 220. |
| **Trustee Rights** | Specifies the users and groups that can use the provisioning request definition. |
| **Global Scripts** | Specifies the global ECMA scripts that provisioning request definition imports while working with workflows and forms. Inline or external scripts can be added by specifying the type of the script. You can also specify when these scripts should be available to you. These scripts can be available on workflows, start activity, or forms. |

3 For Trustees, delete, add, or merge the values for the selected provisioning request definitions.

A merge adds all of the trustees in the selected provisioning request definitions to all of the selected provisioning request definitions.

4 For Global Scripts, delete, add, or modify the values for the selected provisioning request definitions.

5 Click **OK** to save your changes.

# Provisioning and Workflow Example

This section describes a common provisioning and workflow scenario. It is designed to help you understand how the different objects that you create with the provisioning request editor are used by the User Application.

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.

## 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 selected provisioned resource 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.

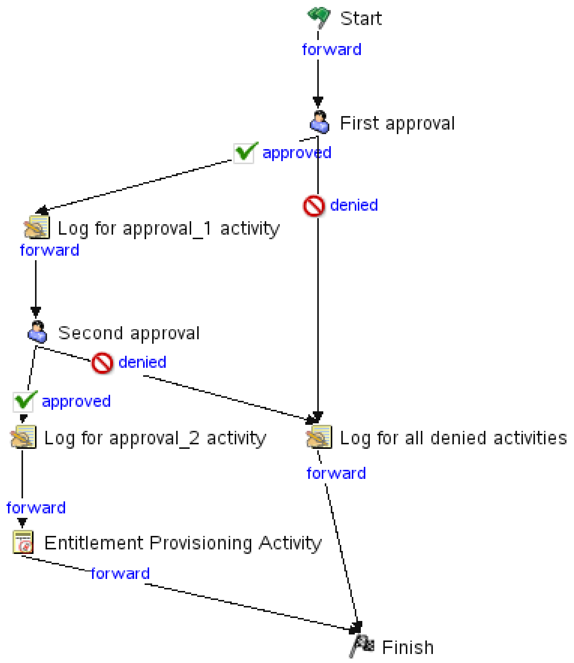
Each provisioning request has an operation associated with it. The operation specifies whether the user wants to grant or revoke the resource.

## Step 2: Approving the Request

After the user has initiated the request, the Provisioning System starts the workflow process, which 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 sequentially or in parallel. In a sequential workflow, as shown in the following figure, each approval task must be processed before the next approval task begins.

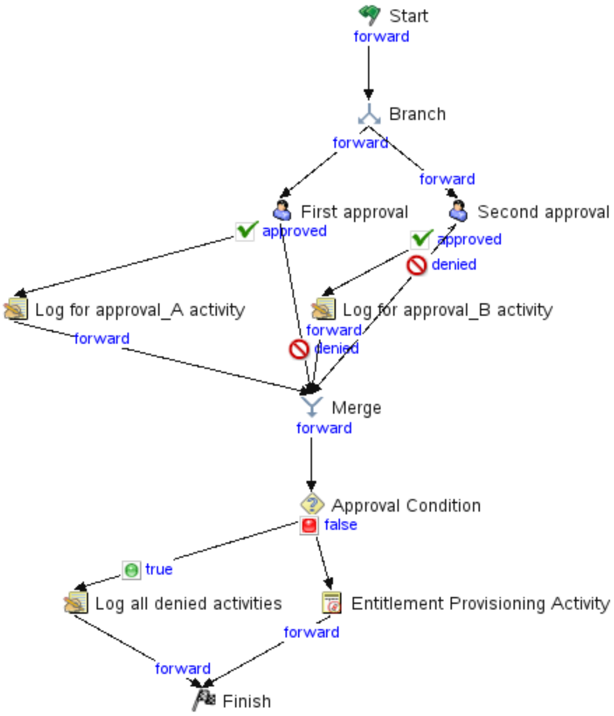
***Figure 4-1*** *Sequential Workflow with Two Approvals*



Start
forward
First approval
approved
denied
Log for approval_1 activity
forward
Second approval
denied
approved
Log for approval_2 activity
Log for all denied activities
forward
forward
Entitlement Provisioning Activity
forward
Finish

In a parallel workflow, as shown in the following figure, users can work on the approval tasks simultaneously.

***Figure 4-2*** *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 the following table:

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

| Process Component | Description |
|---|---|
| Activities | An activity is an object that represents a task. An activity can present information to the user and respond to user interactions. It can also perform background functions that are not visible to the user.<br>In a workflow diagram, the activities are represented by boxes.<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>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>For more information on the **My Tasks** and **My Requests** actions, see the *Identity Manager 3.6.1 User Application: User Guide*. |
| Flow paths | Flow paths tie the activities in a workflow together. A flow path represents a path to be followed between two activities.<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>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. After 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, the Workflow System forwards processing to the first Approval activity in the flow. The Approval activity sends an email to the approver, notifying this user that his or her 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 in "Step 2: Approving the Request" on page 90, "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 email. To perform the work associated with the activity, the addressee can click the URL in the email, 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; otherwise, 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 escalation count specified for the activity. In some situations, the workflow process might be configured to escalate an activity that has timed out. 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.

**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. After the users have completed their work, the Merge activity synchronizes the incoming branches in the flow.

**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 in "Step 2: Approving the Request" on page 90, "Approval Condition" is an example of a Condition activity.

The Condition activity supports three possible outcomes or exit paths:

- True
- False
- Error

## 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 Identity Vault object and its attributes. These actions are performed by either the Entitlement activity or the Entity activity:

- **Entitlement activity:** Fulfills the provisioning request by granting or revoking an entitlement. This activity is not usually executed unless all of the necessary approvals are given.
- **Entity activity:** Fulfills the provisioning request by directly manipulating an eDirectory object and its attributes. This activity is not normally executed unless all of the necessary approvals are given.

## Step 4: Completing the Workflow

When all other activities have terminated, the workflow executes the Finish activity, which 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 executes. The Finish activity sends a final email notification to inform participants of the completion of the workflow.

# 5 Creating Forms for a Provisioning Request Definition

This section provides details on creating and customizing the User Application's request and approval forms.

## About Forms

Forms allow the user to request a resource, approve a resource request, and work on a task. They are available when the user chooses any of the actions in the Task Notifications pane of the User Application's **Work Dashboard** tab.

Figure 5-1 on page 100 is an example of a resource request form. At the top of the form is a read-only area that displays the details of the request (or approval for approval forms). In the **Form Detail** section at the bottom, the user provides input to the resource request (or approval) and takes some action on it.

You use the **Forms** tab of the provisioning request definition editor to define the appearance and behavior of the **Form Detail** section of the User Application's requests and approvals forms.

### 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.

### About Approval Forms

You can define multiple approval forms for a provisioning request definition, but only one form per Approval Activity. You can specify the approval form to associate with an approval activity in the properties for the activity. You can create an approval form via the **Forms** tab or from the approval activities property sheet.

## 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 workflow document called flowdata. These data-item bindings, and any script expressions they use, execute on the application server as preparation of the form before it is sent to the client browser for display to the user. Common uses for pre-activity data-item mappings and their expressions that operate against the flow-data document are for moving previous approval data into the current approval or for setting default values for fields. For more information on data item mappings, see "Defining the Data Item Mappings" on page 156.

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 "Form Control Reference" on page 103.

## About Forms and Events

Designer allows you to define action scripts that execute on the form control’s onLoad, onChange, or custom events. Each form control supports an Events property where you supply the script for the event. The scripts you define have an event-level scope and execute in the browser of the user’s client machine.

The Events property provides access to Designer’s Event Action Expression Builder, which allows you to create script expressions that refer to and modify form and data. Because form control event scripts execute in the client browser, they do not have access to the flow-data document. They do have access to directory abstraction layer queries.

The Event Action Expression Builder also provides access to the Form Action methods (shown in the left column). This column provides access to the form action script API along with directory abstraction layer query objects. The form action script API is written in JavaScript so that you can add conditions, loops, and user-defined functions. For more information about the Form Action API, see “Form Action Script Methods” on page 280. To import or include a JavaScript library, you use the **Scripts** tab of the Form Controls area. For more information, see “Using the Scripts Tab” on page 100.

# 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** tab contains a **Form Selection** section and a **Form Controls** section.

## About Form Selection

Use the **Form Selection** section to create, delete, or preview a form, or to create a form template. Click the **Request** or the **Approval** tab depending on the type of form you want to manipulate.

The **Form Selection** toolbar contains these options:

***Table 5-1*** *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. |

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

## About Form Controls

Use the **Form Controls** section to define or modify the form's appearance and behavior.

- **Fields** tab: Lets you add, delete, and change the data type, control type, and layout order of the controls on the form.

  For information about adding controls, see "Creating Forms" on page 97. For more information about individual form controls, see "Form Control Reference" on page 103.
- **Actions** tab: Lets you define the actions the user can perform on the form. Use the **Actions** toolbar to add, delete, and change the actions and layout order of the actions on the form.

  For more information about the supported actions, see "Action Reference" on page 102.
- **Scripts** tab: Use the **Scripts** tab to define calls to external JavaScript files or to write JavaScript scripts that are stored as part of the form definition. When you have created a script by using this tab, it becomes available in the **Action Script Expression Builder** and you can call it from any form control event. These scripts have a page-scope rather than an event-scope. For more information about using the script tab, see "Using the Scripts Tab" on page 100.
- **Events** tab

# Creating Forms

This section describes how to create new forms and add controls to them.

## Creating New Forms

1. With the 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 click **Cancel** to exit without saving.

## Adding Form Controls and Actions

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

**NOTE:** The Designer places form controls on the form from top to bottom and left to right. Use Linebreaks to force spacing between controls

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 several different locations:<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, in order 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 visual control used to display or edit the data. The selection list is filtered based on the selected data type. |
| **Linebreaks** | Defines the number of lines you want inserted after the control. |

Form field controls do not have Data Item Mappings or email notifications property sheets.

3 For each control, specify its properties in the **Properties** tab (available via **Window > Show View > Properties**). For more information, see "Form Control Reference" on page 103.

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.

A request form must have, at a minimum, a SubmitAction. Without a SubmitAction, the request does not process. Every form should 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>**Bottom**. Places the action buttons on the bottom of the form. (Default.)<br>**Top**: Places the action buttons on the top of the form.<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 "Action Reference" on page 102. |
| **Linebreaks** | Defines the number of lines you want inserted after the action button. |

6 Save the form.

## Defining Events

The scripts you attach to an event handler are scoped to the appropriate control, not the browser window.

### Defining an Event

1 Select the form control where you want to define an event and open the property sheet.

2 Navigate to the Event tab and add an event. Designer adds a row with the default event name **onload**.

3 Click the **Event Name** field and select the **onchange** or **onload** event. For more information on adding other events, see "Creating Custom Events" on page 99.

4 Click the **Action Expression** field. You can type the script directly in this field, or click the button to access the **Event Action Expression Builder**.

5 Define the action script, check the syntax, then click **OK**. Repeat this procedure to add more events to this control.

For more information on the onchange and onload events, see the events property description in "General Form Control Properties" on page 105.

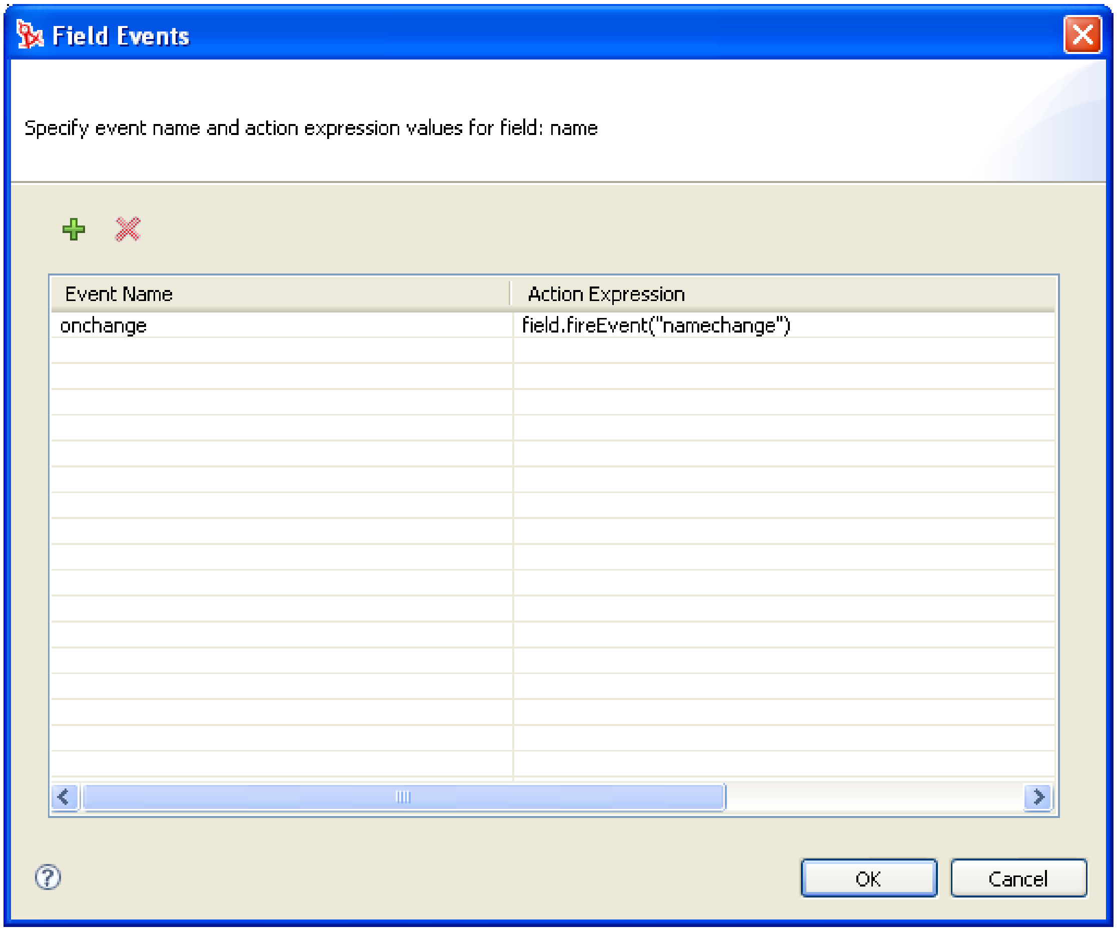
### Creating Custom Events

You can create your own events to notify other controls of conditions or user actions on the form. You create the event using the **Events** property. You can give the event any name. You must explicitly fire the event by using the fireEvent() method and passing in the name of the event.

You might want to perform a query on the Groups container that returns only the groups that match the values entered by a user. In the example shown in Figure 5-7, the user types a value in the name field, When the user tabs to the next field, the contents of the drop-down list are populated from a query launched by the **namechange** custom event.

The **Name** field defines an **Events** property that fires the **namechange** event on an **onchange** event. The definition is shown in Figure 5-1.

*Figure 5-1 Sample field.FireEvent() Method*



Specify event name and action expression values for field: name

| Event Name | Action Expression |
|---|---|
| onchange | field.fireEvent("namechange") |

The **namechange** event contains an expression that executes a query called **groups**.

For more information on using queries, see "Using DAL Queries in Forms" on page 143.

## Using the Scripts Tab

Use the **Scripts** tab to define a script that has a page-level scope. A page-level scope means that the script loads at page load time and is available through the life of the form. You can supply the script in one of the ways described in Table 5-2.

*Table 5-2 Script Types*

| Script Type | Description |
|---|---|
| **external** | The script is incorporated into the page by reference, using the supplied URL. The script block will look something like this: `<script type="test/javascript" scr="http://some.server/custom.js"></script>`. The `custom.js` file is imported at form load. |
| **inline** | The script is inserted directly into the form in a `<script>` block. |

Because these scripts are loaded at page load, the form controls and any of their associated event handler scripts are not in scope when the page is loaded. Avoid coding dependencies between page-level scripts and event-level scripts; however, you can call page-level scripts from within an event-level script.

To add a link to an external JavaScript file:

1. With the Scripts tab open, click **Add** +.
2. Complete the fields as follows:

| Field | Description |
|---|---|
| **ID** | Specify a meaningful name. This value displays in the Event Action Expression Builder. |
| **Type** | Select **external**. |
| **URL/Inline Script** | Click within the field so that the **ECMAScript Editor** button displays to the right, then click the button to display the editor as a dialog box.<br>Type the URL to the `.js` file in the **Enter the URL String** field, then click **Retrieve**. The script is fetched and displays in read-only mode. You can inspect the script, but you cannot change it.<br>When you add an external link to a form, only the link is stored and deployed, for example:<br>`<script src="`*`someURL.com/script.js`*`"/>`<br>**TIP:** Designer cannot validate this external reference. You must ensure that it is accessible at runtime. |

To create an inline script:

1. With the **Scripts** tab open, click **Add** +.
2. Complete the fields as follows:

| Field | Description |
|---|---|
| **ID** | Specify a meaningful name. This value displays in the Event Action Expression Builder. |
| **Type** | Select **inline**. |
| **URL/Inline Script** | Click within the field so that the **ECMAScript Editor** button displays to the right, then click the button to display the editor as a dialog box.<br>You add your JavaScript by using this ECMAScript Editor. To learn more about using the editor, click the editor's help button.<br>For inline scripts, the following is inserted in the page:<br>`<script>whatever you type</script>` |

Both inline and external scripts are executed at page load but before the page loads the controls. In addition, they are also executed when specifically called on a form control event.

## 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 5-3** Valid Actions*

| Action Name | Form Type | Description |
|---|---|---|
| **ApprovalAction** | Approval | Causes the Approval activity to complete and follow the **approved** flow path to the next activity. When you use this action, you must set the Hide If Read Only form property to True; otherwise the form fails validation when you submit it.<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 | For request forms, Cancel returns the user to the Request Resource Search Criteria form. For approval forms, Cancel returns the user to the My Tasks list. |
| **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.<br>Comments are updated and persisted to the workflow database through the UpdateAction.<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; otherwise, the form fails validation when you submit it.<br>**TIP:** A DenyAction requires the Approval Activity associated with the form to have a 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; otherwise, the form fails validation on submit.<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. There is typically a text area associated with an **apwaComment** form field. 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).<br>**NOTE:** The form must contain a field named `apwaComment`; otherwise, the provisioning request definition fails validation.<br>For more information about `apwaComment`, see "Controls for User-Entered Comments" on page 105. |

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

***Table 5-4** 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. You should not set it to False if the action button submits data; otherwise, invalid data can be submitted, causing unexpected results.<br>Designer does not allow you to set this property to False for the ApprovalAction, DenyAction, or RefusalAction. |
| **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 encounter a validation error and cannot deploy the form. |

## Form Control Reference

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

***Table 5-5** Control Types and Supported Data Types*

| Control Type | Data Types | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **Boolean** | **Date** | **Decimal** | **DN** | **Integer** | **String** | **Time** | **Resource Request** |
| CheckBoxPickList | x | | x | x | x | x | | |
| DatePicker | | x | | | | | | |
| DateTimePicker | | | | | | | x | |
| DNContainer | | | | x | | | | |
| DNDisplay | | | | x | | | | |

| Control Type | Data Types | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Boolean | Date | Decimal | DN | Integer | String | Time | Resource Request |
| DNLookup | | | | x | | | | |
| DNMaker | | | | x | | | | |
| DNQuery | | | | x | | | | |
| Global List | | | | | | x | | |
| Html | | | | | | x | | |
| Localized Label | | | | | | x | | |
| MVCheckbox | | | x | x | x | x | | |
| MVEditor | | | x | x | x | x | | |
| Password | | | | | | x | | |
| PickList | | | x | x | x | x | | |
| RadioButtons | x | | | | | | | |
| Static List | x | | x | | x | x | | |
| Text | | | x | x | x | x | | |
| Text Area | | | | | | x | | |
| Title | | | | | | x | | |
| TrueFalseCheckBox | x | | | | | | | |
| TrueFalseRadioButtons | x | | | | | | | |
| TrueFalseSelectBox | x | | | | | | | |

**NOTE:** Note that the session time-out value is not increased when specifying a value for a form control type that does not communicate with the User Application back-end.

## Data Type for Roles Based Request Forms

Designer supports a specialized form control called nrfRequestDN of data type Role Request. The control type is Text. It is defined by default when you create a copy of the standard roles based provisioning request definitions. It represents the Role Request object.

## Data Type for Resource Based Request Forms

Designer supports a specialized form control called nrfResourceRequestDN of data type Resource Request. The control type is Text. It is defined by default when you create a copy of the standard resource based provisioning request definitions. It represents the Resource Request object. You can add this control only once. nrfResourceRequestDN is the field name and it cannot be changed in the GUI.

## 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. Comments 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 "Action Reference" on page 102.

## General Form Control Properties

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

***Table 5-6*** *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.<br>**NOTE:** If this property is set to false, it remains read-only and any change to other properties of this filed cannot change the property value using ECMA scripting.<br>For example, in the `onload` event if you try to enable or disable the field like `form.enable(Name)`, it cannot change the property value of the **Name** field when the editable property is set to false using designer. |
| **Events** | Specifies an event for the control. Possible values include the following:<br>◆ OnChange: Fires when one of the following occurs:<br>◆ Immediately after onload.<br>◆ Another script changes the value of the control.<br>◆ The user commits a change to the data value associated with the control. This occurs when the user has tabbed out of the control or otherwise caused it to lose focus. For example, this can happen when the user tabs away from the control (for text entry based controls like Text, TextArea, DatePicker), or when the user selects a different entry choice for choice-based controls like PickList, MVCheckbox, and StaticList).<br>◆ onLoad: The onload event for a control fires just once, when the control is loaded into the page for the first time. It can be used to set initial values or preselect entries; however, there is no guarantee that controls load in a particular order. |

| Property Name | Description |
|---|---|
| **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.

# CheckBoxPickList

Use the CheckBoxPickList control to allow users to view and choose one or more values from a dynamically generated list of choices displayed as check boxes.

When the associated data type is a DN retrieved from the Identity Vault, you can display the check box label as the fully qualified DN or use the Display expression property to specify the attributes to display instead.

***Figure 5-2*** *Sample CheckBoxPickList Control*

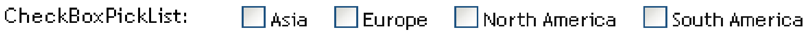


***Table 5-7*** *CheckboxPickList Properties*

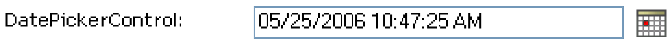
| Property Name | Description |
|---|---|
| **Entity Key for DN expression lookup** | When you populate this control with a DN retrieved from the Identity Vault and you want that value to display in a user-friendly fashion, you should choose an entity from the drop-down list and specify a set of attributes in the Display expression property.<br>Leave this value blank if you want to display the full DN or CN value retrieved from the Identity Vault. The entity you choose must have the directory abstraction layer View property set to True and be the entity whose DN you are retrieving from the Identity Vault. |
| **Display expression** | Required when you specify an **Entity Key for DN expression Lookup**. Choose the attributes to display as the check box labels. For example, to display the user entity’s first and last name attributes, construct an expression like this: FirstName LastName.<br>The attribute’s directory abstraction layer properties for View, Read, Search, and Required must be set to True. |
| **Allow multiple selections** | When this option is set to True, users can select more than one entry. |
| **Field cell style** | Apply an HTML style attribute to the field body.<br>Example: padding-top:5px;background-color:red |

| Property Name | Description |
|---|---|
| **Field CSS class name(s)** | Apply one or more CSS class styles to the field body. You can view the classes that are available for your User Application portal by logging into the User Application, navigating to **Administration > Application Configuration > Theme Administration**, then clicking the **Preview** button for the portal theme you are using.<br>If you leave the field blank, it defaults to the nv-fontExtraSmall class.<br>Example: nv-fontMedium nv-backgroundColor3<br>Separate class names with spaces. |
| **Label cell style** | Apply an HTML style attribute to the field label.<br>Example: color:red |
| **Label CSS class name(s)** | Apply one or more CSS class styles to the field label. You can view the classes that are available for your User Application portal by logging into the User Application, navigating to **Administration > Application Configuration > Theme Administration**, then clicking the **Preview** button for the portal theme you are using.<br>If you leave the field blank, it defaults to the nv-formFieldLabel class.<br>Example: nv-color5<br>Separate class names with spaces. |
| **Show 2 lists** | When this option is set to True, two lists are displayed, one for the unselected values and another for the selected values.<br>The Allow multiple selections property must be set to True. If set to False, **Show 2 lists** is ignored. |
| **Sort entries** | When this option is set to True, sorts results in ascending order. For details, see "Sort Order" on page 106. |

## DatePicker

Use this control for display and entry of a date and time. This 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 by using the format for the user's locale and time zone. 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:

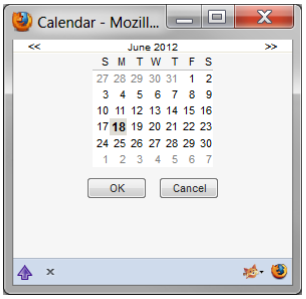
***Figure 5-3*** *Sample DatePicker Control*



DatePickerControl: 05/25/2006 10:47:25 AM

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 looks like this:

***Figure 5-4*** *Sample Calendar Control*



| S | M | T | W | T | F | S |
|---|---|---|---|---|---|---|
| 27 | 28 | 29 | 30 | 31 | 1 | 2 |
| 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |

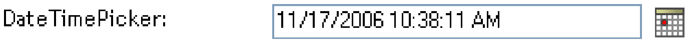
***Table 5-8*** *DatePicker Control Properties*

| Property name | Description |
|---|---|
| **Datetime indicator** | When this option is set to 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. |
| **Field cell style** | Apply an HTML style attribute to the field body.<br>Example: padding-top:5px;background-color:red |
| **Field CSS class name(s)** | Apply one or more CSS class styles to the field body. You can view the classes that are available for your User Application portal by logging into the User Application, navigating to **Administration > Application Configuration > Theme Administration**, then clicking the **Preview** button for the portal theme you are using.<br>If you leave the field blank, it defaults to the nv-fontExtraSmall class.<br>Example: nv-fontMedium nv-backgroundColor3<br>Separate class names with spaces. |
| **Field Width in pixels** | Use this field to configure the field's visible width on the form. The default is 200 pixels. |
| **Label cell style** | Apply an HTML style attribute to the field label.<br>Example: color:red |
| **Label CSS class name(s)** | Apply one or more CSS class styles to the field label. You can view the classes that are available for your User Application portal by logging into the User Application, navigating to **Administration > Application Configuration > Theme Administration**, then clicking the **Preview** button for the portal theme you are using.<br>If you leave the field blank, it defaults to the nv-formFieldLabel class.<br>Example: nv-color5<br>Separate class names with spaces. |
| **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 this option is set to True, displays the calendar pop-up. If it is set to False, the calendar pop-up does not display. Users must type the date in the text field by using the proper format for their locale. |

# DateTimePicker

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

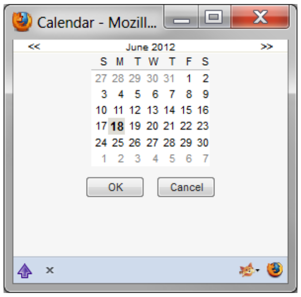
***Figure 5-5*** *Sample DateTimePicker Control*



DateTimePicker: 11/17/2006 10:38:11 AM

When the **Show date picker** property is set to True, the form displays a text field followed by a calendar button. When the user clicks the calendar button, the form launches a calendar control for the user to select the date and time values. The calendar pop-up looks like this:

***Figure 5-6*** *DateTimePicker Calendar Control*



| S | M | T | W | T | F | S |
|---|---|---|---|---|---|---|
| 27 | 28 | 29 | 30 | 31 | 1 | 2 |
| 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |

***Table 5-9*** *DateTimePicker Control Properties*

| Property name | Description |
|---|---|
| **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. |
| **Field width in pixels** | Use this field to configure the field’s visible width on the form. The default is 200 pixels. |
| **isDateTime** | When this option is set to False, the Calendar pop-up does not display the time. |
| **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 this option is set to True, it displays the calendar pop-up. If it is set to False, the calendar pop-up does not display. Users must type the proper format for the locale when they type the date in the text field. |

# DNContainer

Use this control to allow users to select a container object from within the root container that you specify. You can use this control to limit the user to a subtree of a container. This is a specialized version of the DNLookup control.

*Figure 5-7 DNContainer Control With Root Container Specified*

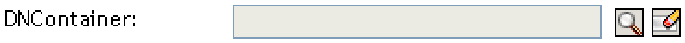


*Table 5-10 DNContainer Control Properties*

| Property name | Description |
|---|---|
| **Entity key used for object lookup** | Choose an entity from the drop-down list. The entity that you choose limits the users ability to look up objects within the specified entity's container. If you specify an entity key and a root container, the entity key takes precedence. |
| **Field cell style** | Apply an HTML style attribute to the field body.<br>Example: padding-top:5px;background-color:red |
| **Field CSS class name(s)** | Apply one or more CSS class styles to the field body. You can view the classes that are available for your User Application portal by logging into the User Application, navigating to **Administration > Application Configuration > Theme Administration**, then clicking the **Preview** button for the portal theme you are using.<br>If you leave the field blank, it defaults to the nv-fontExtraSmall class.<br>Example: nv-fontMedium nv-backgroundColor3<br>Separate class names with spaces. |
| **Field width in pixels** | Use this field to configure the field's visible width on the form. The default is 200 pixels. |
| **Label cell style** | Apply an HTML style attribute to the field label.<br>Example: color:red |
| **Label CSS class name(s)** | Apply one or more CSS class styles to the field label. You can view the classes that are available for your User Application portal by logging into the User Application, navigating to **Administration > Application Configuration > Theme Administration**, then clicking the **Preview** button for the portal theme you are using.<br>If you leave the field blank, it defaults to the nv-formFieldLabel class.<br>Example: nv-color5<br>Separate class names with spaces. |
| **Root container** | Specify a root container for lookups when users click the **Object Selector** button. |
| **Show clear button** | If set to True, the form displays the **Reset field** button |
| **Show object history button** | If set to True, the form displays the **Show history** button. |
| **Show object selector button** | If set to True, the form displays the **Object Selector** button. |

### Displaying the Container Description Instead of the Container O/OU Name

Some additional steps are required to take advantage of an Identity Manager 4.5 DNContainer form field enhancement. This enhancement allows you to display the container description instead of the container O/OU name.

To enable the DNContainer enhancement, you need to manually update the Designer install to add properties to the DNContainer control. Then, you need to create a DAL entity corresponding to the container for which you want to display an attribute. Finally, you need to use the form editor to choose the entity and attribute.

1. Locate the following file in your Designer install:

   `/opt/novell/idm/Designer/plugins/com.novell.core.scriptengineshell_4.5*/lib/UIRegistry.jar`

2. Back it up first, then use a suitable jar/zip tool to modify the file within the jar:

   `com\novell\srvprv\impl\uictrl\UIControlRegistry.xml`

3. Locate the `<ctrl key="DNContainer"` section and add the following properties at the end:

   ```
   <prop name="display-entitydef" type="string" since="1.9">
        <display-label rb-key="LAB_DIS_ENTITYDEF"/>
   </prop>
   <prop name="display-exp" type="expression" since="1.9">
         <display-label rb-key="LAB_DIS_EXPRESSION"/>
   </prop>
   ```

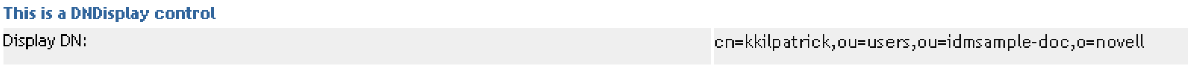
4. Put this file back into the JAR in its original location and start Designer.
5. In Designer, create a new DAL entry with an unused name, such as myDescriptionLookup.
6. For the base class of this DAL entry, choose **Organization**, and pick the attribute you want to show (for example **Description**).
7. When the DAL editor is open, change the LDAP name of the class to **Top**. This allows you to view the Description for Organizations, Organizational Units, and so forth.
8. To use the new DAL entry, open a PRD and go to a form. Add or pick a **dn/DNContainer** field.
9. Fill in the two new fields (**Entity key for DN expression lookup**, **Display expression**) with the values specified above (myDescriptionLookup, Description).
10. Deploy the new DAL entry and the PRD.
11. On the User Application, clear the cache or restart the server.
12. Test the new PRD to ensure that the descriptions are shown instead of the cn in the DNContainer control.

    Make sure the containers you are going to show have a Description value; otherwise, cn is used. Containers, by default, leave this value blank.

## 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. The DNDisplay control cannot be modified by the workflow engine. For this reason, it is not available for post activity mapping.

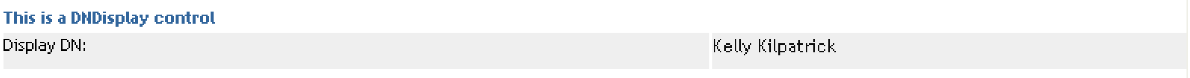
***Figure 5-8*** *Sample DNDisplay*



This is a DNDisplay control

Display DN: cn=kkilpatrick,ou=users,ou=idmsample-doc,o=novell

***Figure 5-9*** *Sample DNDisplay with Display Expression Specified*



This is a DNDisplay control

Display DN: Kelly Kilpatrick

***Table 5-11*** *DNDisplay Control Properties*

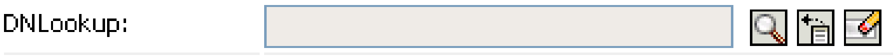
| Property name | Description |
|---|---|
| **Display expression** | Leave this value blank if you want to display the full DN or CN value.<br>If you want to mask the DN by displaying attributes instead, launch the expression builder and select the desired attributes from the list. (You must first specify an **Entity key for DN expression lookup**.)<br>For example, to show the user entity's first and last name attributes, construct an expression like this: `FirstName LastName`.<br>Make sure the attribute's View, Read, Search, and Required properties are set to True in the directory abstraction layer. See "Attribute Properties" on page 63. |
| **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>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>The entity you choose must:<br>◆ Have the directory abstraction layer View property set to True.<br>◆ Be the entity of the DN you are working with.<br>For more information, see "Working with Distinguished Names" on page 140. |
| **Field cell style** | Apply an HTML style attribute to the field body.<br>Example: padding-top:5px;background-color:red |
| **Field CSS class name(s)** | Apply one or more CSS class styles to the field body. You can view the classes that are available for your User Application portal by logging into the User Application, navigating to **Administration > Application Configuration > Theme Administration**, then clicking the **Preview** button for the portal theme you are using.<br>If you leave the field blank, it defaults to the nv-fontExtraSmall class.<br>Example: nv-fontMedium nv-backgroundColor3<br>Separate class names with spaces. |

| Property name | Description |
|---|---|
| **Label cell style** | Apply an HTML style attribute to the field label.<br>Example: color:red |
| **Label CSS class name(s)** | Apply one or more CSS class styles to the field label. You can view the classes that are available for your User Application portal by logging into the User Application, navigating to **Administration > Application Configuration > Theme Administration,** then clicking the **Preview** button for the portal theme you are using.<br>If you leave the field blank, it defaults to the nv-formFieldLabel class.<br>Example: nv-color5<br>Separate class names with spaces. |

## 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 5-10*** *Sample DNLookup Control*



The buttons associated with the DNLookup control are described in Table 5-12.

***Table 5-12*** *DNLookup Control Buttons*

| Button | Description |
|---|---|
| 🔍 | Launches an object lookup dialog box. You define whether the dialog box displays containers or objects via the **Object Selector type** property.<br>The attributes shown in the drop-down list (**Description** in the above example) are specified in the directory abstraction layer. 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 have 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 button property. |

*Table 5-13 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>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>For example, to show the user entity's first and last name attributes, construct an expression like this: `FirstName LastName`.<br>Make sure the attribute's View, Read, Search, and Required properties are set to True in the directory abstraction layer. See "Attribute Properties" on page 63. |
| **Entity key for DN expression lookup** | This property specifies the entity or containers that will be searched when you click the object selector button.<br>If you initialize the control from flowdata, you can configure the control to display an attribute value of the entity rather than the full DN or CN value. If you want to mask the DN or CN by displaying attributes instead, choose the entity from the drop-down list, specify a set of attributes in the **Display expression** property.<br>The entity you choose must:<br>◆ Have the directory abstraction layer View property set to `True`.<br>◆ Be the entity of the DN you are working with.<br>For more information, "Working with Distinguished Names" on page 140. |
| **Field cell style** | Apply an HTML style attribute to the field body.<br>Example: padding-top:5px;background-color:red |
| **Field CSS class name(s)** | Apply one or more CSS class styles to the field body. You can view the classes that are available for your User Application portal by logging into the User Application, navigating to **Administration > Application Configuration > Theme Administration**, then clicking the **Preview** button for the portal theme you are using.<br>If you leave the field blank, it defaults to the nv-fontExtraSmall class.<br>Example: nv-fontMedium nv-backgroundColor3<br>Separate class names with spaces. |
| **Label cell style** | Apply an HTML style attribute to the field label.<br>Example: color:red |

| Property Name | Description |
|---|---|
| **Label CSS class name(s)** | Apply one or more CSS class styles to the field label. You can view the classes that are available for your User Application portal by logging into the User Application, navigating to **Administration > Application Configuration > Theme Administration**, then clicking the **Preview** button for the portal theme you are using.<br>If you leave the field blank, it defaults to the nv-formFieldLabel class.<br>Example: nv-color5<br>Separate class names with spaces. |
| **Object Selector type** | Determines whether the object selector dialog box performs an Object Lookup or a Container Lookup.<br>**paramlist**: Causes the object selector dialog box to perform an object lookup. You specify the lookup criteria via the **Entity key for DN Expression lookup** property.<br>**container**: Causes the object selector dialog box to display one or more containers for selection. The containers for searching are determined by the **Search container** property, which is specified in the directory abstraction layer for the entity named in the required **Entity Key for DN Expression lookup** property. For example, if the **Entity key for DN Expression 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 installation is used. |
| **Field width in pixels** | Use this field to configure the field's visible width on the form. The default is 200 pixels. |
| **Show clear button** | If this option is set to True, the form displays the **Reset field** button. |
| **Show object history button** | If this option is set to True, the form displays the **Show history** button. |
| **Show object selector button** | If this option is set to True, the form displays the **object selector** button. |

## DNMaker

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

***Figure 5-11** Sample DNMaker Control*

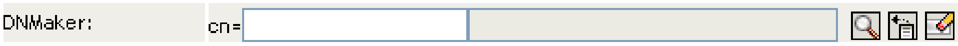


*Table 5-14 DNMaker Control Buttons*

| Button | Description |
|---|---|
| 🔍 | Launches an object selector for container searches.<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 have 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.<br>If the root container is specified for lookups, the Show history button is not shown even if the property is set to True. This has been done for security reasons. |
| ✎ | **Reset field**. Deletes the field contents. The availability of this button is controlled by the Show clear button property. |

*Table 5-15 DNMaker Control Properties*

| Property | Description |
|---|---|
| **Entity key used for object lookup** | A required field. Choose an entity from the drop-down list. This determines the search that is launched when the user clicks the object selector button.<br>If you specify an entity key and a root container, the entity key takes precedence |
| **Naming attribute** | The naming attribute used to construct the final DN. This value displays next to the control's display label as an extra hint to the user. |
| **Field cell style** | Apply an HTML style attribute to the field body.<br>Example: padding-top:5px;background-color:red |
| **Field CSS class name(s)** | Apply one or more CSS class styles to the field body. You can view the classes that are available for your User Application portal by logging into the User Application, navigating to **Administration > Application Configuration > Theme Administration**, then clicking the **Preview** button for the portal theme you are using.<br>If you leave the field blank, it defaults to the nv-fontExtraSmall class.<br>Example: nv-fontMedium nv-backgroundColor3<br>Separate class names with spaces. |
| **Label cell style** | Apply an HTML style attribute to the field label.<br>Example: color:red |

| Property | Description |
|---|---|
| **Label CSS class name(s)** | Apply one or more CSS class styles to the field label. You can view the classes that are available for your User Application portal by logging into the User Application, navigating to **Administration > Application Configuration > Theme Administration**, then clicking the **Preview** button for the portal theme you are using.<br>If you leave the field blank, it defaults to the nv-formFieldLabel class.<br>Example: nv-color5<br>Separate class names with spaces. |
| **Root container** | Specify a root container for lookups when users click the object selector button. If you do not specify a Root container, the User Application uses the container for the entity in the directory abstraction layer property called Search Container. If a search container is not specified for the specified entity, then the Root Container DN specified during the User Application installation is used. If you specify an entity key and a root container, the entity key takes precedence. |
| **Show clear button** | If this option is set to True, the form displays the **Reset field** button. |
| **Show object history button** | If this option is set to True, the form displays the **Show history** button. |
| **Show object selector button** | If this option is set to True, the form displays the **object selector** button. |

## DNQuery

DNQuery is a specialized version of the DNLookup control. Like DNLookup, DNQuery allows users to search and retrieve DNs from the Identity Vault; however, with the DNQuery, the object selector content can be driven by the result of a directory abstraction layer Queries object rather than from properties.

***Table 5-16*** *DNQuery Control Properties*

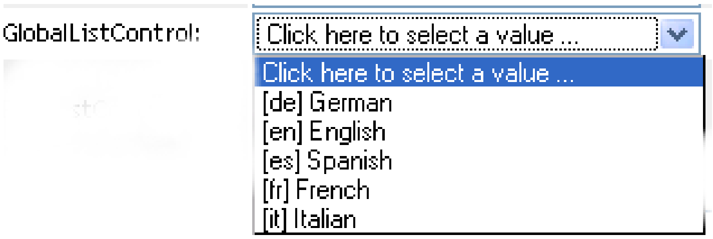
| Property Name | Description |
|---|---|
| **DAL global query key** | Specifies the key of the DAL Queries object you want executed. You can select it from the Event Action Expression Builder. For more information about using DAL queries, see "Using DAL Queries in Forms" on page 143. For more information about defining DAL queries, see "Working with Queries" on page 56. |
| **DAL global query parameter(s)** | Specifies the value for the query parameters. For example, this passes the String Sales to the Queries parameter called groupname:<br>`(function (){return {"groupname":"Sales"}})();` |
| **Display expression** | When you populate the control with initial data from a Data Item Mapping value, use this property to specify the attributes to display. |

| Property Name | Description |
|---|---|
| **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>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>The entity you choose must:<br>◆ Have the directory abstraction layer View property set to True.<br>◆ Be the entity of the DN you are working with.<br>For more information, "Working with Distinguished Names" on page 140. |
| **Field cell style** | Apply an HTML style attribute to the field body.<br>Example: padding-top:5px;background-color:red |
| **Field CSS class name(s)** | Apply one or more CSS class styles to the field body. You can view the classes that are available for your User Application portal by logging into the User Application, navigating to **Administration > Application Configuration > Theme Administration**, then clicking the **Preview** button for the portal theme you are using.<br>If you leave the field blank, it defaults to the nv-fontExtraSmall class.<br>Example: nv-fontMedium nv-backgroundColor3<br>Separate class names with spaces. |
| **Field width in pixels** | Use this field to configure the field's visible width on the form. The default is 200 pixels. |
| **Label cell style** | Apply an HTML style attribute to the field label.<br>Example: color:red |
| **Label CSS class name(s)** | Apply one or more CSS class styles to the field label. You can view the classes that are available for your User Application portal by logging into the User Application, navigating to **Administration > Application Configuration > Theme Administration**, then clicking the **Preview** button for the portal theme you are using.<br>If you leave the field blank, it defaults to the nv-formFieldLabel class.<br>Example: nv-color5<br>Separate class names with spaces. |
| **Show clear button** | If set to True, the form displays the ✎ **Reset field** button |
| **Show object selector button** | If set to True, the form displays the 🔍 **Object Selector** button. |

## 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 5-12*** *Sample Global List Control*



GlobalListControl: Click here to select a value ...

Click here to select a value ...
[de] German
[en] English
[es] Spanish
[fr] French
[it] Italian

***Table 5-17*** *Global List Properties*

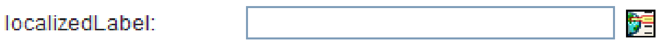
| Property Name | Description |
|---|---|
| **DAL global list key** | Specifies the unique identifier of the global list. This must correspond to the key specified in the directory abstraction layer. |
| **Field cell style** | Apply an HTML style attribute to the field body.<br>Example: padding-top:5px;background-color:red |
| **Field CSS class name(s)** | Apply one or more CSS class styles to the field body. You can view the classes that are available for your User Application portal by logging into the User Application, navigating to **Administration > Application Configuration > Theme Administration**, then clicking the **Preview** button for the portal theme you are using.<br>If you leave the field blank, it defaults to the nv-fontExtraSmall class.<br>Example: nv-fontMedium nv-backgroundColor3<br>Separate class names with spaces. |
| **Label cell style** | Apply an HTML style attribute to the field label.<br>Example: color:red |
| **Label CSS class name(s)** | Apply one or more CSS class styles to the field label. You can view the classes that are available for your User Application portal by logging into the User Application, navigating to **Administration > Application Configuration > Theme Administration**, then clicking the **Preview** button for the portal theme you are using.<br>If you leave the field blank, it defaults to the nv-formFieldLabel class.<br>Example: nv-color5<br>Separate class names with spaces. |

For more information about global lists, see "Working with Lists" on page 54.

## Localized Label

Use this control if you want to allow the user to provide translated text for the field. This control displays the standard icon that indicates the text can be localized.

***Figure 5-13** LocalizedLabel Control*



If the user clicks the icon, they are able to type the text for each language supported by the User Application driver. The list of languages displayed for this control is determined by the contents of the locale resource group called base-resgrp.

***Figure 5-14** LocalizedLabel Control*

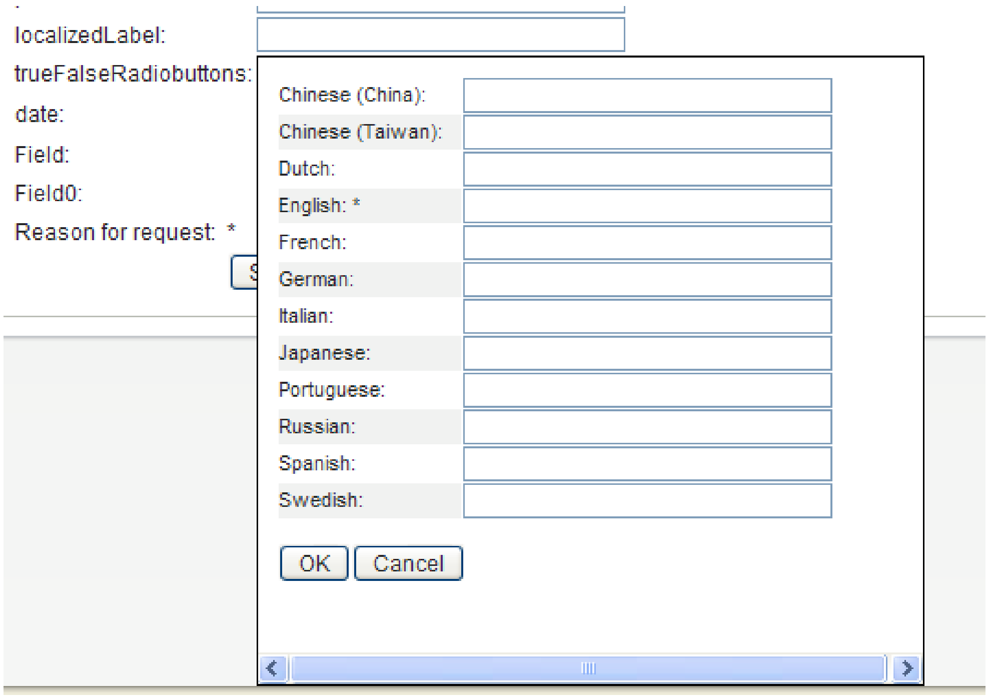


*Table 5-18 LocalizedLabel Control Properties*

| Property Name | Description |
|---|---|
| **Field CSS class name(s)** | Apply one or more CSS class styles to the field body. You can view the classes that are available for your User Application portal by logging into the User Application, navigating to **Administration > Application Configuration > Theme Administration**, then clicking the **Preview** button for the portal theme you are using.<br>If you leave the field blank, it defaults to the nv-fontExtraSmall class.<br>Example: nv-fontMedium nv-backgroundColor3<br>Separate class names with spaces. |
| **Label CSS class name(s)** | Apply one or more CSS class styles to the field label. You can view the classes that are available for your User Application portal by logging into the User Application, navigating to **Administration > Application Configuration > Theme Administration,** then clicking the **Preview** button for the portal theme you are using.<br>If you leave the field blank, it defaults to the nv-formFieldLabel class.<br>Example: nv-color5<br>Separate class names with spaces. |

## Html

Use this control to add HTML fragments to the Form Detail. You can do this by specifying the HTML fragments in the HTML content property. In addition, you can conditionally add the HTML fragment via an event on the form control. In either case, specify the HTML through the use of an anonymous function, such as: `( function() { return "<yourTag yourAttr='your attr value' />"; } )();`

For example:

```
(function(){ return "<table bgcolor='#C0C0C0'><th colspan='3' align='center'>Table
Header Goes Here</th><tr><td>Value 1.1</td><td>Value 1.2</td></tr><tr><td>Value
2.1</td><td>Value 2.2</td></tr></table>"; })()
```

*Table 5-19 HTML Control Properties*

| Property Name | Description |
|---|---|
| **Field cell style** | Apply an HTML style attribute to the field body.<br>Example: padding-top:5px;background-color:red |

| Property Name | Description |
|---|---|
| **Field CSS class name(s)** | Apply one or more CSS class styles to the field body. You can view the classes that are available for your User Application portal by logging into the User Application, navigating to **Administration > Application Configuration > Theme Administration**, then clicking the **Preview** button for the portal theme you are using.<br>If you leave the field blank, it defaults to the nv-fontExtraSmall class.<br>Example: nv-fontMedium nv-backgroundColor3<br>Separate class names with spaces. |
| **Label cell style** | Apply an HTML style attribute to the field label.<br>Example: color:red |
| **Label CSS class name(s)** | Apply one or more CSS class styles to the field label. You can view the classes that are available for your User Application portal by logging into the User Application, navigating to **Administration > Application Configuration > Theme Administration**, then clicking the **Preview** button for the portal theme you are using.<br>If you leave the field blank, it defaults to the nv-formFieldLabel class.<br>Example: nv-color5<br>Separate class names with spaces. |

## MVCheckbox

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

***Figure 5-15*** *Sample MVCheckbox Control*

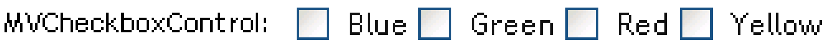
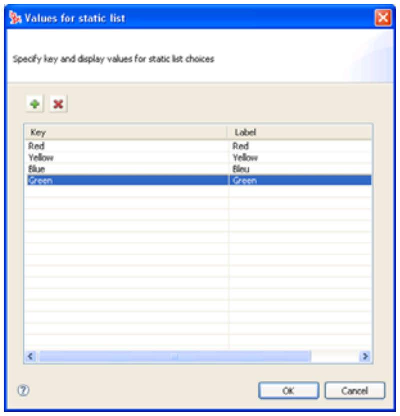


***Table 5-20*** *MVCheckbox Control Properties*

| Property Name | Description |
|---|---|
| **Field cell style** | Apply an HTML style attribute to the field body.<br>Example: padding-top:5px;background-color:red |

| Property Name | Description |
|---|---|
| **Field CSS class name(s)** | Apply one or more CSS class styles to the field body. You can view the classes that are available for your User Application portal by logging into the User Application, navigating to **Administration > Application Configuration > Theme Administration**, then clicking the **Preview** button for the portal theme you are using.<br>If you leave the field blank, it defaults to the nv-fontExtraSmall class.<br>Example: nv-fontMedium nv-backgroundColor3<br>Separate class names with spaces. |
| **Label cell style** | Apply an HTML style attribute to the field label.<br>Example: color:red |
| **Label CSS class name(s)** | Apply one or more CSS class styles to the field label. You can view the classes that are available for your User Application portal by logging into the User Application, navigating to **Administration > Application Configuration > Theme Administration**, then clicking the **Preview** button for the portal theme you are using.<br>If you leave the field blank, it defaults to the nv-formFieldLabel class.<br>Example: nv-color5<br>Separate class names with spaces. |
| **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><br>Values for static list<br>Specify key and display values for static list choices<br>Key — Label<br>Red — Red<br>Yellow — Yellow<br>Blue — Bleu<br>Green — Green<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 "Form Control Examples" on page 279.

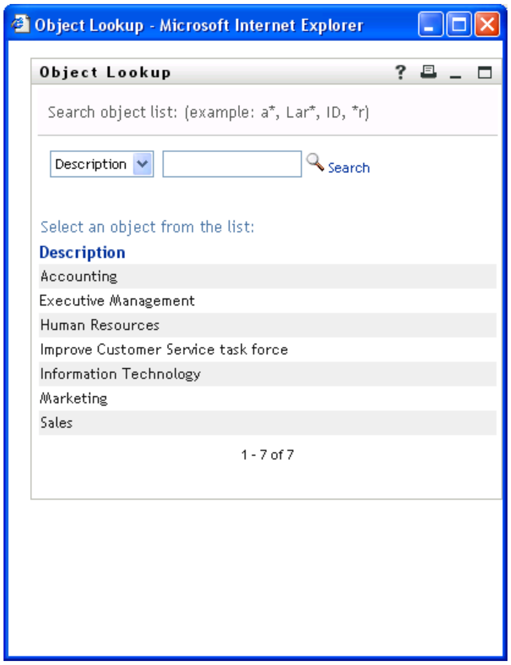
## MVEditor

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 display 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 for DN expression lookup property.

There are also properties that let you specify a DAL Global Query to execute or specify a root DN to drive the object picker.

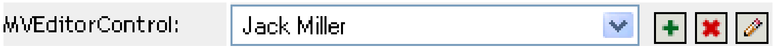
***Table 5-21*** *MVEditor with Object Selector Properties Set Control Buttons*

| Button | Description |
|---|---|
| 🔍 | Launches a search dialog box called an object selector. The object selector dialog box looks like this:<br><br>Object Lookup - Microsoft Internet Explorer<br>Object Lookup<br>Search object list: (example: a*, Lar*, ID, *r)<br>Description<br>Search<br>Select an object from the list:<br>**Description**<br>Accounting<br>Executive Management<br>Human Resources<br>Improve Customer Service task force<br>Information Technology<br>Marketing<br>Sales<br>1 - 7 of 7<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 65. 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 have 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 button property. |

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

***Figure 5-16*** *Sample MVEditor without Object Lookup Properties Set*



MVEditorControl: Jack Miller

***Table 5-22*** *MVEditor Control Buttons*

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

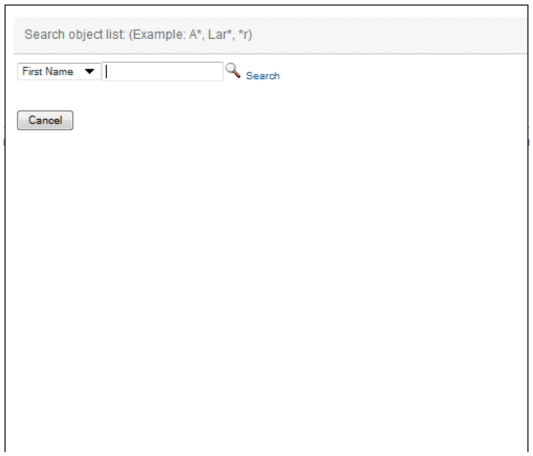
**TIP:** When the MVEditor control's Editable property is set to False, this control is read-only and the form does not display any MVEditor control buttons.

***Table 5-23*** *MVEditor Control Properties*

| Property Name | Description |
|---|---|
| **Add data entry text field** | When this option is set to True and there is a single row of data (and the data is not a DN), the control displays a data entry text field. The text field is displayed when the field is empty or contains only one value. Otherwise, the drop-down list is displayed. If more than one row of data exists, then the drop-down list always displays. |
| **DAL Global Query** | Specify this value if you want the control populated by the results of the Global Query that you specify. You specify the key name. You can select it from the Event Action Expression Builder. For more information about using queries in forms, see "Using DAL Queries in Forms" on page 143. For information about defining queries, see "Working with Queries" on page 56. |
| **DAL Global Query Parameter(s)** | Specifies the value for the query parameters. For example, this passes the String Sales to the queries parameter called groupname.<br>`(function (){return {"groupname":"Sales"}})();` |

| Property Name | Description |
|---|---|
| **Display expression** | Leave this value blank if you want to display the full DN or CN value.<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>For example, to show the user entity’s first and last name attributes, construct an expression like this: `FirstName LastName`.<br>Make sure the attribute’s View, Read, Search, and Required properties are set to True in the directory abstraction layer. See “Attribute Properties” on page 63. |
| **Enforce uniqueness** | Forces user-entered list items to be unique. |
| **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>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>The entity you choose must<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>See “Working with Distinguished Names” on page 140 for more information. |
| **Field cell style** | Apply an HTML style attribute to the field body.<br>Example: padding-top:5px;background-color:red |
| **Field CSS class name(s)** | Apply one or more CSS class styles to the field body. You can view the classes that are available for your User Application portal by logging into the User Application, navigating to **Administration > Application Configuration > Theme Administration**, then clicking the **Preview** button for the portal theme you are using.<br>If you leave the field blank, it defaults to the nv-fontExtraSmall class.<br>Example: nv-fontMedium nv-backgroundColor3<br>Separate class names with spaces. |
| **Label cell style** | Apply an HTML style attribute to the field label.<br>Example: color:red |

| Property Name | Description |
|---|---|
| **Label CSS class name(s)** | Apply one or more CSS class styles to the field label. You can view the classes that are available for your User Application portal by logging into the User Application, navigating to **Administration > Application Configuration > Theme Administration**, then clicking the **Preview** button for the portal theme you are using.<br>If you leave the field blank, it defaults to the nv-formFieldLabel class.<br>Example: nv-color5<br>Separate class names with spaces. |
| **Field width in pixels** | Use this field to configure the field's visible width on the form. The default is 200 pixels. |
| **Ignore case** | If this option is set to 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. The control blocks input when this value is reached. |
| **Minimum length** | Minimum number of characters for string values. The control validates that the user enters at least this number of characters. |
| **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.<br>You can set the number of lines to 1 or to 3 or greater. You cannot set it to 2 because it does not leave enough space for the browser to display scroll bars. If you set it to 2, Designer generates a warning in the Project Checker view and resets it to 3.<br>**NOTE:** When displaying large result sets in a multivalued control, different browsers render controls at different speeds. We recommend limiting the number of lines displayed in a control to a maximum of 150 lines. |
| **Numbers only** | If this option is set to 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. The following is an example of an Object Lookup:<br><br> Search object list: (Example: A*, Lar*, *r)<br>First Name<br>Search<br>Cancel<br>**paramlist**: Causes the object selector dialog box to perform an object lookup. You specify the lookup criteria via the Entity key for DN expression lookup property.<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, which is specified in the directory abstraction layer for the entity named in the Entity key for DN expression lookup property. For example, if the Entity key for DN expression 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 installation is used. |
| **Resolve DN** | When set to False, the DN is displayed rather than the Display expression. Consider using this when you expect a large number of DNs to be returned, and you are concerned about performance. |
| **Root Container** | Specify a root container for lookups when users click the object selector button. If you specify an entity key and a root container, the entity key takes precedence. |
| **Show object history button** | When this option is set to True, displays the **Object History** button next to the control. |
| **Show object selector button** | When this option is set to True, displays the **Object Selector** button next to the control. |
| **Sort entries** | When this option is set to True, sorts the results in ascending order. For details, see "Sort Order" on page 106. |
| **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 271.

## Password

Use the Password control to allow users to mask all of the user's entries with the * character.

The password control can only be submitted. It does not support any script such as getValues() or setValues().

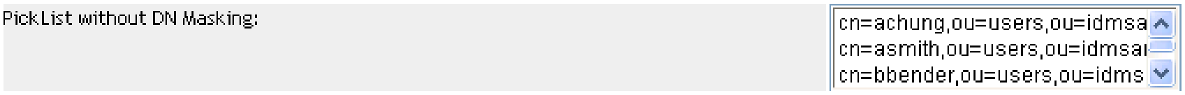
*Table 5-24 Password Control Properties*

| Property Name | Description |
|---|---|
| **Field width in pixels** | Use this field to configure the field's visible width on the form. The default is 200 pixels. |
| **Lower bounds (for numbers only)** | The lowest number allowed for decimal or integer values. |
| **Maximum length** | Maximum number of characters for string values. The control blocks input when this value is reached. |
| **Minimum length** | Minimum number of characters for string values. The control validates that the user enters at least this number of characters. |
| **Number of characters allowed** | Specifies the number of characters a user is allowed to enter. This is related to Field width in pixels. |
| **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. Designer provides a default set of validation masks by default. You must enable them through **Windows > Preferences > Provisioning > Validation Mask**. For more information, see "Setting Provisioning View Preferences" on page 27. |

## 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.

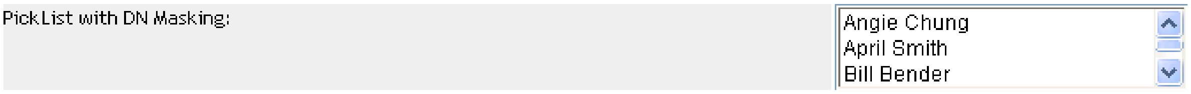
*Figure 5-17 Sample PickList Control without DN Masking*



PickList without DN Masking:

cn=achung,ou=users,ou=idmsa
cn=asmith,ou=users,ou=idmsa
cn=bbender,ou=users,ou=idms

*Figure 5-18 Sample PickList Control with DN Masking*



PickList with DN Masking:

Angie Chung
April Smith
Bill Bender

*Table 5-25 PickList Control Properties*

| Property Name | Description |
|---|---|
| **Allow multiple selections** | When this option is set to True, the user can select more than one list value using their platform-specific multi-select keys.<br>When this option is set to True, the control displays a minimum of three lines regardless of the value specified in the Number of lines displayed property. If this value is False, the Number of lines displayed property is used. |
| **Display expression** | Leave this value blank if you want to display the full DN or CN value.<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>For example, to show the user entity's first and last name attributes, construct an expression like this: `FirstName LastName`.<br>Make sure the attribute's View, Read, Search, and Required properties are set to True in the directory abstraction layer. See "Attribute Properties" on page 63. |
| **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>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>The entity you choose must:<br>◆ Have the directory abstraction layer View property set to True.<br>◆ Be the entity whose DN you are retrieving from the Identity Vault. |
| **Field cell style** | Apply an HTML style attribute to the field body.<br>Example: padding-top:5px;background-color:red |

| Property Name | Description |
|---|---|
| **Field CSS class name(s)** | Apply one or more CSS class styles to the field body. You can view the classes that are available for your User Application portal by logging into the User Application, navigating to **Administration > Application Configuration > Theme Administration**, then clicking the **Preview** button for the portal theme you are using.<br>If you leave the field blank, it defaults to the nv-fontExtraSmall class.<br>Example: nv-fontMedium nv-backgroundColor3<br>Separate class names with spaces. |
| **Field width in pixels** | Use this field to configure the field's visible width on the form. The default is 200 pixels. |
| **Label cell style** | Apply an HTML style attribute to the field label.<br>Example: color:red |
| **Label CSS class name(s)** | Apply one or more CSS class styles to the field label. You can view the classes that are available for your User Application portal by logging into the User Application, navigating to **Administration > Application Configuration > Theme Administration**, then clicking the **Preview** button for the portal theme you are using.<br>If you leave the field blank, it defaults to the nv-formFieldLabel class.<br>Example: nv-color5<br>Separate class names with spaces. |
| **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.<br>The number of lines displayed is related to the Allow multiple selections setting. When Allow multiple selections is set to True, the number of lines displayed is always 3 (or more). When Allow multiple selections is set to False, you can set the number of lines to 1 or to 3 or greater. You cannot set it to 2 because it does not leave enough space for the browser to display scroll bars. If you set it to 2, Designer generates a warning in the Project Checker view and resets it to 3.<br>**NOTE:** When displaying large result sets in a multivalued control, different browsers render controls at different speeds. We recommend limiting the number of lines displayed in a control to a maximum of 150 lines. |

| Property Name | Description |
|---|---|
| **Show 2 lists** | When this option is set to True, two lists are displayed. A list on the left displays the unselected values, and the list on the right displays the selected values.<br>The Allow multiple selections property must be set to True. If set to false **Show 2 lists** is ignored. |
| **Sort Entries** | When this option is set to True, sorts results in ascending order. For details, see "Sort Order" on page 106. |

**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 "Form Control Examples" on page 279.

## RadioButtons

Use this control to display a choice of items as a set of radio buttons. Only one item can be selected.

***Table 5-26** RadioButtons Control Properties*

| Property Name | Description |
|---|---|
| **Allow multiple selections** | When this option is set to True, the user can select more than one list value using their platform-specific multi-select keys.<br>When this option is set to True, the control displays a minimum of three lines regardless of the value specified in the Number of lines displayed property. If this value is False, the Number of lines displayed property is used. |
| **Display expression** | Leave this value blank if you want to display the full DN or CN value.<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>For example, to show the user entity's first and last name attributes, construct an expression like this: `FirstName LastName`.<br>Make sure the attribute's View, Read, Search, and Required properties are set to True in the directory abstraction layer. See "Attribute Properties" on page 63. |

| 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>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>The entity you choose must:<br>◆ Have the directory abstraction layer View property set to True.<br>◆ Be the entity whose DN you are retrieving from the Identity Vault. |
| **Field cell style** | Apply an HTML style attribute to the field body.<br>Example: padding-top:5px;background-color:red |
| **Field CSS class name(s)** | Apply one or more CSS class styles to the field body. You can view the classes that are available for your User Application portal by logging into the User Application, navigating to **Administration > Application Configuration > Theme Administration**, then clicking the **Preview** button for the portal theme you are using.<br>If you leave the field blank, it defaults to the nv-fontExtraSmall class.<br>Example: nv-fontMedium nv-backgroundColor3<br>Separate class names with spaces. |
| **Field width in pixels** | Use this field to configure the field’s visible width on the form. The default is 200 pixels. |
| **Label cell style** | Apply an HTML style attribute to the field label.<br>Example: color:red |
| **Label CSS class name(s)** | Apply one or more CSS class styles to the field label. You can view the classes that are available for your User Application portal by logging into the User Application, navigating to **Administration > Application Configuration > Theme Administration**, then clicking the **Preview** button for the portal theme you are using.<br>If you leave the field blank, it defaults to the nv-formFieldLabel class.<br>Example: nv-color5<br>Separate class names with spaces. |

| Property Name | Description |
|---|---|
| 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.<br>The number of lines displayed is related to the Allow multiple selections setting. When Allow multiple selections is set to True, the number of lines displayed is always 3 (or more). When Allow multiple selections is set to False, you can set the number of lines to 1 or to 3 or greater. You cannot set it to 2 because it does not leave enough space for the browser to display scroll bars. If you set it to 2, Designer generates a warning in the Project Checker view and resets it to 3.<br>**NOTE:** When displaying large result sets in a multivalued control, different browsers render controls at different speeds. We recommend limiting the number of lines displayed in a control to a maximum of 150 lines. |
| Sort Entries | When this option is set to True, sorts results in ascending order. For details, see "Sort Order" on page 106. |

## 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 5-19** Sample Static List Control*

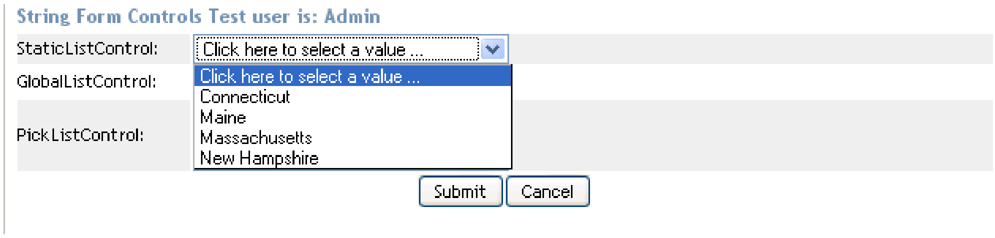
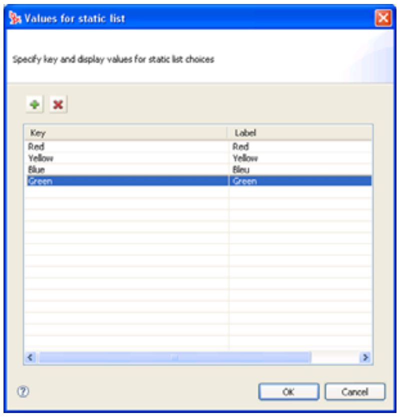


***Table 5-27** Static List Properties*

| Property Name | Description |
|---|---|
| Autoselect first value | Allows you to configure the form to automatically select the first value in the static list. You can select **yes**, **no**, or **only if required is true**.<br>If you select only if required is true, the form only automatically selects the first value in the list if the value of the default **Required** property is **true**. |

| Property Name | Description |
|---|---|
| **Field cell style** | Apply an HTML style attribute to the field body.<br>Example: padding-top:5px;background-color:red |
| **Field CSS class name(s)** | Apply one or more CSS class styles to the field body. You can view the classes that are available for your User Application portal by logging into the User Application, navigating to **Administration > Application Configuration > Theme Administration**, then clicking the **Preview** button for the portal theme you are using.<br>If you leave the field blank, it defaults to the nv-fontExtraSmall class.<br>Example: nv-fontMedium nv-backgroundColor3<br>Separate class names with spaces. |
| **Field width in pixels** | Use this field to configure the field's visible width on the form. The default is 200 pixels. |
| **Label cell style** | Apply an HTML style attribute to the field label.<br>Example: color:red |
| **Label CSS class name(s)** | Apply one or more CSS class styles to the field label. You can view the classes that are available for your User Application portal by logging into the User Application, navigating to **Administration > Application Configuration > Theme Administration**, then clicking the **Preview** button for the portal theme you are using.<br>If you leave the field blank, it defaults to the nv-formFieldLabel class.<br>Example: nv-color5<br>Separate class names with spaces. |
| **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>Values for static list<br>Specify key and display values for static list choices<br>Key / Label<br>Red / Red<br>Yellow / Yellow<br>Blue / Bleu<br>Green / Green<br> <br>Click **Add** to add list items. Each list item must have a unique key. The dialog box automatically generates a unique key when you insert a new list item. You can click 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. |

## Text

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

***Figure 5-20*** *Sample Text Control*

TextControl:

***Table 5-28*** *Text Control Properties*

| Property Name | Description |
|---|---|
| **Field cell style** | Apply an HTML style attribute to the field body.<br>Example: padding-top:5px;background-color:red |
| **Field CSS class name(s)** | Apply one or more CSS class styles to the field body. You can view the classes that are available for your User Application portal by logging into the User Application, navigating to **Administration > Application Configuration > Theme Administration**, then clicking the **Preview** button for the portal theme you are using.<br>If you leave the field blank, it defaults to the nv-fontExtraSmall class.<br>Example: nv-fontMedium nv-backgroundColor3<br>Separate class names with spaces. |
| **Field width in pixels** | Use this field to configure the field's visible width on the form. The default is 200 pixels. |
| **Label cell style** | Apply an HTML style attribute to the field label.<br>Example: color:red |
| **Label CSS class name(s)** | Apply one or more CSS class styles to the field label. You can view the classes that are available for your User Application portal by logging into the User Application, navigating to **Administration > Application Configuration > Theme Administration**, then clicking the **Preview** button for the portal theme you are using.<br>If you leave the field blank, it defaults to the nv-formFieldLabel class.<br>Example: nv-color5<br>Separate class names with spaces. |
| **Lower bounds (for numbers only)** | The lowest number allowed for decimal or integer values. |
| **Maximum length** | The maximum length for string values. Blocks input when this length is reached. |
| **Minimum length** | The minimum length for string values. Validates that the user enters a string at least this long. |
| **Number of characters allowed** | Specifies the number of characters a user is allowed to enter. This is related to Field width in pixels. |
| **Upper bound (for numbers only)** | The highest number allowed for decimal or integer values. |

| Property Name | Description |
|---|---|
| **Validation Mask (regular expression)** | An expression used for validating the field's data. Designer provides a default set of validation masks by default. You must enable them through **Windows > Preferences > Provisioning > Validation Mask**. For more information, see "Setting Provisioning View Preferences" on page 27. |

# Text Area

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

***Figure 5-21*** *Sample Text Area Control*

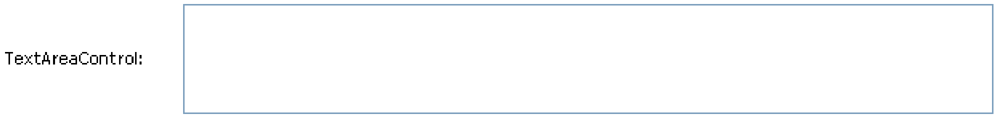


***Table 5-29*** *Text Area Control Properties*

| Property Name | Description |
|---|---|
| **Field cell style** | Apply an HTML style attribute to the field body.<br>Example: padding-top:5px;background-color:red |
| **Field CSS class name(s)** | Apply one or more CSS class styles to the field body. You can view the classes that are available for your User Application portal by logging into the User Application, navigating to **Administration > Application Configuration > Theme Administration**, then clicking the **Preview** button for the portal theme you are using.<br>If you leave the field blank, it defaults to the nv-fontExtraSmall class.<br>Example: nv-fontMedium nv-backgroundColor3<br>Separate class names with spaces. |
| **Label cell style** | Apply an HTML style attribute to the field label.<br>Example: color:red |

| Property Name | Description |
|---|---|
| **Label CSS class name(s)** | Apply one or more CSS class styles to the field label. You can view the classes that are available for your User Application portal by logging into the User Application, navigating to **Administration > Application Configuration > Theme Administration**, then clicking the **Preview** button for the portal theme you are using.<br>If you leave the field blank, it defaults to the nv-formFieldLabel class.<br>Example: nv-color5<br>Separate class names with spaces. |
| **Number of columns displayed** | The visible width of the control; the number of characters wide. |
| **Number of lines displayed** | The number of lines to display at one time. |

## Title

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

***Table 5-30*** *Title Control Properties*

| Property Name | Description |
|---|---|
| **Display title in signed form document** | When this option is set to False and the form is a signed form (using digital signatures), the title control is not displayed. |
| **Font-size** | Specify small, medium, or large. |
| **Style class** | Choose a font style (such as bold) and colors from a palette. |

## TrueFalseCheckBox

Use this control to allow the user to select or deselect a choice. The values returned by the control are `true` (selected) and `false` (not selected). The returned values are always strings.

**NOTE:** This control returns a boolean value of True or False depending on the selection made by the user. The display values of the control can be set to a more user friendly or informative value such as Yes/No or Accept/Reject depending on the intended use of the form; however, the underlying value always returns either True or False. If the control is used to return anything other than True or False or more than two values, for example (Yes/No/Maybe), by default it returns the value as False if it is not True.

You can initialize the control from a JavaScript Boolean Object. The `field.getValue()` returns a JavaScript Boolean Object.

Use `setValues(["true"])` not `setValues[true])`, the `setValues()` method expects a string or an array of string values.

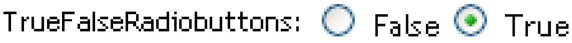
*Table 5-31 TrueFalseCheckBox Control Properties*

| Property Name | Description |
|---|---|
| **Field cell style** | Apply an HTML style attribute to the field body.<br>Example: padding-top:5px;background-color:red |
| **Field CSS class name(s)** | Apply one or more CSS class styles to the field body. You can view the classes that are available for your User Application portal by logging into the User Application, navigating to **Administration > Application Configuration > Theme Administration**, then clicking the **Preview** button for the portal theme you are using.<br>If you leave the field blank, it defaults to the nv-fontExtraSmall class.<br>Example: nv-fontMedium nv-backgroundColor3<br>Separate class names with spaces. |
| **Field width in pixels** | Use this field to configure the field's visible width on the form. The default is 200 pixels. |
| **Label cell style** | Apply an HTML style attribute to the field label.<br>Example: color:red |
| **Label CSS class name(s)** | Apply one or more CSS class styles to the field label. You can view the classes that are available for your User Application portal by logging into the User Application, navigating to **Administration > Application Configuration > Theme Administration**, then clicking the **Preview** button for the portal theme you are using.<br>If you leave the field blank, it defaults to the nv-formFieldLabel class.<br>Example: nv-color5<br>Separate class names with spaces. |

## TrueFalseRadioButtons

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

*Figure 5-22 Sample TrueFalseRadioButtons Control*



TrueFalseRadiobuttons: ○ False ● True

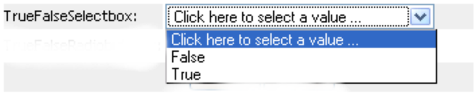
This control has no custom properties.

**NOTE:** This control returns a Boolean value of True or False depending on the selection made by the user. The display values of the control can be set to a more user friendly or informative value such as Yes/No or Accept/Reject depending on the intended use of the form; however, the underlying value always returns either True or False. If the control is used to return anything other than True or False or more than two values, for example (Yes/No/Maybe), by default it returns the value as False if it is not True.

## TrueFalseSelectBox

Use this control to display a choice of True or False in a drop-down list.

***Figure 5-23*** *Sample TrueFalseSelectBox Control*


TrueFalseSelectbox:
Click here to select a value ...
Click here to select a value ...
False
True

**NOTE:** This control returns a Boolean value of True or False depending on the selection made by the user. The display values of the control can be set to a more user friendly or informative value such as Yes/No or Accept/Reject depending on the intended use of the form; however, the underlying value always returns either True or False. If the control is used to return anything other than True or False or more than two values, for example (Yes/No/Maybe), by default it returns the value as False if it is not True.

***Table 5-32*** *TrueFalsSelectBox Properties*

| Property Name | Description |
|---|---|
| **Field width in pixels** | Use this field to configure the field's visible width on the form. The default is 200 pixels. |

# Working with Distinguished Names

The following controls provide specialized support for distinguished names (DNs):

- DNDisplay
- DNLookup
- DNMaker
- MVEditor
- PickList

This section describes the specialized support, including the following:

- "Formatting DNs" on page 140
- "Working with Object Selectors" on page 141

## Formatting DNs

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 controls 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.

## Working with Object Selectors

In some cases, you might want the user to search for and select a DN from a list of possible values. The object selector dialog box (also called the object lookup dialog box) provides this functionality. The contents of the object selector dialog box are controlled by the form control's properties (see Table 5-33), and by how DAL properties are defined (see "DNLookup Control Type Definitions and Object Selector Contents" on page 141).

The object selector only supports attributes of String or DNLookup data types whose directory abstraction layer access properties for required and searchable are set to true.

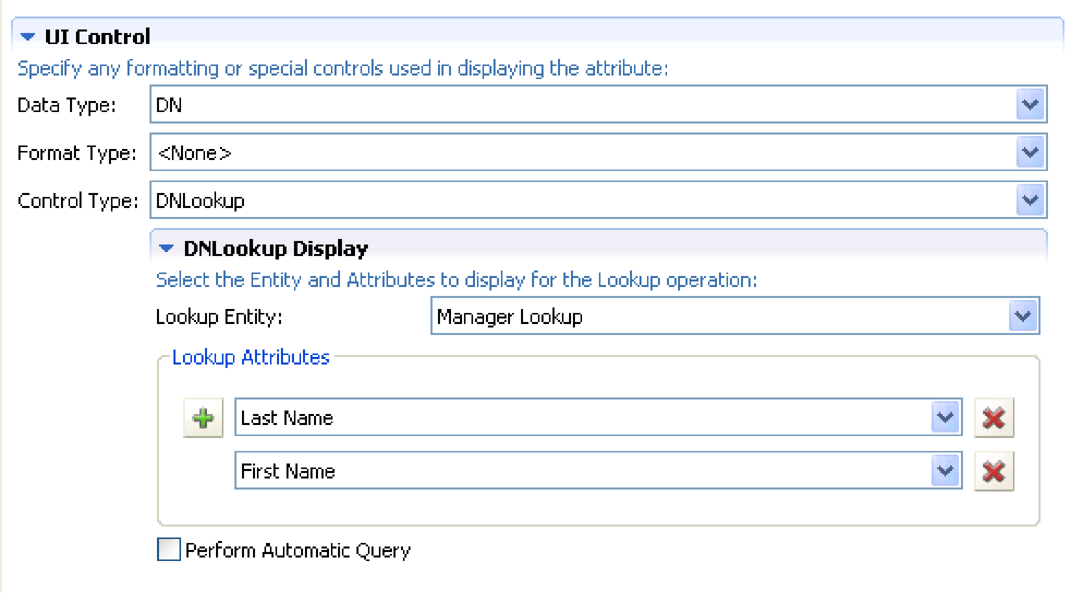
*Table 5-33 Properties for Defining the Object Selector Dialog Box*

| Property | Description |
|---|---|
| **Entity key for DN expression 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. |
| **Object selector type** | **paramlist:** Causes the object selector dialog box to perform an object lookup. You specify the lookup criteria via the Entity key for DN expression lookup property.<br>**container:** Causes the object selector dialog box to display one or more containers for selection. The containers for searching are determined by the Search container property that is specified in the directory abstraction layer for the entity named in the Entity key for DN expression lookup property. For example, if the Entity key for DN expression lookup property is Group, the search container is set to **%group-root%** by default. If no entity is specified you will get an error message about missing or bad Container Selector properties |
| **Show object selector button** | If this option is set to 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 for DN expression lookup, the object selector's contents are defined by 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 because 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 in Figure 5-24.

*Figure 5-24 Manager Attribute on User DNLookup Property Definition*


UI Control

Specify any formatting or special controls used in displaying the attribute:

Data Type: DN

Format Type: <None>

Control Type: DNLookup

DNLookup Display

Select the Entity and Attributes to display for the Lookup operation:

Lookup Entity: Manager Lookup

Lookup Attributes

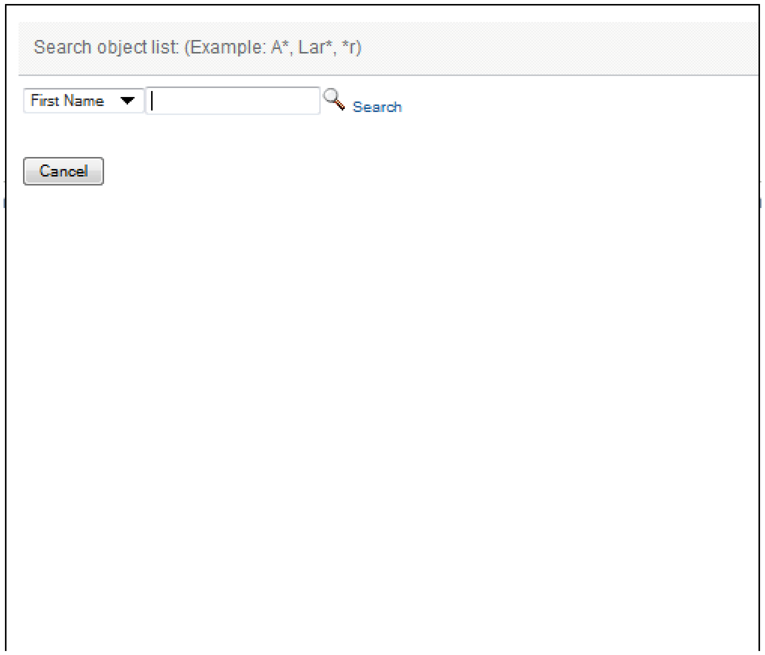
Last Name

First Name

□ Perform Automatic Query

The resulting object selector is shown in Figure 5-25.

*Figure 5-25 Sample Object Selector*


Search object list: (Example: A*, Lar*, *r)

First Name

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; otherwise, they won’t appear in the object selector dialog box.

3. If the attribute does not already exist for the Lookup Entity, you must do the following:
   - Add the attribute to the Lookup Entity. For example, to display another attribute in a manager lookup like the one above, add the attribute to the Manager Lookup entity. For more information, see “Adding Attributes” on page 53.
   - 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.

# Using DAL Queries in Forms

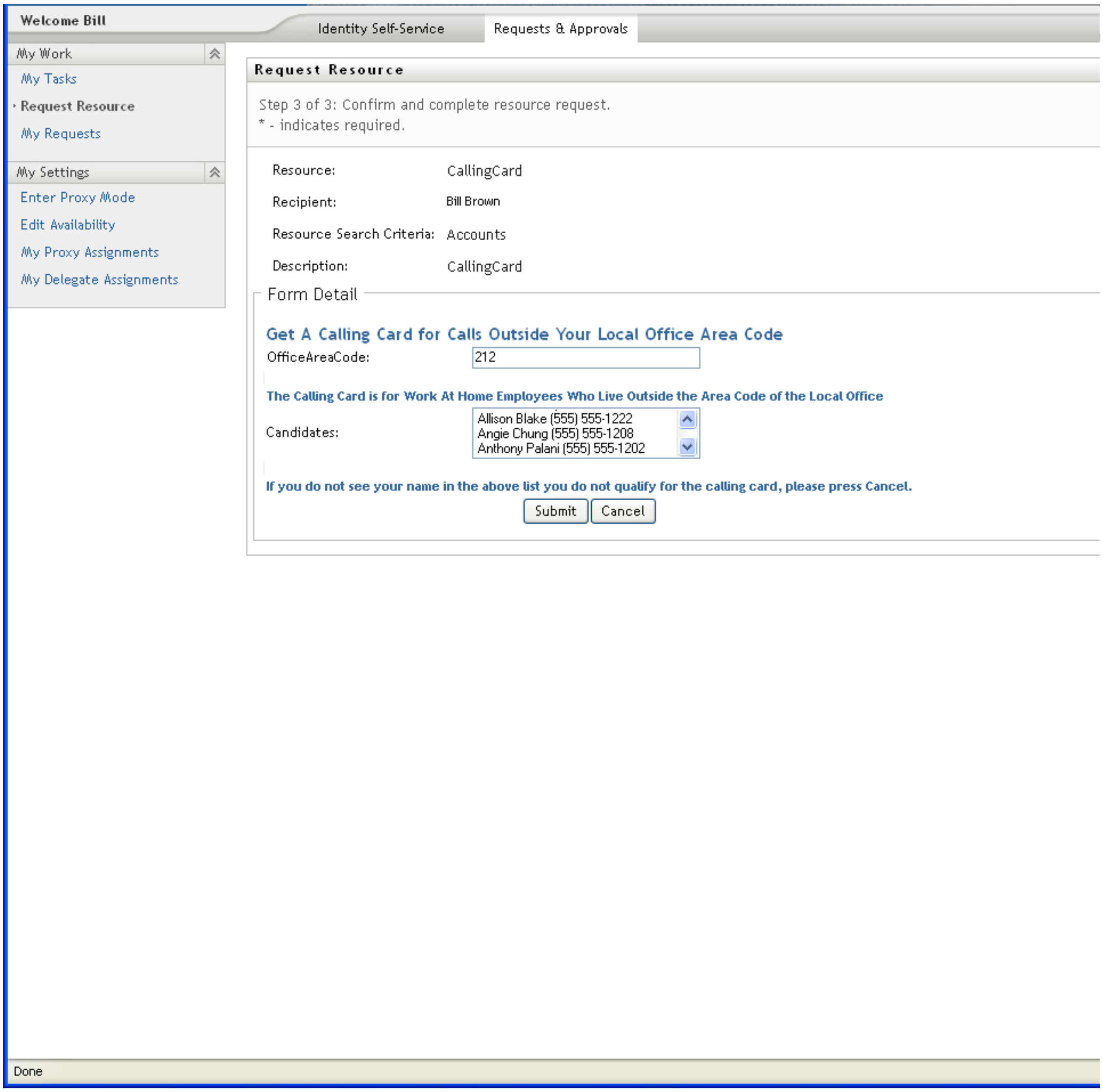
The Query objects defined in the directory abstraction layer let you predefine LDAP searches that you can then execute from a workflow form. The information in this section illustrates how you can define a query and use it in a form.

Suppose that you want to distribute calling cards to certain employees, but you only want to distribute calling cards to employees who work at home, and whose homes are located outside of the local office’s area code. You create a workflow form that allows the employee to:

- Verify that they qualify to receive a card.
- Submit a request for a card if they do qualify.

On your form, you allow users to enter the area code of their own local office (and based on that area code) review a list of users that qualify for a card. The runtime form is shown in Figure 5-26.

*Figure 5-26 Sample Calling Card Request Form*



Step 3 of 3: Confirm and complete resource request.
* - indicates required.

Resource: CallingCard
Recipient: Bill Brown
Resource Search Criteria: Accounts
Description: CallingCard

Get A Calling Card for Calls Outside Your Local Office Area Code
OfficeAreaCode: 212

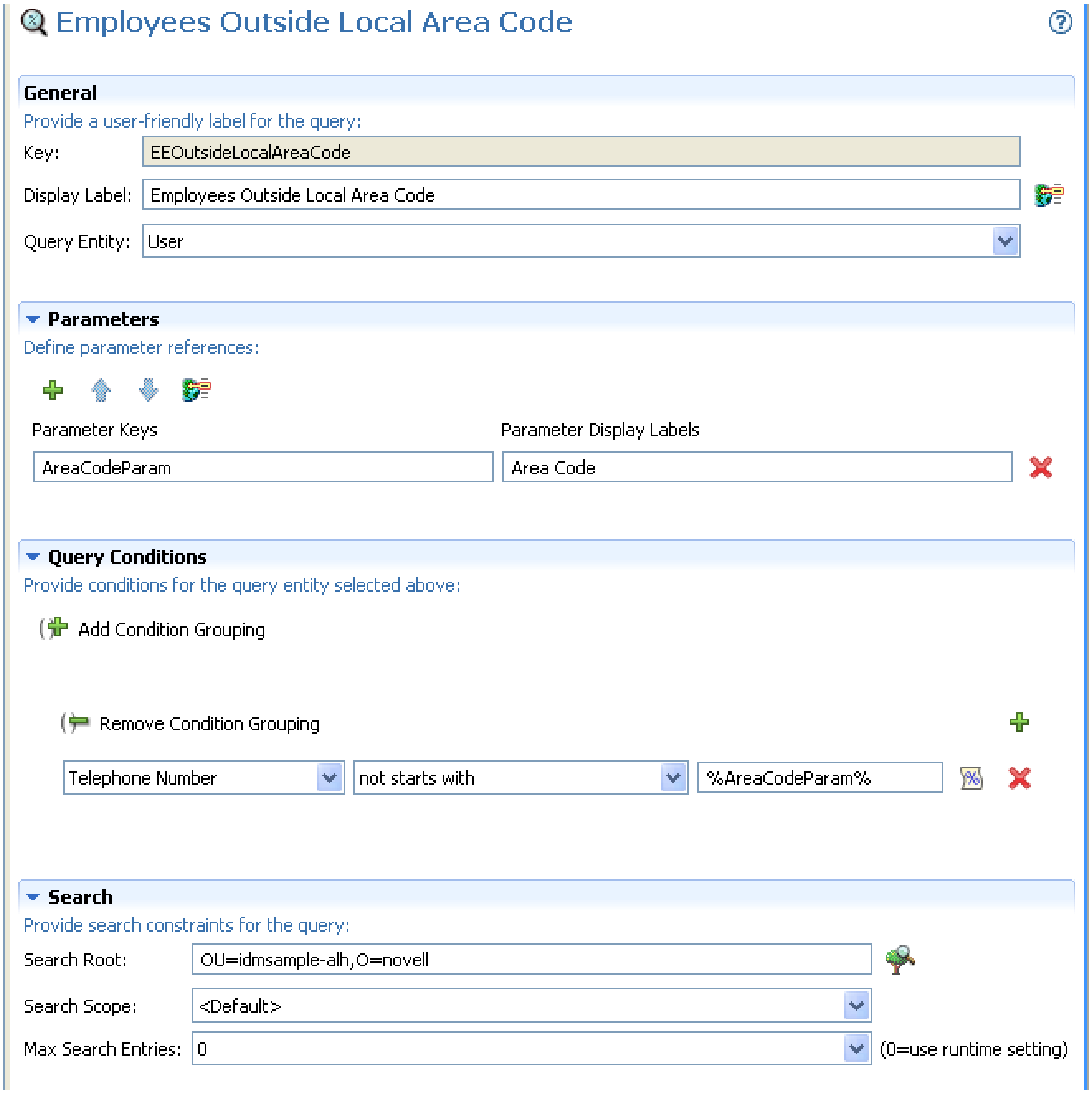
The Calling Card is for Work At Home Employees Who Live Outside the Area Code of the Local Office

Candidates:
Allison Blake (555) 555-1222
Angie Chung (555) 555-1208
Anthony Palani (555) 555-1202

If you do not see your name in the above list you do not qualify for the calling card, please press Cancel.

The data in the Candidates Picklist control is populated from the results of a query that is defined as shown in Figure 5-27.

*Figure 5-27 Calling Card Queries Definition*



Employees Outside Local Area Code

General

Provide a user-friendly label for the query:

Key: EEOutsideLocalAreaCode

Display Label: Employees Outside Local Area Code

Query Entity: User

Parameters

Define parameter references:

| Parameter Keys | Parameter Display Labels |
|---|---|
| AreaCodeParam | Area Code |

Query Conditions

Provide conditions for the query entity selected above:

Add Condition Grouping

Remove Condition Grouping

Telephone Number | not starts with | %AreaCodeParam%

Search

Provide search constraints for the query:

Search Root: OU=idmsample-alh,O=novell

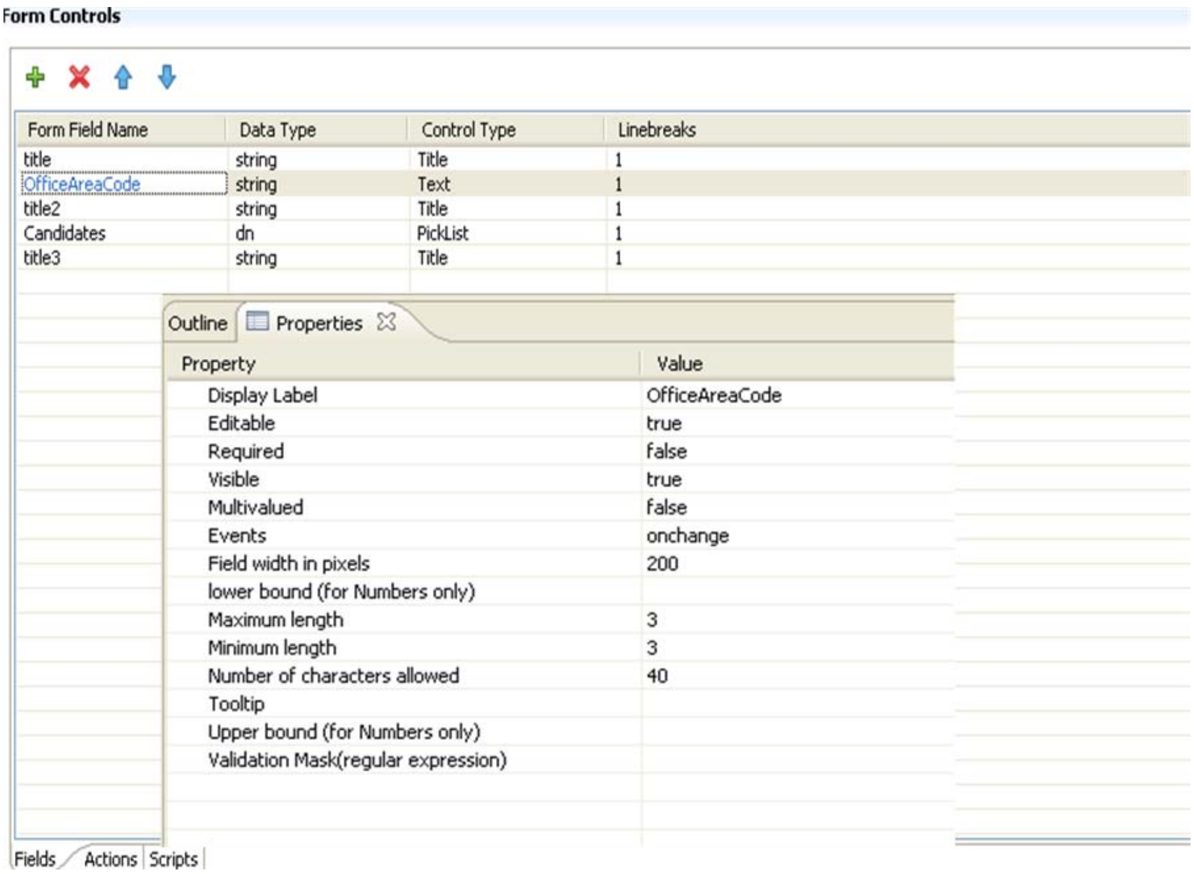
Search Scope: <Default>

Max Search Entries: 0 (0=use runtime setting)

The query takes a single input parameter, **AreaCodeParam**, for the user-entered area code. The query then searches the **User** entity (in the idmsample-alh container) and returns the users whose telephone numbers do not start with the same value entered in the **AreaCodeParam**.

The form has an input field called **OfficeAreaCode**. It is the text field where the user enters the area code of the local office. The properties for OfficeAreaCode are show in Figure 5-28.

*Figure 5-28 OfficeAreaCode Properties*

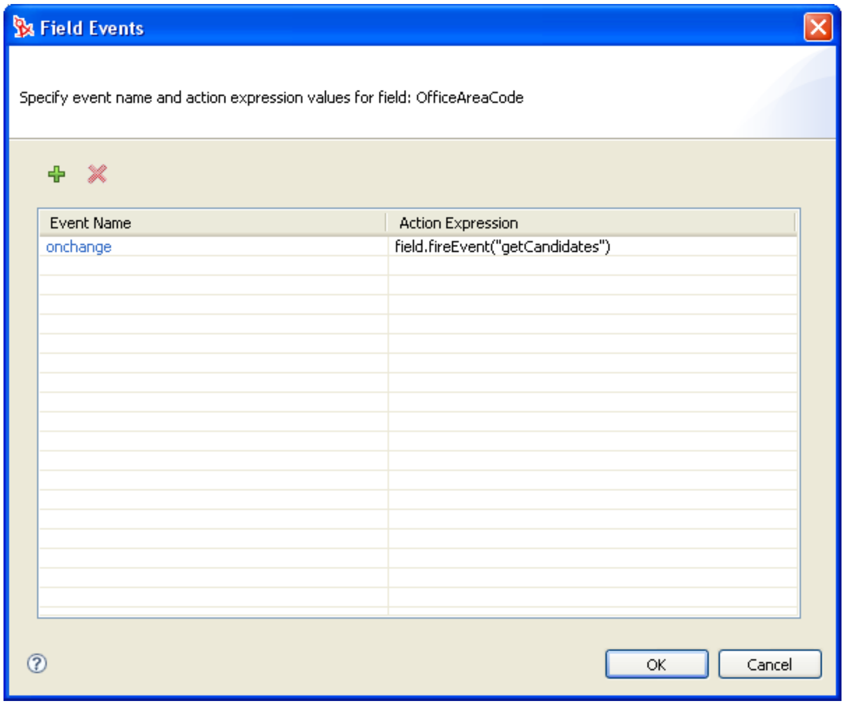


| Form Field Name | Data Type | Control Type | Linebreaks |
|---|---|---|---|
| title | string | Title | 1 |
| OfficeAreaCode | string | Text | 1 |
| title2 | string | Title | 1 |
| Candidates | dn | PickList | 1 |
| title3 | string | Title | 1 |

| Property | Value |
|---|---|
| Display Label | OfficeAreaCode |
| Editable | true |
| Required | false |
| Visible | true |
| Multivalued | false |
| Events | onchange |
| Field width in pixels | 200 |
| lower bound (for Numbers only) | |
| Maximum length | 3 |
| Minimum length | 3 |
| Number of characters allowed | 40 |
| Tooltip | |
| Upper bound (for Numbers only) | |
| Validation Mask(regular expression) | |

Notice that the Text control defines an **onchange** event. The **onchange** event fires when the user tabs from the Text control. The **onchange** event fires the **getCandidates** custom event as shown in Figure 5-29.

*Figure 5-29 OnChange Event Properties*

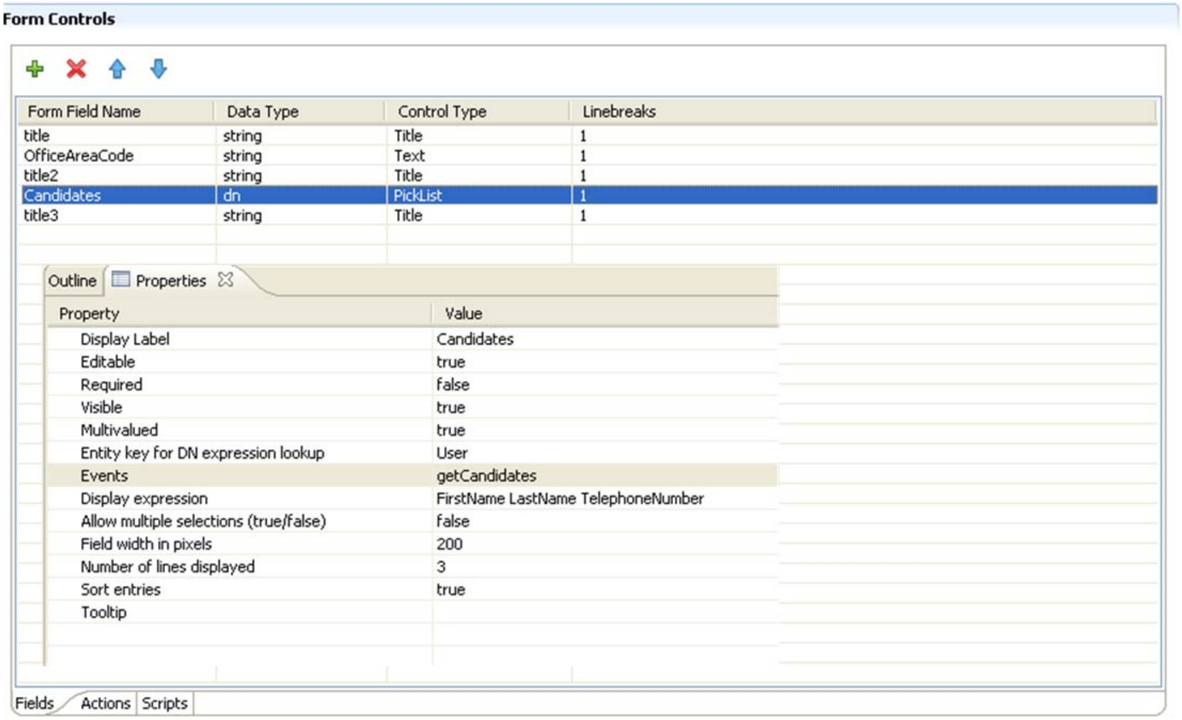


Specify event name and action expression values for field: OfficeAreaCode

| Event Name | Action Expression |
|---|---|
| onchange | field.fireEvent("getCandidates") |

The **getCandidates** event is defined as a property on the **Candidates** Picklist control.

*Figure 5-30 Candidates PickList Properties*

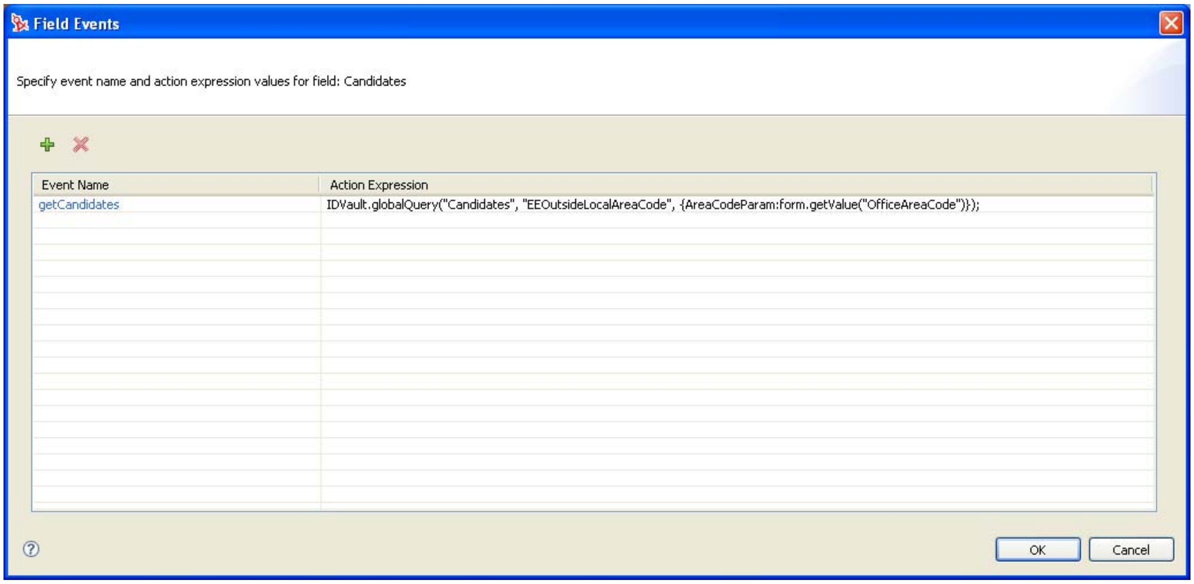


| Form Field Name | Data Type | Control Type | Linebreaks |
|---|---|---|---|
| title | string | Title | 1 |
| OfficeAreaCode | string | Text | 1 |
| title2 | string | Title | 1 |
| Candidates | dn | PickList | 1 |
| title3 | string | Title | 1 |

| Property | Value |
|---|---|
| Display Label | Candidates |
| Editable | true |
| Required | false |
| Visible | true |
| Multivalued | true |
| Entity key for DN expression lookup | User |
| Events | getCandidates |
| Display expression | FirstName LastName TelephoneNumber |
| Allow multiple selections (true/false) | false |
| Field width in pixels | 200 |
| Number of lines displayed | 3 |
| Sort entries | true |
| Tooltip | |

When the event is fired, the **getCandidates** event performs an action expression that calls the **globalQuery()** method (as shown in Figure 5-31). This method populates the value of the **Candidates** PickList control with the results of the query called **EEOutsideLocalAreaCode** (defined in Figure 5-27 on page 145). It passes the value of the **OfficeAreaCode** text field as the query's input parameter **AreaCodeParam**.

***Figure 5-31*** *GetCandidates Event*



Specify event name and action expression values for field: Candidates

| Event Name | Action Expression |
|---|---|
| getCandidates | IDVault.globalQuery("Candidates", "EEOutsideLocalAreaCode", {AreaCodeParam:form.getValue("OfficeAreaCode")}); |

# Printing Forms

You can add a **Print** button to a request form by using JavaScript.

**TIP:** Approval forms automatically contain a **Print** button.

To add the ability to print a request form, add the URL to the JavaScript library `PrintForm.js`. The library is located in the User Application WAR at this URL: `./javascript/JUICE/form/PrintForm.js`. Two techniques for including a print button are described below:

- To display a print preview popup when submitting a request form (after data is validated), add the following to the form onload event:

  ```
  form.interceptAction("SubmitAction", "around",
        function (invocation)

          {var pf = new PrintForm("SubmitAction");
           pf.printFormInterceptor(invocation);
         } );
  ```

- To add a **Print** button next to one of the fields on your form, add the following script to the field's onload event:

  ```
  var ctrl = JUICE.UICtrlUtil.getControl(field.getName());
  var btn = JUICE.UICtrlUtil.addButton(ctrl, "printid", "Print",    "Print",
  "javascript:var p = new PrintForm();
  p.printFormAfterValidation(\"printid\");");
  ```

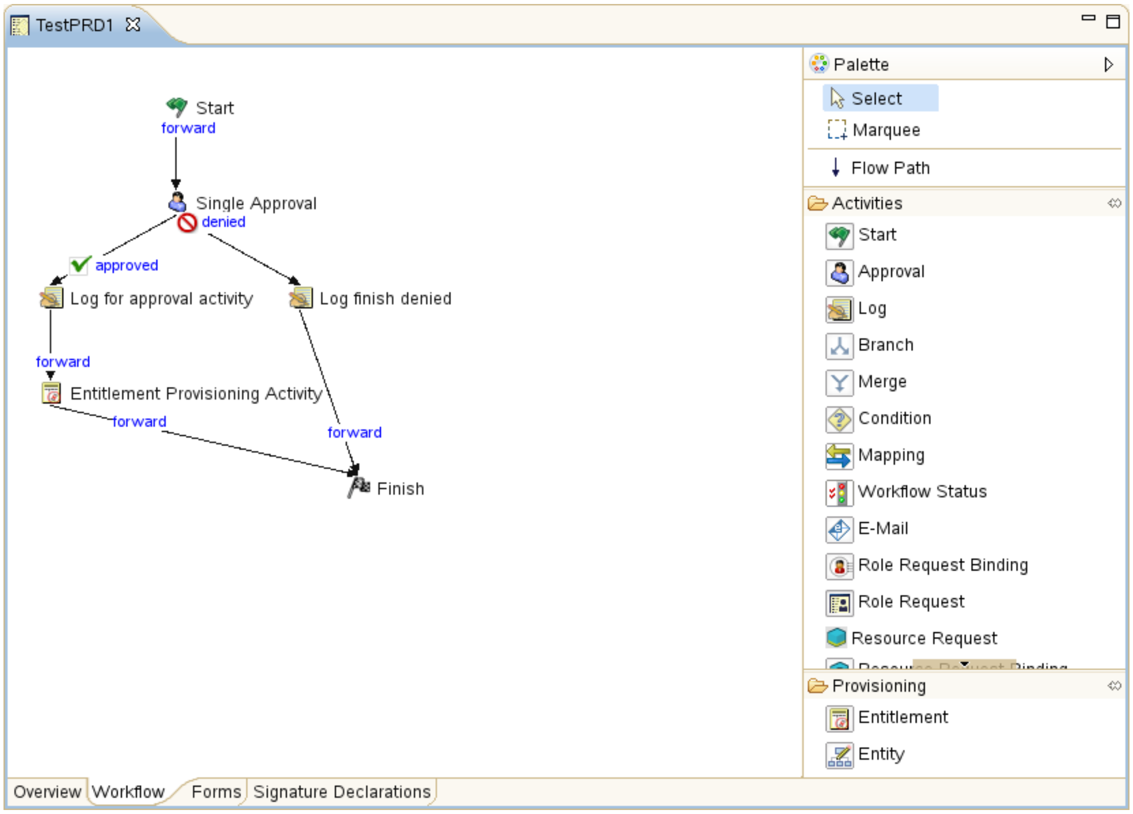
# 6 Creating the Workflow for a Provisioning Request Definition

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

## 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 6-1*** *Workflow Page*



### 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 activity and Finish activity.

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

*Table 6-1 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.<br>Use the Activity Id property to specify a meaningful value for the Activity Id. By default, the value is **ActivityNN** where *NN* is a unique number (associated with the order in which the activity was added to the palette). When 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, turn on the display of activity IDs in order to locate the activity on the canvas.<br>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. If no activity is selected, it displays the Properties view for the Workflow itself. |
| **Show Data Item Mapping** | Displays the Data Item Mapping view for the selected activity. |
| **Show E-Mail Notification** | Displays the E-Mail Notification view for the selected activity. |

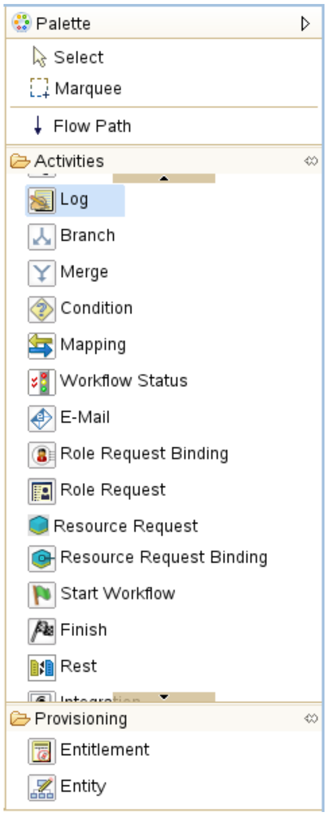
You can use the **Zoom** and **Scale** sliders on the toolbar to make it easier to view the workflow:

- **Zoom:** Increases or decreases the magnification of the workflow display. You can make portions of the workflow display larger and view more detail, or make the workflow display smaller and view more of the workflow. Click the rectangle to the right of the **Zoom** slider to return to 100% magnification.
- **Scale:** Increases or decreases the spacing between items in the workflow display. For example, if your workflow has items with many flowpaths between them, you can increase the scale to make it easier to see individual flow paths. Click the rectangle to the right of the **Scale** slider to return to 100% scale.

# Palette

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

*Figure 6-2 Workflow Palette*



The palette includes the following tools:

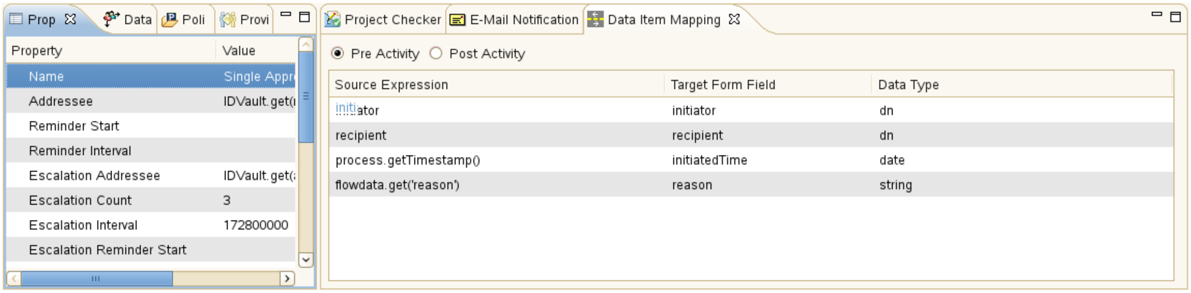
***Table 6-2** Workflow Palette*

| 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>When multiple items are selected, only the properties for the first item selected are displayed in the Properties view (see "Views" on page 152 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 "Adding Flow Paths" on page 158. |
| Activities (for example, Start, Approval, Log) | Inserts the selected activity into the workflow. For information about adding activities, see "Adding Activities to a Workflow" on page 153. For detailed descriptions of the activities, see Chapter 7, "Workflow Activity Reference," on page 187. |

# Views

The Workflow page also includes the Properties, Data Item Mapping, and E-Mail Notification views:

***Figure 6-3** Workflow Views*



| Property | Value |
|---|---|
| Name | Single Appr |
| Addressee | IDVault.get( |
| Reminder Start | |
| Reminder Interval | |
| Escalation Addressee | IDVault.get( |
| Escalation Count | 3 |
| Escalation Interval | 172800000 |
| Escalation Reminder Start | |

● Pre Activity ○ Post Activity

| Source Expression | Target Form Field | Data Type |
|---|---|---|
| initiator | initiator | dn |
| recipient | recipient | dn |
| process.getTimestamp() | initiatedTime | date |
| flowdata.get('reason') | reason | string |

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:

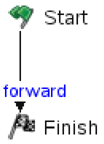
***Table 6-3** Views for Activities*

| Activity | Properties | E-Mail Notification | Data Item Mapping |
|---|---|---|---|
| Start | X | | X |
| Approval | X | X | X |
| Log | X | | |
| Branch | X | | |

| Activity | Properties | E-Mail Notification | Data Item Mapping |
|---|---|---|---|
| Merge | X | | |
| Condition | X | | |
| Mapping | X | | X |
| Workflow Status | X | | X |
| Email | X | X | |
| Role Binding | X | | |
| Resource Request | X | | |
| Role Request | X | | |
| Start Workflow | X | | X |
| Finish | X | X | |
| Rest | X | | X |
| Integration | X | | X |
| Entitlement | X | | X |
| Entity | X | | X |

# 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 activity and Finish activity, these activities are added to the canvas automatically. The Start Activity is connected to the Finish Activity with a forward link.

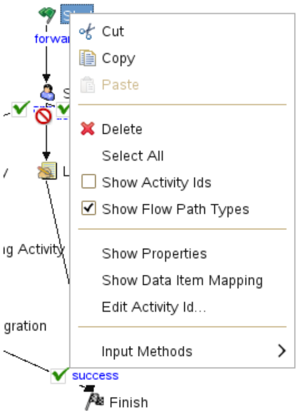
**2** To add an activity to the workflow, click the icon for the desired activity in the palette and drag the icon onto the workspace.

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 “Adding Flow Paths” on page 158. After you have added an activity to the workflow, you should set the properties of the activity (see “Setting the General Properties of an Activity” on page 154). For detailed information about configuring the different types of activities, see Chapter 7, “Workflow Activity Reference,” on page 187 and Chapter 8, “Working with Integration Activities,” on page 235.

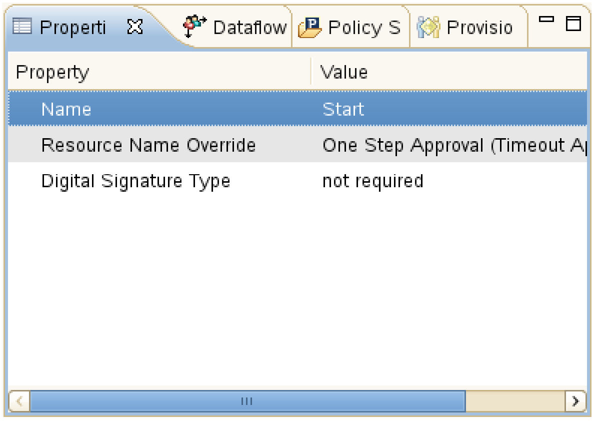
## Setting the General Properties of an Activity

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

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



The Properties view is displayed:

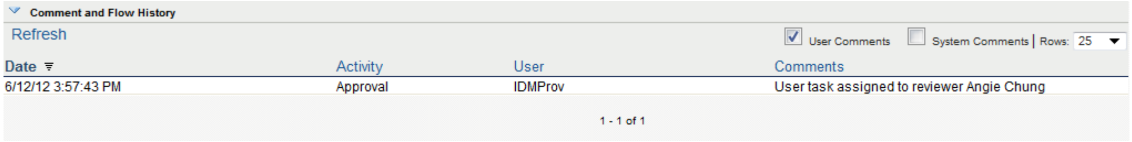


| Property | Value |
|---|---|
| Name | Start |
| Resource Name Override | One Step Approval (Timeout A |
| Digital Signature Type | not required |

2 Click in the column for a property to set the property. For information about the properties for each activity, see Chapter 7, "Workflow Activity Reference," on page 187.

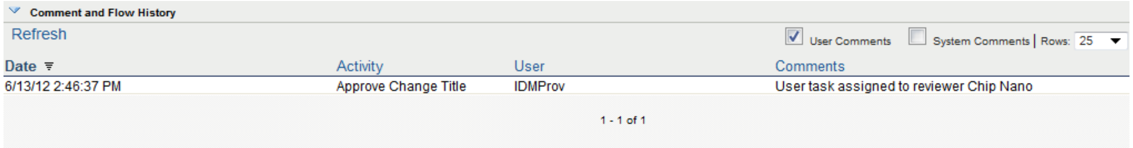
Each activity has a default name. We strongly recommend that you replace the default names of activities with a descriptive Activity Id that describe the specific purpose of the activity in the workflow. This makes it easier to understand the workflow when you look at the graphical display of the workflow. It also makes comments displayed in the User Application easier to understand. For example, the following figures show comments in the User Application using default IDs and descriptive IDs.

*Figure 6-4 Activity in User Comments Using Default Name*



| Date | Activity | User | Comments |
|---|---|---|---|
| 6/12/12 3:57:43 PM | Approval | IDMProv | User task assigned to reviewer Angie Chung |

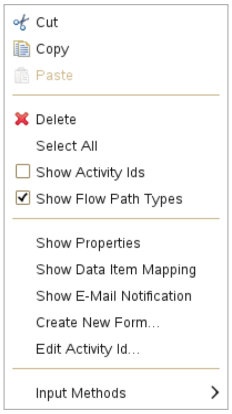
*Figure 6-5 Activity in User Comments Using Descriptive Name*



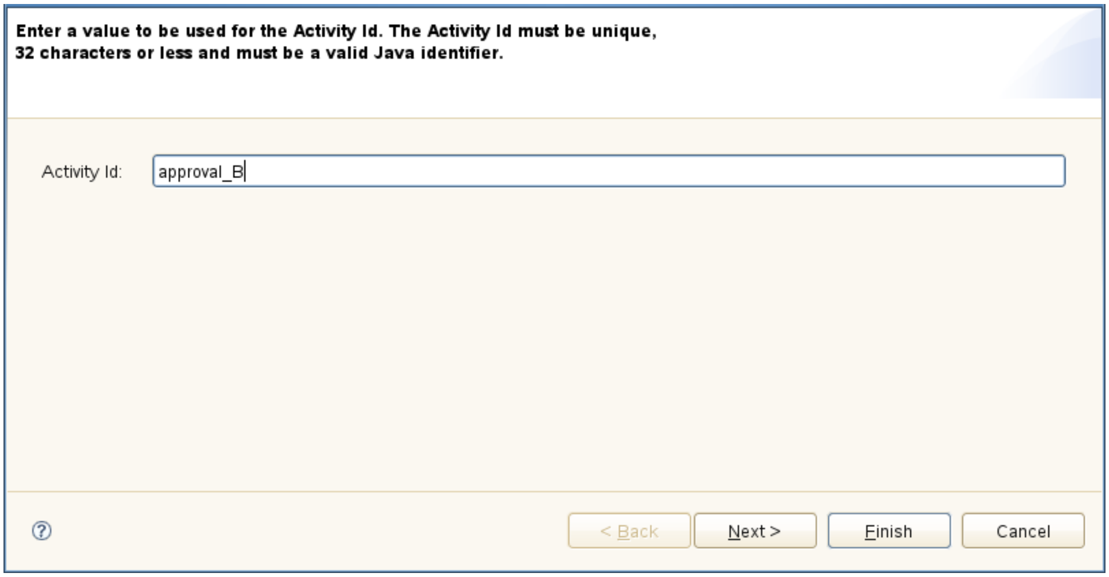
| Date | Activity | User | Comments |
|---|---|---|---|
| 6/13/12 2:46:37 PM | Approve Change Title | IDMProv | User task assigned to reviewer Chip Nano |

To change the Activity Id:

1 Right-click the activity icon for which you want to change the Activity Id and select **Edit Activity Id**.



The Edit Activity Id wizard displays.

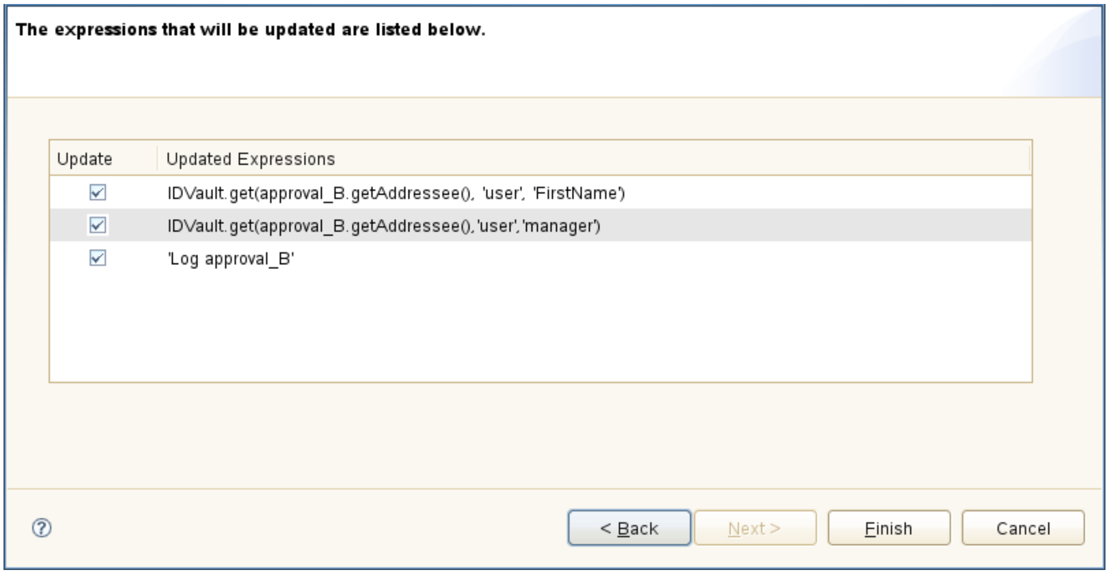


**Enter a value to be used for the Activity Id. The Activity Id must be unique, 32 characters or less and must be a valid Java identifier.**

Activity Id: approval_B

2 Type the Activity Id name you want to use. The name can include letters, numbers, and the underscore (_) character.

3 If the activity was not used to define any expressions, click **Finish**. If the activity was used in expressions, click **Next**. The wizard displays the expressions where the Activity Id should be updated with the new name.



The expressions that will be updated are listed below.

| Update | Updated Expressions |
|---|---|
| ☑ | IDVault.get(approval_B.getAddressee(), 'user', 'FirstName') |
| ☑ | IDVault.get(approval_B.getAddressee(),'user','manager') |
| ☑ | 'Log approval_B' |

4 Review the items displayed in the panel. For items that you do not want updated, deselect the checkbox in the Update column.

5 Click **Finish** when complete.

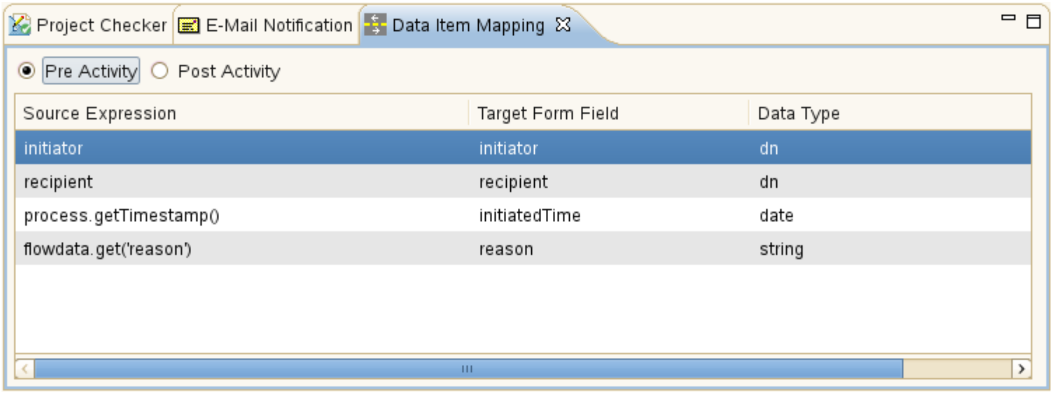
# Defining the Data Item Mappings

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

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

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

The Data Item Mapping view is displayed:



● Pre Activity ○ Post Activity

| Source Expression | Target Form Field | Data Type |
|---|---|---|
| initiator | initiator | dn |
| recipient | recipient | dn |
| process.getTimestamp() | initiatedTime | date |
| flowdata.get('reason') | reason | string |

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 detailed information about data item mapping for the different types of activities, see Chapter 7, "Workflow Activity Reference," on page 187.

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 "General Form Control Properties" on page 105 for information about setting the properties of form fields.

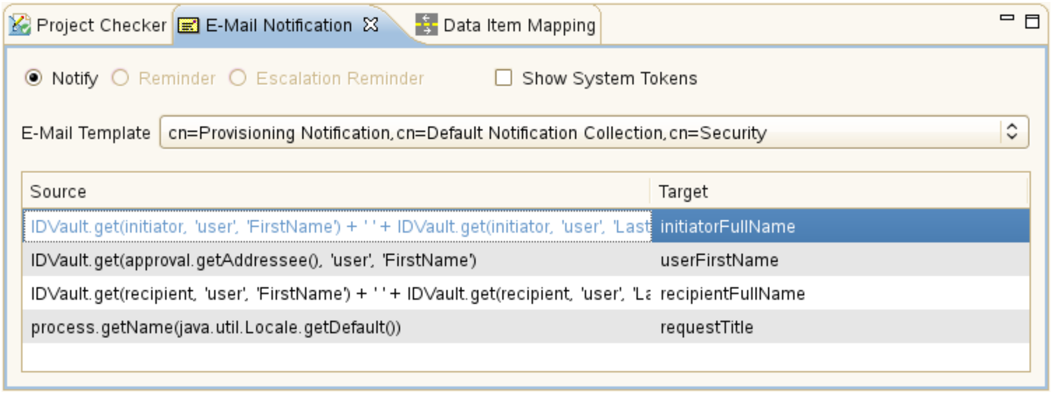
# Defining the E-Mail Notification Settings

You use the **E-Mail Notification** view to select an email template, and to specify expressions to provide values for named parameters included in the email template. Emails are sent when a new Approval activity starts (to notify approvers 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 E-Mail Notification** from the menu.

You can also display the **E-Mail Notification** tab by selecting **Show E-Mail Notification** from the **PRD** menu.

The **E-Mail Notification** view is displayed:



● Notify ○ Reminder ○ Escalation Reminder □ Show System Tokens

E-Mail Template: cn=Provisioning Notification,cn=Default Notification Collection,cn=Security

| Source | Target |
|---|---|
| IDVault.get(initiator, 'user', 'FirstName') + ' ' + IDVault.get(initiator, 'user', 'Last | initiatorFullName |
| IDVault.get(approval.getAddressee(), 'user', 'FirstName') | userFirstName |
| IDVault.get(recipient, 'user', 'FirstName') + ' ' + IDVault.get(recipient, 'user', 'La | recipientFullName |
| process.getName(java.util.Locale.getDefault()) | requestTitle |

2 Click the **E-Mail Template** field, then select an email template from the list of defined templates.

**Editing an email template:** You can edit an email 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**.

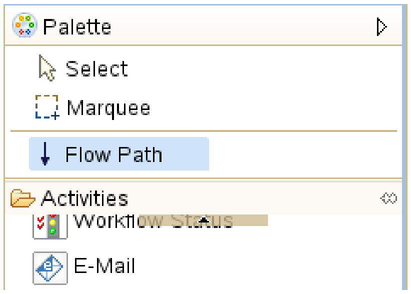
**Localized email templates:** By default, Designer displays the default email notification templates. When you select a default template, the email is in the user's default language (if the default is a supported language). You can set the **Show all localized e-mail templates** preference to True so that Designer also allows you to select from the list of localized email templates. The localized templates have the same name as the default, but the Java language code is appended to the name of the email template. For example, `cn=Provisioning Notification Activity_es, cn=Default Notification Collection, cn=security` indicates this is the Spanish language version of this template. When you select a localized template, the email is in the language of the template regardless of the user's default language.

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

See Chapter 7, "Workflow Activity Reference," on page 187 for information about email notification settings.

# Adding 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**.

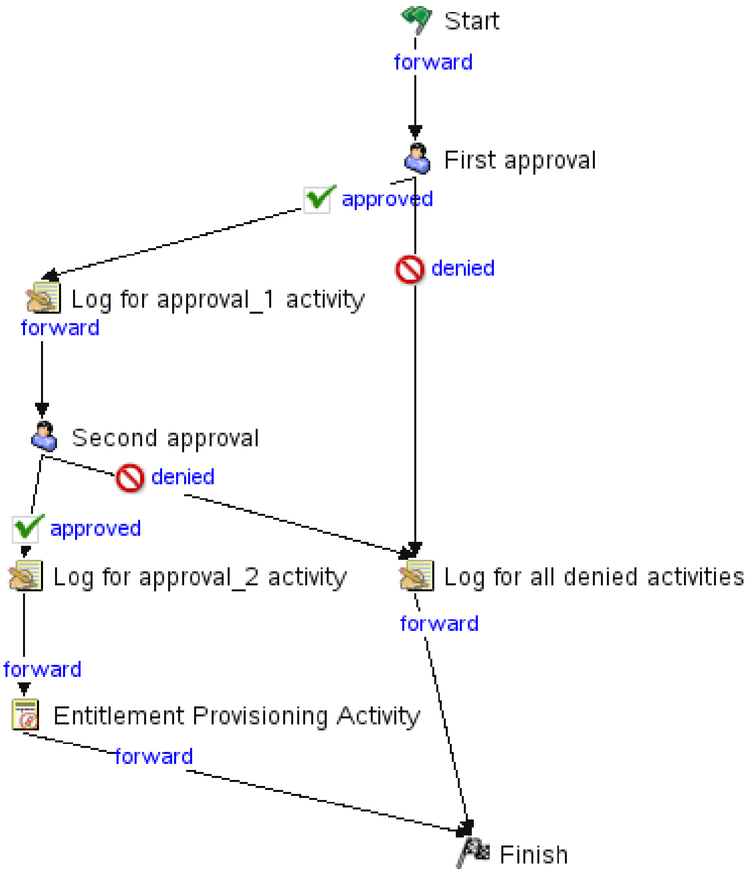
For information about configuring flow paths, see "Configuring Flow Paths" on page 158.

# Configuring Flow Paths

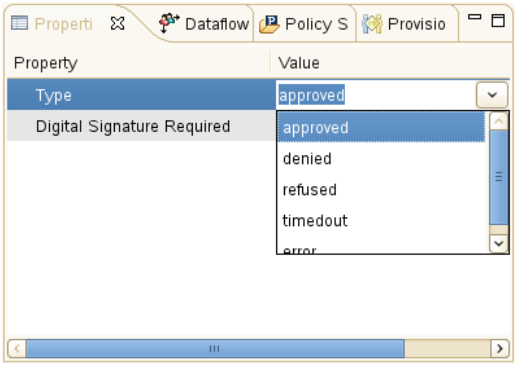
After you have added a flow path to a workflow diagram, you can specify the path type. For details on adding flow paths to a workflow, see "Adding Flow Paths" on page 158.

To configure a flow path:

1 Click the flow path in the workflow diagram.


Start
forward
First approval
approved
denied
Log for approval_1 activity
forward
Second approval
denied
approved
Log for approval_2 activity
Log for all denied activities
forward
forward
Entitlement Provisioning Activity
forward
Finish

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



| Property | Value |
|---|---|
| Type | approved |
| Digital Signature Required | |

approved
denied
refused
timedout
error

The flow path types are described in the following table:

| 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. |
| false | Determines what happens when a conditional expression evaluates to False.<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**.

# 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.

**NOTE:** You can validate a provisioning request definition before you deploy it. For more information, see “Validating Provisioning Objects” on page 28.

## Rules for Activities

When adding activities to a workflow, follow these rules:

- A workflow must have only one Start activity and one Finish activity.

- A workflow can have zero or more of the following activity types:

  Approval activity
  Log activity
  Branch activity
  Merge activity
  Condition activity
  Mapping activity
  Workflow Status
  Email activity
  Role Request activity
  Role Request Binding activity
  Start Workflow
  Rest activity
  Integration activity
  Entitlement activity
  Entity activity
- Each Branch activity must have a corresponding Merge activity.
- The role activities (Role Request and Role Request Binding) can only be used for workflows that support roles.
- The resource activities (Resource Request and Resource Request Binding) can only be used for workflows that support resources.
- To ensure that the provisioning step is performed, a workflow must have at least one Entitlement activity or Entity activity.

## Rules for Flow Paths

When adding flow paths to a workflow, 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.
- Flows paths out of Role Binding activities must connect to the Finish activity.
- There can be between one and three flow paths out of the Rest activity. The valid flow path types are forward, error, and timeout.

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

*Table 6-4 Number of Flow Paths Permitted for Each Activity*

| Activity | Inbound Paths | Outbound Paths |
|---|---|---|
| Start | 0 | 1 Must always be forward. |
| Approval | 1 to n | 1 to 5 Approved, denied, refused, timedout, or error |
| Log | 1 to n | 1 Must always be forward |
| Branch | 1 to n | 1 to n |
| Merge | 1 to n | 1 Must always be forward |
| Condition | 1 to n | 2 to 3 True and false are required; error is optional |
| Mapping | 1 to n | 1 |
| Workflow Status | 1 to n | 1 Must always be forward |
| Email | 1 to n | 1 Must always be forward |
| Role Request Binding | 1 | 1 Must always be forward and connect to Finish activity |
| Role Request | 1 | 2 forward, error |
| Resource Request Binding | 1 | 1 Must always be forward and connect to Finish activity |
| Resource Binding | 1 | 2 forward, error |
| Start Workflow | 1 | 2 forward, error |
| Finish | 1 to n | 0 |
| Rest | 1 to n | |
| Integration | 1 to n | 1 to 4 Success, timedout, error, fault |
| Entitlement | 1 to n | 1 Must always be forward |
| Entity | 1 to n | 1 Must always be forward |

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

*Table 6-5 Flow Path Types Allowed for Each Activity*

| Activity | Forward | Approved | Denied | Refused | Timedout | True | False | Error | Success | Fault |
|---|---|---|---|---|---|---|---|---|---|---|
| Start | Source | | | | | | | | | |
| Approval | Target | Source/ Target | Source/ Target | Source/ Target | Source/ Target | Target | Target | Source /Target | | |
| Log | Source/ Target | Target | Target | Target | Target | Target | Target | Target | | |
| Branch | Source/ Target | Target | Target | Target | Target | Target | Target | Target | | |
| Merge | Source/ Target | Target | Target | Target | Target | Target | Target | Target | | |

| Activity | Forward | Approved | Denied | Refused | Timedout | True | False | Error | Success | Fault |
|---|---|---|---|---|---|---|---|---|---|---|
| Condition | Target | Target | Target | Target | Target | Source/Target | Source/Target | Source/Target | | |
| Mapping | Source | Target | Target | Target | Target | Target | Target | Target | | |
| Workflow Status | Source/Target | | | | | | | | | |
| Email | Source/Target | | | | | | | | | |
| Role Request Binding | Source/Target | Target | Target | Target | Target | | | Target | | |
| Role Request | Source/Target | Target | Target | Target | Target | | | Source/Target | | |
| Resource Request Binding | Source/Target | Target | Target | Target | Target | | | Target | | |
| Resource Request | Source/Target | Target | Target | Target | Target | | | Source/Target | | |
| Start Workflow | Source/Target | Target | Target | Target | Target | | | Source/Target | | |
| Finish | Target | Target | Target | Target | Target | Target | Target | Target | | |
| Rest | Source/Target | Target | Target | Target | Source/Target | Target | Target | Source/Target | Target | Target |
| Integration | Source/Target | Target | Target | Target | Source/Target | Target | Target | Source/Target | Source | Source |
| Entitlement | Source/Target | Target | Target | Target | Target | Target | Target | Target | | |
| Entity | Source/Target | Target | Target | Target | Target | Target | Target | Target | | |

## Understanding Workflow Data

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

- “Data Objects and Variables” on page 164
- “Creating New Data Items” on page 165
- “Modifying Data Items” on page 166
- “Working with Complex Data Item Mappings” on page 166
- “Moving Form Control Data to Flowdata” on page 166
- “Moving Flowdata to Form Controls” on page 167
- “About Mapping Activity Operations” on page 167

## Data Objects and Variables

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. After 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 6-6*** *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 6-7*** *Data Variables in a Workflow*

| Object | Variable | Description |
|---|---|---|
| process | approvalStatus | The current status of the process. |
| | category | The provisioning category (for example, Entitlements) selected by the person who initiated the request. |
| | container dn | The distinguished name of the container defined for the user application at install time. |
| | description | The description of the provisioning request definition. |
| | group container dn | The distinguished name of the group container defined for the user application at install time. |
| | id | The unique IDVault ID (CN) of the provisioning request definition. |
| | initiator | The distinguished name of the person who initiated the request. |
| | locale | The current locale. |
| | name | The workflow process name. |

| Object | Variable | Description |
|---|---|---|
| | provisioning driver dn | The distinguished name of the provisioning driver defined for the user application at install time. |
| | recipient | The distinguished name of the intended target of the provisioned resource. |
| | user container dn | The distinguished name of the user container defined for the user application at install time. |
| | 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. |
| | user | The user who is associated with the current activity. |
| | workId | The system generated unique ID of the current workflow activity. |
| *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.<br>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 sign 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:

```
Number(flowdata.get('cost')) + Number(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 6-8*** *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 workflows 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 271.

## Moving Form Control Data to Flowdata

All form controls you create (except for DNDisplay) 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. 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.

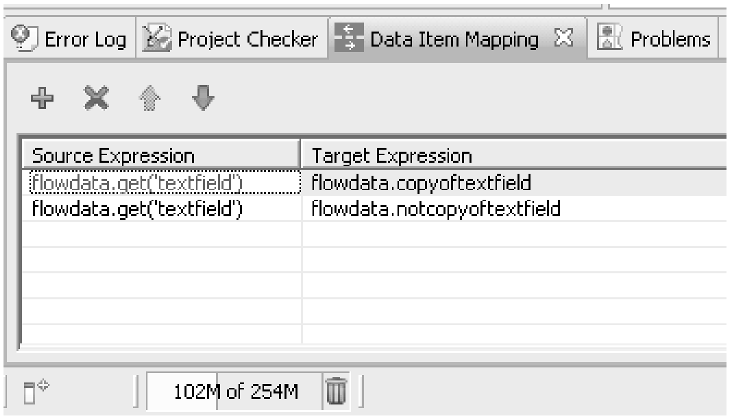
## About Mapping Activity Operations

In the mapping activity, the source expressions are evaluated before they are assigned to the target expression. If the source expression does not exist prior to the mapping activity, no value is assigned to the target expression.

For example, if flowdata.get(“textfield”) maps to flowdata.copyoftextfield and flowdata.get("copyoftextfield") maps to flowdata.copyoftextfield2, the value of flowdata.copyoftextfield2 is empty at the end of the mapping activity because the value of the flowdata.copyoftextfield is assigned only after the mapping activity.

To assign values to the target expression, you can use either of these options:

- Multiple mapping-activities.
- Single mapping-activity but repeat the source expression.



| Source Expression | Target Expression |
|---|---|
| flowdata.get('textfield') | flowdata.copyoftextfield |
| flowdata.get('textfield') | flowdata.notcopyoftextfield |

When you repeat the source expression, the flowdata.get(“textfield”) maps to flowdata.copyoftextfield and flowdata.get("textfield") maps to flowdata.copyoftextfield2.

# Guidelines for Creating Roles Based Workflows

Roles based workflows must follow the same guidelines outlined in “Guidelines for Creating Workflows” on page 160. In addition, roles based workflows have their own unique requirements. They are described in the following sections:

- “About Role Approval Workflows” on page 168
- “Writing Custom Role Workflows” on page 170
- “About Separation of Duties Approval Workflows” on page 174
- “Customizing the Standard Separation of Duties Workflow” on page 177

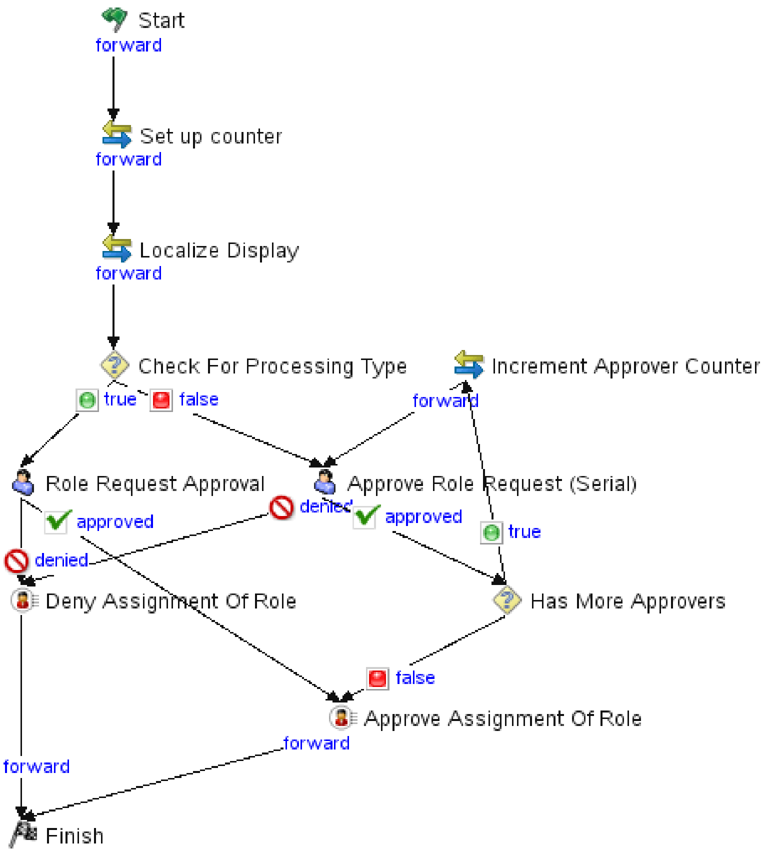
## About Role Approval Workflows

Role approval workflows are specialized workflows that provide support for role approval and revocation on the User Application’s **Roles** tab. The Roles Based Provisioning Module includes a read-only Role Approval workflow (named Role Approval) whose design pattern supports:

- The ability to process role approvals in either serial or quorum mode.
- The retrieval of approver DNs from the role object (nrfRequest). If you create a custom workflow, the approvers must be defined in the workflow. However, this might lead to addressee evaluation problems and less security concerning who can approve a role.
- The ability to display the role using localized display names.
- All nrfRequest object mappings for request and approval forms.
- Logging and reporting functions.
- Read-only display of request information. The role approval workflow does not allow changes to the request. Approvers have only the ability to approve or deny the role request.
- An email notification is sent to all approvers of role approval workflows. A completed notification email is sent upon completion of the role approval workflow. The recipient email address is used when the workflow is intended to be assigned to a user identity.

This pattern is shown in Figure 6-6.

*Figure 6-6 Default Role Approval Workflow*



The components of this workflow, and their responsibilities are summarized in Table 6-9.

*Table 6-9 Standard Role Approval Activities*

| Activity Name | Activity Type | Description |
|---|---|---|
| Start | Start | Logical starting point for all workflows. For role approvals it must instantiate the nrfRequest object. |
| Set up counter | Mapping | Sets up the counter for the number of approvers in case the mode is Serial. |
| Localize Display | Mapping | Sets up the display labels for each of the associated display names for the user’s locale. |

| Activity Name | Activity Type | Description |
|---|---|---|
| Check for Processing Type | Condition | Determines whether the approval is a quorum condition by setting the Condition property to this ECMA expression:<br>`nrfRequest.isQuorumProcess()`<br>If the quorum condition exists, control proceeds to the Approve Role Request (Quorum). If the quorum condition does not exist, control proceeds to the Approve Role Request (Serial).<br>You specify the processing type for the role approval when you set up the Role Catalog. |
| Approve Role Request (Quorum) | Approval | This is where the decision to approve or deny the request is recorded as part of the workflow instance. The quorum condition required to make the process successful is retrieved from the nrfQuorum attribute of the nrfRequest object. |
| Approve Role Request (Serial) | Approval | This is where the decision to approve or deny the request is recorded as part of the data flow associated with the workflow instance.<br>The workflow loops through the list of approvers found in the nrfRequest object. The request is approved if all approvers in the serial process approve the request. The request is denied upon the first rejection from an approver in the serial process |
| Deny Assignment of Role | Role Binding | Changes the deny attribute in the nrfRequest object to true. |
| Approve Assignment of Role | Role Binding | Changes the approve attribute in the nrfRequest object to true. |
| Finish | Finish | Logical end point of all workflows. |

To use the standard Role Approval workflow in your user application, you must specify your own users as Trustees. For information on setting the Trustees property, see “Modifying Settings of a Provisioning Request Definition” on page 86.

## Writing Custom Role Workflows

If the standard role approval workflow does not support your business needs, and cannot be customized to do so, you can write your own. At a minimum, a custom role approval workflow must:

- Contain two Role Binding activities

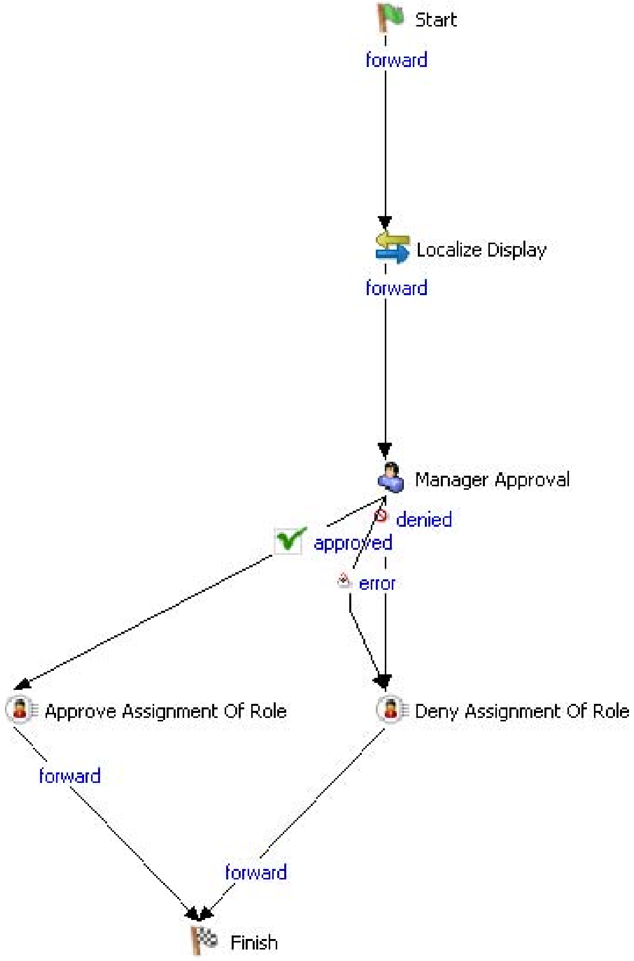
  One Role Binding activity must be set to approved and the other set to denied. You must link each of the Role Binding activities to the Finish activity. If the workflow does not meet this requirement, it is invalid, and Designer prevents you from deploying it. The Role Service driver needs these values to set the status for the workflow and to then apply the logic to associate the role to the identity.

- Contain the following control in the request form:
  - Form Field Name: nrfRequestDN
  - Data Type: Role Request
  - Control Type: Text
- Instantiate the nrfRequestDN in the Pre Activity Data Item Mapping.
- Contain the following in the Post Activity Data Item Mapping:
  - Source Form Field: nrfRequestDN
  - Target Expression: flowdata.nrfRequest/DN
  - Data Type: dn
- Not contain the following ECMA expressions in the Data Item Mapping or Properties definitions because they might return null:
  - `getApprovalDN()`
  - `getAllApproversDN()`
  - `getAllSodApproversDN()`

  Because Designer and the User Application user interface do not allow entry of approvers for custom role approval workflows, you must specify the approvers in the workflow itself. Therefore, if you create a custom workflow based on a copy of the Role Approval or SoD Conflict Approval provisioning request definitions, you must remove the ECMA methods from Data Item Mapping or Properties definitions.

In the following example, a user requests a role and the user’s manager approves it.

*Figure 6-7 Sample Custom Role Approval Workflow*



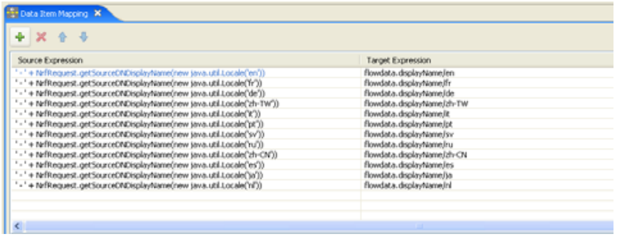
The components of this workflow, and their responsibilities are summarized in Table 6-10.

***Table 6-10** Sample Custom Workflow Components*

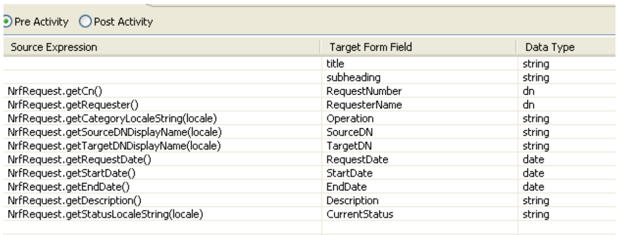
| Activity Name | Activity Type | Description |
|---|---|---|
| Start | Start | Logical starting point of all workflows. |
| Localize Display | Mapping | Sets up the display labels for the user's locale. |
| Manager Approval | Approval | This is where the decision to approve or deny the request is recorded as part of the workflow instance. The role request approval is needed only by the requestor's manager. |
| Approve Assignment of Role | Role Binding | Changes the approve attribute in the nrfRequest object to true. |
| Deny Assignment of Role | Role Binding | Changes the deny attribute in the nrfRequest object to true. |
| Finish | Finish | Logical end point of all workflows. |

The data item mapping for the sample custom role approval workflow is defined in Table 6-11

***Table 6-11** Sample Custom Role Approval Workflow Data Item Mapping*

| Activity Name | Property Type | Property Value |
|---|---|---|
| Start | Data Item Pre Activity | Source Expression: None |
| | | Target Form Field: nrfRequestDN |
| | | Data Type: dn |
| | Data Item Post Activity | Source Form Field: nrfRequestDN |
| | | Target Expression: flowdata.nrfRequest/DN |
| | | Data Type: DN |
| Localize Display | Data Item Source and Target mapping |  |
| Manager Approval | Addressee Property | `Addressee IDVault.get(recipient,'user','manager')` |

| Source Expression | Target Expression |
|---|---|
| '-' + NrfRequest.getSourceDNDisplayName(new java.util.Locale('en')) | flowdata.displayName/en |
| '-' + NrfRequest.getSourceDNDisplayName(new java.util.Locale('fr')) | flowdata.displayName/fr |
| '-' + NrfRequest.getSourceDNDisplayName(new java.util.Locale('de')) | flowdata.displayName/de |
| '-' + NrfRequest.getSourceDNDisplayName(new java.util.Locale('zh-TW')) | flowdata.displayName/zh-TW |
| '-' + NrfRequest.getSourceDNDisplayName(new java.util.Locale('it')) | flowdata.displayName/it |
| '-' + NrfRequest.getSourceDNDisplayName(new java.util.Locale('pt')) | flowdata.displayName/pt |
| '-' + NrfRequest.getSourceDNDisplayName(new java.util.Locale('sv')) | flowdata.displayName/sv |
| '-' + NrfRequest.getSourceDNDisplayName(new java.util.Locale('ru')) | flowdata.displayName/ru |
| '-' + NrfRequest.getSourceDNDisplayName(new java.util.Locale('zh-CN')) | flowdata.displayName/zh-CN |
| '-' + NrfRequest.getSourceDNDisplayName(new java.util.Locale('es')) | flowdata.displayName/es |
| '-' + NrfRequest.getSourceDNDisplayName(new java.util.Locale('ja')) | flowdata.displayName/ja |
| '-' + NrfRequest.getSourceDNDisplayName(new java.util.Locale('nl')) | flowdata.displayName/nl |

<table>
<tr><th>Activity Name</th><th>Property Type</th><th>Property Value</th></tr>
<tr><td></td><td>Data Item Pre Activity</td><td>
● Pre Activity ○ Post Activity
<table>
<tr><th>Source Expression</th><th>Target Form Field</th><th>Data Type</th></tr>
<tr><td></td><td>title</td><td>string</td></tr>
<tr><td></td><td>subheading</td><td>string</td></tr>
<tr><td>NrfRequest.getCn()</td><td>RequestNumber</td><td>dn</td></tr>
<tr><td>NrfRequest.getRequester()</td><td>RequesterName</td><td>dn</td></tr>
<tr><td>NrfRequest.getCategoryLocaleString(locale)</td><td>Operation</td><td>string</td></tr>
<tr><td>NrfRequest.getSourceDNDisplayName(locale)</td><td>SourceDN</td><td>string</td></tr>
<tr><td>NrfRequest.getTargetDNDisplayName(locale)</td><td>TargetDN</td><td>string</td></tr>
<tr><td>NrfRequest.getRequestDate()</td><td>RequestDate</td><td>date</td></tr>
<tr><td>NrfRequest.getStartDate()</td><td>StartDate</td><td>date</td></tr>
<tr><td>NrfRequest.getEndDate()</td><td>EndDate</td><td>date</td></tr>
<tr><td>NrfRequest.getDescription()</td><td>Description</td><td>string</td></tr>
<tr><td>NrfRequest.getStatusLocaleString(locale)</td><td>CurrentStatus</td><td>string</td></tr>
</table></td></tr>
<tr><td></td><td>Data Item Post Activity</td><td>None</td></tr>
<tr><td>Approve Assignment of Role</td><td>Action Property</td><td>approved</td></tr>
<tr><td>Deny Assignment of Role</td><td>Action Property</td><td>denied</td></tr>
<tr><td>Finish</td><td></td><td>None</td></tr>
</table>

## About Separation of Duties Approval Workflows

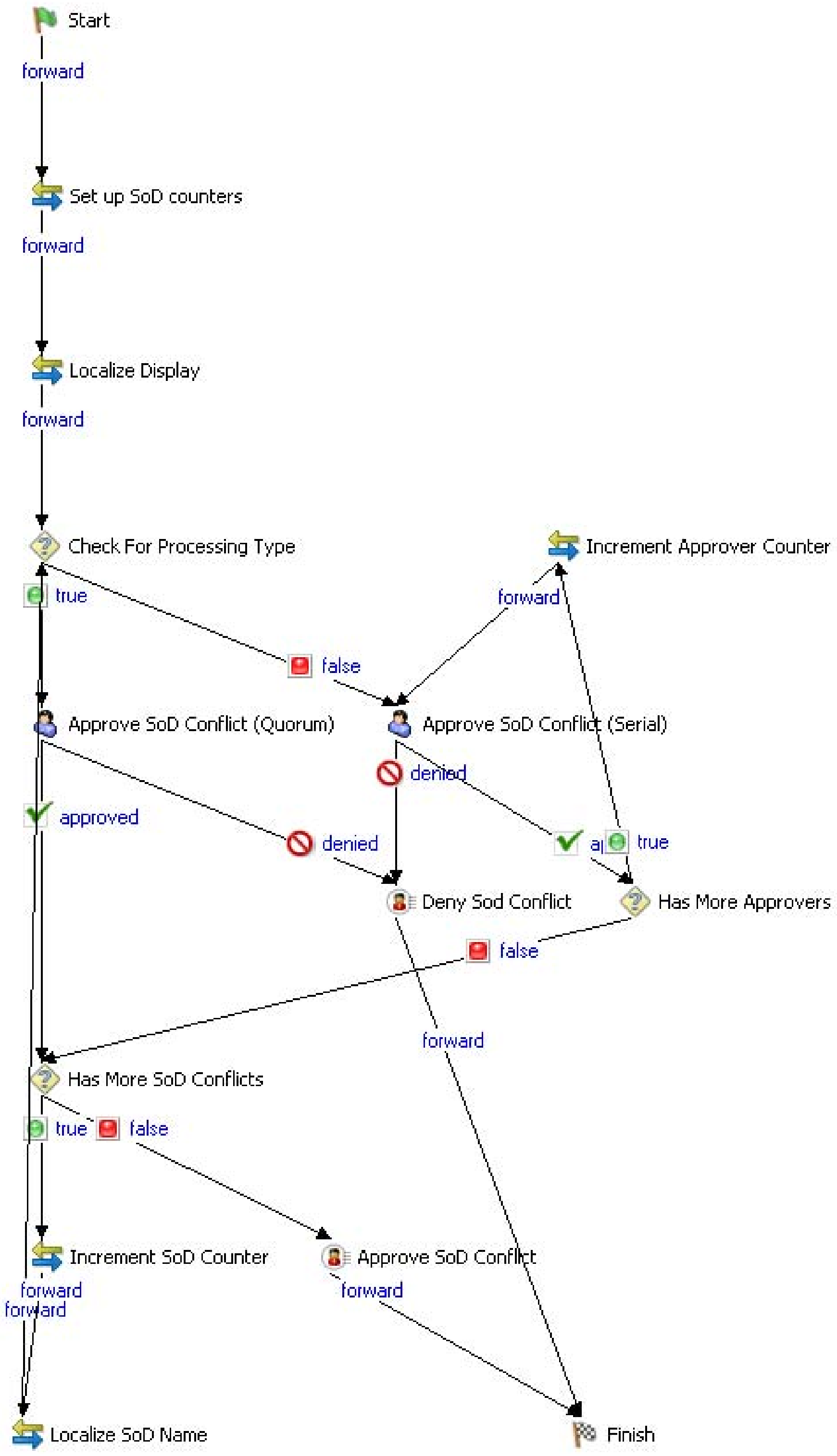
Separation of Duties approval workflows are specialized workflows that allows a Separation of Duties constraint to be overridden. The Roles Based Provisioning Module includes a read-only Separation of Duties Approval workflow (named SoD Conflict Approval) whose design pattern supports:

- The ability to process SoD conflicts in either serial or quorum mode.
- The retrieval of SoD approver DNs from the request object (nrfRequest). If you create a custom workflow, the approvers must be defined in the workflow; however, this might lead to addressee evaluation problems and less security concerning who can approve an SoD.
- The ability to display the SoD using localized display names.
- All nrfRequest object mappings for request and approval forms.
- Logging and reporting functions.
- Read-only display of requests. Approvers can only approve or deny the SoD conflict.
- An email notification is sent to all approvers per SoD conflict found for SoD workflow approvals. A completed notification email is sent upon completion of the SoD approval workflow. The recipient email address is used when the workflow is intended to be assigned to a user identity.

This pattern is shown in Figure 6-6 on page 169.

The roles subsystem allows one Separation of Duties approval flow for the Role subsystem. If you choose to use a custom SoD approval flow, make sure that it works for all SoD situations.

***Figure 6-8*** *Standard SoD Approval Workflow*



The components of the workflow are described in the following table:

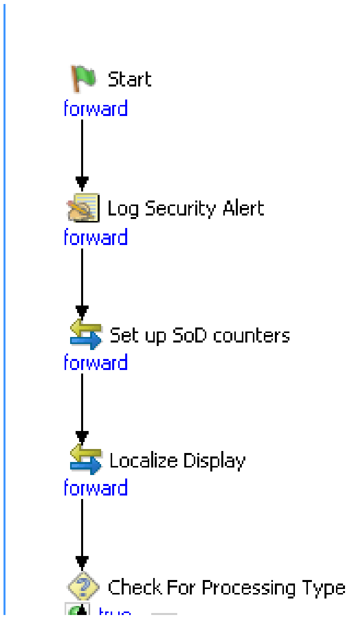
***Table 6-12** Standard SoD Constraint Exception Approval Workflow Activities*

| Activity Name | Activity Type | Description |
|---|---|---|
| Start | Start | Logical starting point of all workflows. |
| Localize Display | Mapping | Sets up the display labels for each of the associated Display Names for the user's locale for the SoD conflicting Role. |
| Localize SoD Name | Mapping | Sets up the display labels for each of the associated Display Names for the user's locale for the SoD conflicting Role. |
| Check for Processing Type | Condition | Determines whether the approval is a quorum condition by setting the Condition property to this ECMA expression:<br>`nrfRequest.isSodQuorumProcess()`<br>If the quorum condition exists, control proceeds to the Approve SoD Conflict (Quorum). If the quorum condition does not exist, control proceeds to the Approve SoD Conflict (Serial).<br>You specify the processing type for the role approval when you set up the Role Catalog. |
| Approve SoD Conflict (Quorum) | Approval | This is where the decision to approve or deny the request is recorded as part of the workflow instance. The quorum condition required to make the process successful is retrieved from the nrfQuorum attribute of the nrfRequest object. |
| Approve SoD Conflict (Serial) | Approval | This is where the decision to approve or deny the request is recorded as part of the data flow associated with the workflow instance.<br>The workflow loops through the list of approvers found in the nrfRequest object. The request is approved if all approvers in the serial process approve the request. The request is denied upon the first rejection from an approver in the serial process |
| Deny SoD Conflict | Role Binding | Changes the deny attribute in the nrfRequest object to true. |
| Approve SoD Conflict | Role Binding | Changes the approve attribute in the nrfRequest object to true. |
| Has More SoD Conflicts and Increment SoD Counter | Condition and Mapping activity | Loops through the SoD requests. |
| Finish | Finish | Logical end point of all workflows. |

## Customizing the Standard Separation of Duties Workflow

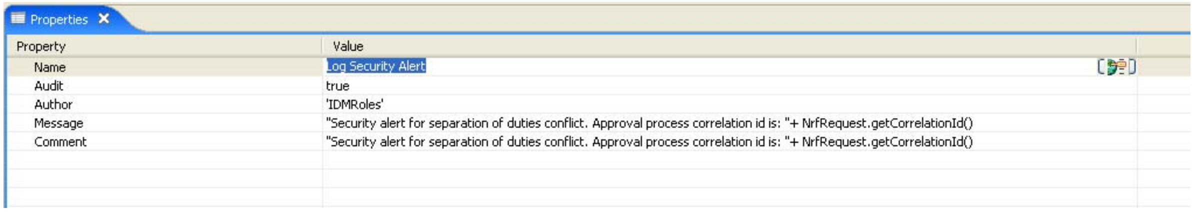
Separation of Duties conflict approval workflows are complex. Therefore, it is not recommended that you write a custom version. Rather, it is recommended that you add new activities to a copy of the standard SoD approval workflow. For example, you might want to add additional logging or messages. This example illustrates a customized workflow that includes a new logging activity.

***Figure 6-9** Adding Activities to the SoD Workflow*



The Log Activity properties are shown in Figure 6-10.

***Figure 6-10** Log Activity Properties*



| Property | Value |
|---|---|
| Name | Log Security Alert |
| Audit | true |
| Author | 'IDMRoles' |
| Message | "Security alert for separation of duties conflict. Approval process correlation id is: "+ NrfRequest.getCorrelationId() |
| Comment | "Security alert for separation of duties conflict. Approval process correlation id is: "+ NrfRequest.getCorrelationId() |

SoD Conflict approval workflows must follow the same rules as the role approval workflows as described on "Writing Custom Role Workflows" on page 170.

# Guidelines for Creating Resource Based Workflows

Resource based workflows must follow the same guidelines outlined in "Guidelines for Creating Workflows" on page 160. In addition, resource based workflows have the unique requirements described in the following sections:

- "About Resource Approval Workflows" on page 178
- "Writing Custom Resource Workflows" on page 180

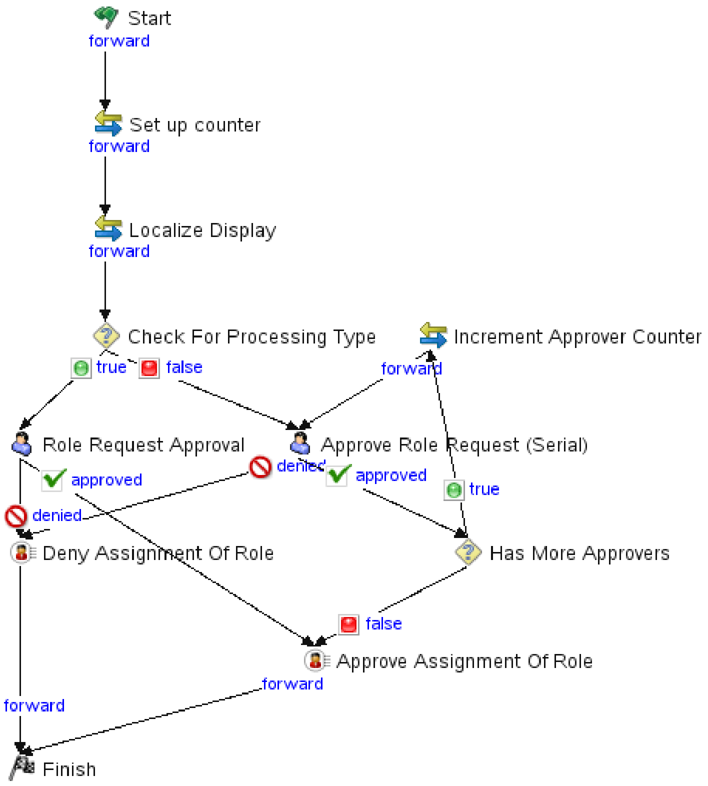
# About Resource Approval Workflows

Resource approval workflows are specialized workflows that provide support for resource approval and revocation on the User Application's **Roles** tab. The Roles Based Provisioning Module includes a read-only Resource Approval workflow (named Resource Approval) whose design pattern supports:

- The ability to process resource approvals in either serial or quorum mode.
- The retrieval of approver DNs from the resource object (nrfResourceRequest). If you create a custom workflow, the approvers must be defined in the workflow; however, this might lead to addressee evaluation problems and less security concerning who can approve a resource.
- The ability to display the resource using localized display names.
- All nrfResourceRequest object mappings for request and approval forms.
- Logging and reporting functions.
- Read-only display of request information. The resource approval workflow does not allow changes to the request. Approvers have only the ability to approve or deny the resource request.
- An email notification is sent to all approvers of resource approval workflows. A completed notification email is sent upon completion of the resource approval workflow. The recipient email address is used when the workflow is intended to be assigned to a user identity.

This pattern is shown in Figure 6-11.

***Figure 6-11*** *Default Resource Approval Workflow*



The components of this workflow, and their responsibilities are summarized in Table 6-13.

***Table 6-13*** *Standard Resource Approval Activities*

| Activity Name | Activity Type | Description |
|---|---|---|
| Start | Start | Logical starting point for all workflows. For resource approvals it must instantiate the nrfResourceRequest object. |
| Set up counter | Mapping | Sets up the counter for the number of approvers in case the mode is Serial. |
| Localize Display | Mapping | Sets up the display labels for each of the associated display names for the user's locale. |
| Check for Processing Type | Condition | Determines whether the approval is a quorum condition by setting the Condition property to this ECMA expression:<br>`nrfResourceRequest.isQuorumProcess()`<br>If the quorum condition exists, control proceeds to the Approve Resource Request (Quorum). If the quorum condition does not exist, control proceeds to the Approve Resource Request (Serial). |
| Approve Resource Request (Quorum) | Approval | This is where the decision to approve or deny the request is recorded as part of the workflow instance. The quorum condition required to make the process successful is retrieved from the nrfQuorum attribute of the nrfResourceRequest object. |
| Approve Resource Request (Serial) | Approval | This is where the decision to approve or deny the request is recorded as part of the data flow associated with the workflow instance.<br>The workflow loops through the list of approvers found in the nrfResourceRequest object. The request is approved if all approvers in the serial process approve the request. The request is denied upon the first rejection from an approver in the serial process |
| Deny Assignment of Resource | Resource Request Binding | Changes the deny attribute in the nrfResourceRequest object to true. |
| Approve Assignment of Role | Resource Request Binding | Changes the approve attribute in the nrfResourceRequest object to true. |
| Finish | Finish | Logical end point of all workflows. |

To use the standard Resource Approval workflow in your user application, you must specify your own users as Trustees. For information on setting the Trustees property, see "Modifying Settings of a Provisioning Request Definition" on page 86.

## Writing Custom Resource Workflows

If the standard resource approval workflow does not support your business needs, and cannot be customized to do so, you can write your own. At a minimum, a custom resource approval workflow must:

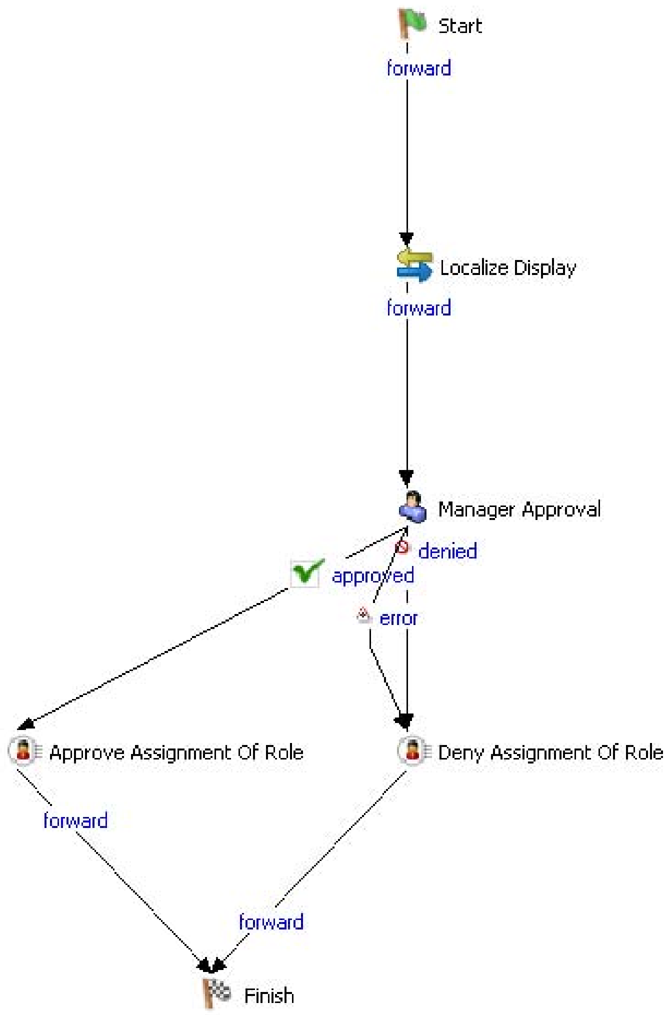
- Contain two Resource Request Binding activities

  One Resource Request Binding activity must be set to approved and the other set to denied. You must link each of the Resource Request Binding activities to the Finish activity. If the workflow does not meet this requirement, it is invalid, and Designer prevents you from deploying it. The Role and Resource Service driver needs these values to set the status for the workflow and to then apply the logic to associate the resource to the identity.
- Contain the following control in the request form:
  - Form Field Name: nrfResourceRequestDN
  - Data Type: Resource Request
  - Control Type: Text
- Instantiate the nrfResourceRequestDN in the Pre Activity Data Item Mapping.
- Contain the following in the Post Activity Data Item Mapping:
  - Source Form Field: nrfResourceRequestDN
  - Target Expression: flowdata.nrfResourceRequestDN
  - Data Type: dn
- Not contain the following ECMA expressions in the Data Item Mapping or Properties definitions because they might return null:
  - `getApprovalDN()`
  - `getAllApproversDN()`
  - `getAllSodApproversDN()`

  Because Designer and the User Application user interface do not allow entry of approvers for custom resource approval workflows, you must specify the approvers in the workflow itself. Therefore, if you create a custom workflow based on a copy of the Resource Approval provisioning request definition, you must remove the ECMA methods from Data Item Mapping or Properties definitions.

In the following example, a user requests a resource and the user's manager approves it.

*Figure 6-12 Sample Custom Resource Approval Workflow*



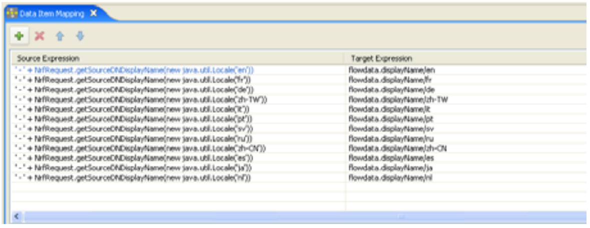
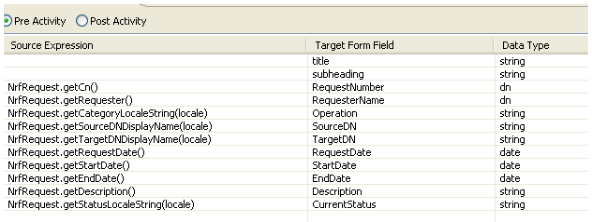
The components of this workflow, and their responsibilities are summarized in Table 6-14.

*Table 6-14 Sample Custom Workflow Components*

| Activity Name | Activity Type | Description |
|---|---|---|
| Start | Start | Logical starting point of all workflows. |
| Localize Display | Mapping | Sets up the display labels for the user's locale. |
| Manager Approval | Approval | This is where the decision to approve or deny the request is recorded as part of the workflow instance. The resource request approval is needed only by the requestor's manager. |
| Approve Assignment of Role | Role Binding | Changes the approve attribute in the nrfResourceRequest object to true. |
| Deny Assignment of Role | Role binding | Changes the deny attribute in the nrfResourceRequest object to true. |
| Finish | Finish | Logical end point of all workflows. |

The data item mapping for the sample custom resource approval workflow is defined in Table 6-15

*Table 6-15 Sample Custom Resource Approval Workflow Data Item Mapping*

| Activity Name | Property Type | Property Value |
|---|---|---|
| Start | Data Item Pre Activity | Source Expression: None<br>Target Form Field: nrfResourceRequestDN<br>Data Type: dn |
| | Data Item Post Activity | Source Form Field: nrfResourceRequestDN<br>Target Expression: flowdata.nrfResourceRequest/DN<br>Data Type: DN |
| Localize Display | Data Item Source and Target mapping |  |
| Manager Approval | Addressee Property | `Addressee IDVault.get(recipient,'user','manager')` |
| | Data Item Pre Activity |  |

| Source Expression | Target Expression |
|---|---|
| '-' + NrfRequest.getSourceDNDisplayName(new java.util.Locale('en')) | flowdata.displayName/en |
| '-' + NrfRequest.getSourceDNDisplayName(new java.util.Locale('fr')) | flowdata.displayName/fr |
| '-' + NrfRequest.getSourceDNDisplayName(new java.util.Locale('de')) | flowdata.displayName/de |
| '-' + NrfRequest.getSourceDNDisplayName(new java.util.Locale('zh-TW')) | flowdata.displayName/zh-TW |
| '-' + NrfRequest.getSourceDNDisplayName(new java.util.Locale('it')) | flowdata.displayName/it |
| '-' + NrfRequest.getSourceDNDisplayName(new java.util.Locale('pt')) | flowdata.displayName/pt |
| '-' + NrfRequest.getSourceDNDisplayName(new java.util.Locale('sv')) | flowdata.displayName/sv |
| '-' + NrfRequest.getSourceDNDisplayName(new java.util.Locale('ru')) | flowdata.displayName/ru |
| '-' + NrfRequest.getSourceDNDisplayName(new java.util.Locale('zh-CN')) | flowdata.displayName/zh-CN |
| '-' + NrfRequest.getSourceDNDisplayName(new java.util.Locale('es')) | flowdata.displayName/es |
| '-' + NrfRequest.getSourceDNDisplayName(new java.util.Locale('ja')) | flowdata.displayName/ja |
| '-' + NrfRequest.getSourceDNDisplayName(new java.util.Locale('nl')) | flowdata.displayName/nl |

● Pre Activity ○ Post Activity

| Source Expression | Target Form Field | Data Type |
|---|---|---|
| | title | string |
| | subheading | string |
| NrfRequest.getCn() | RequestNumber | dn |
| NrfRequest.getRequester() | RequesterName | dn |
| NrfRequest.getCategoryLocaleString(locale) | Operation | string |
| NrfRequest.getSourceDNDisplayName(locale) | SourceDN | string |
| NrfRequest.getTargetDNDisplayName(locale) | TargetDN | string |
| NrfRequest.getRequestDate() | RequestDate | date |
| NrfRequest.getStartDate() | StartDate | date |
| NrfRequest.getEndDate() | EndDate | date |
| NrfRequest.getDescription() | Description | string |
| NrfRequest.getStatusLocaleString(locale) | CurrentStatus | string |

| Activity Name | Property Type | Property Value |
|---|---|---|
|  | Data Item Post Activity | None |
| Approve Assignment of Resource | Action Property | approved |
| Deny Assignment of Resource | Action Property | denied |
| Finish |  | None |

# Debugging a Workflow

When testing a workflow, you might need to see the values of the variables you're using in the flow. Some options include:

- "Using the Log Activity" on page 183
- "Using the Workflow Database" on page 183
- "Changing Log Levels" on page 183

## Using the Log Activity

Use the Log activity to display messages containing the variables you need to look at. After 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 "Log Activity" on page 204.

## Using the Workflow Database

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.

## Changing Log Levels

During the debugging process, you can change the log levels associated with the workflow system (`com.novell.soa.af.impl`), the provisioning requests component of the User Application (`com.novell.srvprv.apwa`), and the evaluation of server side scripts (`com.novell.soa.script`). This approach might 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.

# Provisioning Multiple Individuals with One Workflow Instance

You can configure a provisioning request definition so that one individual (for example, a manager) can provision multiple individuals (for example, members of a group) with one workflow. The provisioning request definition can be configured to provision any one of the following:

- Multiple individual users from the default user container
- All members of a group from the default group container (for example, Sales, Marketing, HR, IT)
- All members of any arbitrary Identity Vault container

To create this type of workflow, create the provisioning request definition as you normally would. On the Overview panel, select **Single Flow Provision Members** from the **Flow Strategy** list.

- "Basic Steps for Using the Workflow" on page 184

## Basic Steps for Using the Workflow

This section describes the basic steps for using a workflow that utilizes the **Single Flow Provision Members** flow strategy. For more detailed information about making process requests, see "Making a Team Process Request" in the *NetIQ Identity Manager - User's Guide to the Identity Applications*.

1 Log in to the user application as a user application administrator.

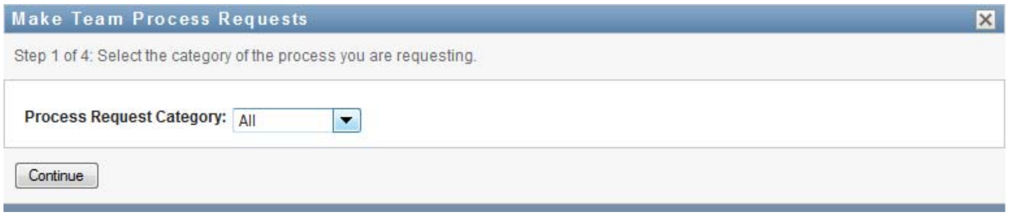
2 Click **Work Dashboard.**

3 In the left pane, click **Settings** > **Team Settings** > **Make Team Process Requests**.

4 Click **Select a team** to select the team for which you have been designated a Team Manager.

5 Click **Continue**.

6 Select the process request category to which the request belongs, then click **Continue**.



Make Team Process Requests

Step 1 of 4: Select the category of the process you are requesting.

Process Request Category: All

7 Click the name of the workflow you want to use.

8 Click the resource name you want to request.

9 Click the name of the recipient you want to receive the request, then click **Continue**.

# Making Distinguished Name References Portable

When you use a DN in an expression in a provisioning request definition, the expression might fail if you deploy the provisioning request definition to an Identity Vault with a different structure. You typically specify DNs in:

- Overview panel: Trustee specification.
- User activity: Addressee and escalation addressee.

- Entity activity: Entitlement reference and entity DN.
- Many other expressions, for example, IDVault.get(dn, class, attribute).

Some expressions, such as recipient, are portable. The following expressions, which are used by default in the User activity, are also portable:

```
IDVault.get(recipient,'user','manager')
```

```
IDVault.get(approval_A.getAddressee(),'user','manager')
```

To ensure that your DN expressions are portable across Identity Vaults, you can use one of the following variables:

- `ROOT_CONTAINER`: For example, `ou=idm-prov,o=novell`
- `PROVISIONING_DRIVER`: For example, `cn=UserApplication,cn=TestDrivers,o=novell`
- `USER_CONTAINER`: For example, `ou=users,ou=idm-prov,o=novell`
- `GROUP_CONTAINER`: For example, `ou=groups,ou=idm-prov,o=novell`

These variables are defined during installation of the user application and are resolved at runtime by the ECMAScript engine. You can find them in the ECMA Expression Builder under the process node. Suppose you wanted to reference an entitlement at the following DN:

```
'cn=myEntitlement,cn=UserApplication,cn=TestDrivers,o=novell'
```

You could use the following expression to make the DN portable to any identity vault:

```
''cn=MyEntitlement,' + PROVISIONING_DRIVER
```

You can use this technique for users and groups also.

**NOTE:** Trustees are not expressions so you cannot use this technique with Trustees.

# 7 Workflow Activity Reference

This section provides details on configuring the different types of workflow activities.

The display names for all activities can be localized by clicking the **Localize Strings** button (see "Localizing Provisioning Objects" on page 39) for the activity name property. Activity display names are also exported as part of the Provisioning view's **Export > Export Localization to File** (see "Exporting and Importing Data to Localize" on page 41) so that you can send the activity names to be localized as part of the rest of the User Application display labels and strings.

## 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 can 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 5, "Creating Forms for a Provisioning Request Definition," on page 95.

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 properties:

***Table 7-1** Start Activity Properties*

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

| Property Name | Description |
|---|---|
| **Resource Name Override** | Allows you to override the provisioning request definition’s display name. This is the name that displays in the **Resource** column when the user selects **My Requests** or **Requests**. The name can be a constant or the result of an ECMA expression, and it can be localized for each supported locale. To access the dialog boxes that let you specify the constant or ECMA expression, the **Localize Strings** button in the **Value** column.<br>To specify a constant, click k and type the value in the field. The value is displayed exactly as entered in this field.<br>To specify an ECMA expression, click E and specify the expression.<br>**TIP:** If the value is an ECMA expression, any constants within the expression must be in single quotes.<br>If you specify both a constant and ECMA expression, the runtime displays whichever value was the last one entered as shown in the dialog box (above). |

# 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 7-2*** *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>The Pre-activity Mapping expression is evaluated twice before the form is presented, once during the initial presentation to the form and then again prior to the post to ensure that all the values on the form have a valid type, even those that were not initialized. Because of this behavior, any calls made to external systems are made twice. For example, a call that retrieves a unique counter for a value makes two calls that allocates two counters with the last one requested being used.<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 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>The DNDisplay control is not available for post activity mappings.<br>**NOTE:** When the **Post-Activity** option is selected, the cells in the **Source Form Field** column are not editable. |

| Setting | Description |
|---|---|
| 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 271.

## Email Notification

Not supported with this activity.

# 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 5, “Creating Forms for a Provisioning Request Definition,” on page 95.

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” on page 189
- “Data Item Mapping” on page 194
- “Available ECMAScript Methods” on page 195
- “Email Notification” on page 196
- “Addressing an Approval Activity” on page 198

## Properties

The Approval activity has the following properties:

***Table 7-3*** *Approval Activity Properties*

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

| Property Name | Description |
|---|---|
| **Addressee** | Specifies a dynamic expression that identifies the addressee for the activity. The addressee is the approver of the workflow.<br>The addressee is determined at runtime, based on evaluation of the expression. Designer validates that the expression is a valid ECMA expression. It cannot validate whether the expression resolves to a valid object (such as a role) or whether that object will exist at runtime.<br>For information on specifying addressees (such as specifying a role as the approver), see "Specifying the Addressee Property" on page 198.<br>For more information about developing valid Addressee expressions, and about how Addressee interacts with the Approver Type property, see "Addressing an Approval Activity" on page 198.<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. After the testing phase is complete, you can change the addressee to the correct value.<br>For details on building ECMA expressions, see Chapter 9, "Working with ECMA Expressions," on page 271. For descriptions of the system variables available in a workflow, see "Understanding Workflow Data" on page 163. |
| **Reminder Start** | Specifies a dynamic expression that defines, in milliseconds, the time at which the first reminder email should be sent. The start value is an offset from the time of the first assignment associated with the activity. You can pick predefined expressions that represent common intervals (for example, hour, day, week) in the **ECMAScript Objects** pane of the ECMA Expression Builder.<br>This is part of the reminder email function. If this activity is considered important and needs to be acted on quickly, you can configure the activity to send a reminder email to the activity addressee. For example, you can set the reminder settings to send a reminder email 5 days before the activity times out, and on a daily basis until the activity times out. To do this, specify a **Reminder Start** time, a **Reminder Interval**, and the email to be sent (see "Email Notification" on page 196).<br>For details on building ECMA expressions, see Chapter 9, "Working with ECMA Expressions," on page 271. For descriptions of the system variables available in a workflow, see "Understanding Workflow Data" on page 163. |
| **Reminder Interval** | Specifies a dynamic expression that defines the interval between which reminder emails are sent. You can pick predefined expressions that represent common intervals (for example, hour, day, week) in the **ECMAScript Objects** pane of the ECMA Expression Builder. |

| Property Name | Description |
|---|---|
| **Escalation Addressee** | Not available when the approver type is **Multiple** or **Quorum**<br>Specifies a dynamic expression that identifies the user who should get this task if the timeout limit has been reached.<br>The escalation addressee is determined at runtime, based on how the expression is evaluated.<br>For details on building ECMA expressions, see Chapter 9, “Working with ECMA Expressions,” on page 271. For descriptions of the system variables available in a workflow, see “Understanding Workflow Data” on page 163. |
| **Escalation Count** | Not available when the approver type is **Multiple** or **Quorum**.<br>Specifies the number of times to retry the activity in the event of a timeout.<br>When an activity times out, the workflow process can try to complete the activity again, depending on the escalation 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.<br>The **Timeout** interval (see **Timeout** in this table) takes precedence over the **Escalation Interval.** For example, if you set the timeout to 10 minutes, and specify an Escalation Count of 3 and Escalation Interval of 5 minutes, the activity finishes after 10 minutes without attempting all of the retries. In this example, the second retry would be canceled, and the workflow would finish processing for the activity. At the conclusion of the activity, the workflow engine would follow the link defined by the final timeout action. |
| **Escalation Interval** | Not available when the approver type is **Multiple** or **Quorum**.<br>Specifies a dynamic expression that defines the period of time allotted for the addressee to complete the task. The escalation interval applies each time the activity is executed by the addressee.<br>The **Timeout** interval (see **Timeout** in this table) takes precedence over the **Escalation Interval.** For example, if you set the timeout to 10 minutes, and specify an **Escalation Count** of 3 and **Escalation Interval** of 5 minutes, the activity will finish after 10 minutes without attempting all of the retries. In this example, the second retry would be canceled, and the workflow would finish processing for the activity. At the conclusion of the activity, the workflow engine would follow the link defined by the final timeout action.<br>For details on building ECMA expressions, see Chapter 9, “Working with ECMA Expressions,” on page 271. For descriptions of the system variables available in a workflow, see “Understanding Workflow Data” on page 163. |

| Property Name | Description |
|---|---|
| **Escalation Reminder Start** | Not available when the approver type is **Multiple** or **Quorum**.<br>Specifies a dynamic expression that defines the time at which the first reminder email (see **Reminder Start** in this table) should be sent to the **Escalation Addressee**. The start value is an offset from the time of the escalation assignment. You can pick predefined expressions that represent common intervals (for example, hour, day, week) in the **ECMAScript Objects** pane of the ECMA Expression Builder. |
| **Escalation Reminder Interval** | Not available when the approver type is **Multiple** or **Quorum**.<br>Specifies a dynamic expression that defines how often messages are sent to the **Escalation Addressee** after the first escalation reminder is sent. You can pick predefined expressions that represent common intervals (for example, hour, day, week) in the **ECMAScript Objects** pane of the ECMA Expression Builder. |
| **Final Timeout Action** | Determines the final state of the request in the event that the workflow times out. The choices are<br>◆ approved<br>◆ denied<br>◆ refused<br>◆ timedout<br>◆ error |
| **Timeout** | 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>The Timeout setting takes precedence over the **Escalation Count** and **Escalation Interval** values. If the **Timeout** setting for the activity is reached before one or more of the escalation attempts have been tried, the activity finishes processing without executing these escalation attempts. For example, if you set the timeout to 10 minutes, and specify an **Escalation Count** of 3 and **Escalation Interval** of 5 minutes, the activity finishes after 10 minutes without attempting all of the escalation attempts. In this example, the second escalation attempt would be canceled, and the workflow would finish processing for the activity. At the conclusion of the activity, the workflow engine would follow the link defined by the final timeout action.<br>For details on building ECMA expressions, see Chapter 9, “Working with ECMA Expressions,” on page 271. For descriptions of the system variables available in a workflow, see “Understanding Workflow Data” on page 163. |
| **Timeout Units** | Determines the unit of measure used for the timeout interval. The choices are<br>◆ Milliseconds<br>◆ Days<br>◆ Hours<br>◆ Minutes<br>◆ Seconds |

| Property Name | Description |
|---|---|
| **Form** | Specifies the name of the approval form to display at runtime, or lets you define a new form. Select the name of the form you want to use or **create new form**. When you choose to create a new form, the Create New Form Wizard launches<br>Select the data items to include in the form from the data items listed, then click **Finish**. The Approval Form Wizard generates each of the selected data items as a String type field in the new form.<br>An Approval activity must have a form associated with it. If no form is specified, an error message is displayed at runtime. |
| **Exclude Requestor** | Specifies whether requestors can approve their own provisioning requests.<br>◆ **True:** The requestor is not allowed to approve their own provisioning requests.<br>◆ **False**: The requestor is allowed to approve their own provisioning requests. |
| **Approver Type** | Specifies the number of addresses that are allowed and the approval pattern that is enforced for this activity. The choices are<br>◆ **Normal:** Action by the addressee is required to complete the approval.<br>◆ **Group:** Action by one addressee in the group is required to complete the approval.<br>◆ **Multiple:**  Action by all of the addressees is required to complete the approval.<br>You cannot use post activity data item mapping with the **Multiple** Approver Type.<br>◆ **Quorum:** Action by a percentage of addressees or an absolute number of addressees (see **Quorum** property in this table) is required to complete the approval.<br>You cannot use post-activity data item mapping with the **Quorum** Approver Type.<br>For information about how the **Approver Type** property interacts with the **Addressee** property, see "Addressing an Approval Activity" on page 198. |
| **Notify by E-Mail** | Specifies whether this activity should send email notifications. Set to True to notify by email; otherwise, set to False.<br>You specify the email to send using the E-Mail Notification tab (see "Email Notification" on page 196).<br>To use this feature, the **Notify participants by E-Mail** parameter for the provisioning request definition must be set to True (see Table 4-3, "Overview Properties," on page 86). |

| Property Name | Description |
|---|---|
| Quorum | Not available when the approver type is Normal, Group, or Multiple.<br>Allows you to specify a constant value or to create an ECMA expression that specifies a percentage (for example, '75%') of approvals that is required before a quorum is achieved, or an absolute number (for example, '3') of approvals that are required before a quorum is achieved. |
| Priority | Specifies a dynamic expression that defines the priority of the approval activity. Valid priority values are 1, 2, or 3. You can also define an expression to determine the priority from workflow data. For example, `flowdata.get("Priority")`.<br>In the User Application, users can sort the list of tasks by the priority values of the tasks. |

**NOTE:** To enable delegation to a group DN, you can have an approver type of Group or Normal, but the Addressee value must be an expression that returns the user DNs for each member of that group For example, IDVault.get(groupdn, ‘sales’, ‘members’)

# 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 data retrieved via expression calls to the directory abstraction layer. The post-activity mappings move form data back into the flowdata object.

***Table 7-4*** *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 a cell in the **Source Expression** column to specify where the approval form gets data for a particular target form field.<br>The Pre-activity Mapping expression is evaluated twice before the form is presented, once during the initial presentation to the form and then again prior to the post to ensure that all the values on the form have a valid type, even those that were not initialized. Because of this behavior, any calls made to external systems are made twice. For example, a call that retrieves a unique counter for a value makes two calls that allocates two counters with the last one requested being used.<br>**NOTE:** When the **Pre-Activity** choice is selected, the cells in the **Target Form Field** column are not editable. |

| Setting | Description |
|---|---|
| **Post Activity** | Allows you to specify one or more **Post Activity** mappings. When this option is selected, you can double-click 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>You cannot use **Post Activity** mapping with the **Multiple** and **Quorum** approver types (see "Properties" on page 189).<br>The DNDisplay control is not available for post activity mappings.<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 "DNMaker" on page 115.<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 271.

## Available ECMAScript Methods

The Approval activity provides several default methods to use in ECMAScript expressions. Some methods are displayed in the ECMAScript Objects pane of the ECMA Expression Builder, while others must be entered manually into the text area provided in the ECMA Expression Builder.

***Table 7-5** Available ECMAScript Methods for Approval Activities*

| Object | Method | Description |
|---|---|---|
| action | `*Activity-name*.getAction()` | Returns the approval action taken by the activity. Possible options are:<br>◆ APPROVED<br>◆ DENIED<br>◆ REFUSED<br>◆ TIMEDOUT<br>◆ ERROR |
| addressee | `*Activity-name*.getAddressee()` | Returns the DN of the user who needs to approve or deny the requested action. |
| name | `*Activity-name*.getName(*locale*)` | Returns the name of the approval activity for the specified locale. |

| Object | Method | Description |
|---|---|---|
| timestamp | `Activity-name.getTimestamp()` | Returns the date and time of any Approval actions taken by the user or system, including APPROVED, DENIED, REFUSED, TIMEDOUT, or ERROR. |
| user | `Activity-name.getUser()` | Returns the DN of the owner of the work task for the activity. |
| workId | `Activity-name.getWorkId()` | Returns the work item ID for the activity. |
| N/A | `Activity-name.getAddresseeList()` | Returns a list of DNs of all users who need to approve or deny the requested action.<br>**NOTE:** This method is not displayed in the ECMAScript Objects pane and must be entered manually. |
| N/A | `Activity-name.getNotifyAddressee()` | Returns the full name of the user who needs to approve or deny the requested action.<br>**NOTE:** This method is not displayed in the ECMAScript Objects pane and must be entered manually. |

## Email Notification

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

***Table 7-6*** *E-mail Notification Settings for the Approval Activity*

| Setting | Description |
|---|---|
| **Notify** | Specifies that this email notification is a notification email. |
| **Reminder** | Specifies that this email notification is a reminder email. |
| **Retry Reminder** | Specifies that this email notification is a retry reminder email. |
| **Show System Tokens** | Displays system tokens (for example, TO, CC, BCC, REPLYTO, TO_DN, CC_DN, and BCC_DN) in the **Target** column. |
| **E-Mail Template** | Specifies the name of the email template to use. By default, the Approval activity uses the Provisioning Notification template.<br>You can edit an email template in Designer. For more information, see "Editing an email template:" on page 158. |

| Setting | Description |
|---|---|
| **Source/Target** | Specifies the source expressions for target tokens in the email body.<br>The list of target tokens is determined by the selected email template. You cannot add new tokens, but you can assign values to the tokens by building your own source expressions. At runtime, source expressions are evaluated to determine the value of each token.<br>The available target tokens are listed below:<br>◆ TO<br>◆ CC<br>◆ BCC<br>◆ REPLYTO<br>◆ TO_DN<br>◆ CC_DN<br>◆ BCC_DN<br>◆ recipientFullName<br>◆ initiatorFullName<br>◆ requestTitle<br>◆ userFirstName<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 abstraction layer (IDVault object). You can modify these expressions to suit your application requirements.<br>For details on building ECMA expressions, see Chapter 9, “Working with ECMA Expressions,” on page 271. |

**NOTE**

- Email notification is supported only when the **Notify participants by E-Mail** check box is selected on the **Overview** tab, and the **Notify by E-Mail** property for the Approval activity is set to True.
- When you create a workflow for use with the Resource Request portlet, and you use the “_default_” as the expression for the TO token, the addressee expression must be an IDVault expression.
- If you create an activity using any of the target tokens TO_DN, CC_DN, or BCC_DN, you must specify a user’s DN or an expression that resolves to a user’s DN as the source expression for the token.
- If you create an activity using both the target tokens TO and TO_DN, the workflow sends out duplicate notification emails to the target users.

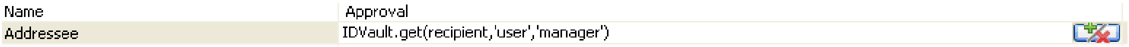
# Addressing an Approval Activity

To address an Approval activity, you must enter a valid expression for the **Addressee** property. The Addressee is the approver for the activity. The number of approvals that are required to approve the activity is determined by the relationship between the **Addressee** property and the **Approver Type** property as described in "Relationship Between Addressee and Approver Type" on page 199.

## Specifying the Addressee Property

To build the addressee expression:

**1** Click the  button in the Addressee property Value column.



| Name | Approval |
|---|---|
| Addressee | IDVault.get(recipient,'user','manager') |

Designer launches the dialog box where you can add or remove an expression. The following dialog only displays when the Approver Type is Group, Multiple, or Quorum.

**2** Click + to add a new addressee expression by using the Expression Builder.

You can choose one of the ECMAScript Objects to build the addressee expression, or use the **Identity Vault** or **Search Roles** buttons to select a specific object. The **Search Roles** button is not available when Approver Type is Normal.

**2a** To specify a Role as the Addressee, click **Search Roles**.

**2b** In the dialog box, specify the **CN**, **Display Name**, **Description**, **Role Category**, and **Role Level** on which you want to search.

For **CN**, **Display Name**, and **Description**, you can enter a wildcard (such as S*, *S) or regular expressions (such as [A-Zoo-z]*).

You can enter a value for all of the fields or none of the fields. If you do not supply a value in a particular field, the search returns all of the possible values for that field. If you enter values in one or more of the fields, the values are ANDed together to create the search filter. The search occurs on the roles defined locally. Roles matching the search criteria are displayed in the **Matching Roles** selection list.

**2c** Select a role from the **Roles** selection list, then click **OK**. The role is added to the expression area.

**3** Click **OK** after you are satisfied with expression.

## Valid Addressee Expressions

An Addressee expression must resolve to one of the following at runtime:

- A valid individual addressee that can be a user DN, a group DN, or a role DN.
- A valid list of addressees (for example, created using a Java vector object) that can contain multiple User DNs, multiple group DNs, or multiple role DNs, or a mixture of both.

Because the addressee is the approver, the maximum number of approvals possible equals the number of Addressees (the number of User DNs plus the number of Group DNs or Role DNs) and does not include or count the individual members of a Group or Roles.

**NOTE:** A Group DN or a Role DN is always processed to contribute a single vote (that is, when one member of a group or role claims an activity, the rest of the members of the group or role can no longer see or claim the activity), regardless of the **Approver Type**.

The following table provides examples of valid addressee expressions that you can create using the ECMA Expression Builder.

***Table 7-7** Examples of Addressee Expressions*

| Type of Expression | Example |
|---|---|
| Individual user DN | `'cn=jdoe,ou=users,ou=mysample,o=myorg'` |
| Individual group DN | `'cn=Accounting,ou=groups,ou=mysample,o=myorg'` |
| Individual role DN | `'CN=Administer Drugs,CN=Level10,CN=RoleDefs,CN=RoleConfig,CN=AppConfig,' + PROVISIONING_DRIVER'` |
| A vector of DNs (can include user, group, or role DNs | `function DNVector() { v=new java.util.Vector(); v.add('CN=jdoe,' + USER_CONTAINER); v.add('CN=Accounting,' + GROUP_CONTAINER); v.add('CN=jsmith,' + USER_CONTAINER); v.add('CN=bsmith,' + USER_CONTAINER);`<br>`v.add('CN=Administer Drugs,CN=Level10,CN=RoleDefs,CN=RoleConfig,CN=AppConfig,' + PROVISIONING_DRIVER);`<br>`return v; };`<br>`DNVector();` |

## Relationship Between Addressee and Approver Type

Because the addressee is the approver, the behavior of the workflow and the total number of affirmative approvals needed varies depending on the type of Addressee that is specified by the Addressee expression, and the Approver Type that is selected.

- "Normal Approver Type" on page 199
- "Group Approver Type" on page 200
- "Multiple Approver Type" on page 201
- "Quorum Approver Type" on page 202

### Normal Approver Type

The following table describes the workflow behavior when different types of addressee are used with the Normal **Approver Type**.

***Table 7-8** Workflow Behavior with the Normal Approver Type*

| Addressee Value | Description |
|---|---|
| Individual User DN | ◆ Only the user can see the **Approval** activity in his or her task list.<br>◆ Only one approval is needed to complete the activity as Approved. |
| Individual Group DN | ◆ Each member of Group can see the activity in the task list.<br>◆ When one member claims the activity, it is removed from the task lists of others.<br>◆ Only one approval is needed to complete the activity as Approved. |

| Addressee Value | Description |
|---|---|
| Individual Role DN | ◆ Each member of the role can see the activity in the task list.<br>◆ When one role member claims the activity, it is removed from the task lists of others.<br>◆ Only one approval is needed to complete the activity as Approved. |
| Multiple User DNs | Not allowed. |
| Multiple Group DNs | Not allowed. |
| Multiple Role DNs | Not allowed. |
| Mixture of Users, Groups, and Roles | Not allowed. |

## Group DNs and Proxy Processing

If a workflow is assigned to a Group and email notification is used for the approvals, all members of the group are sent an email. If a proxy user is assigned to any members of the group, the processing works as follows:

- If the approver is a single user then the email notification is sent to both users (the original and proxy users).
- If the approver is a group DN and one of the users in the group is assigned a proxy user, the user who is the proxy is not notified by email when a new request is placed in the task list.

  If you want the proxy user to be notified by email, assign the approval task to the members of the group and set the approver type to **Group Approver**. For example, if you assign the approval activity to:

  ```
  IDVault.get('cn=Marketing,ou=groups,ou=idmsample,o=novell' , 'group',
  'Member')
  ```

  When you set the approval type to Group, a notification is sent to each member's proxy, if the member has a proxy. One member of the group can claim and act on the approval task which is the same behavior as if you assigned it directly to the group DN.

## Group Approver Type

The following table describes the workflow behavior when different types of addressee are used with the Group **Approver Type**.

***Table 7-9*** *Workflow Behavior with the Group Approver Type*

| Addressee Value | Description |
|---|---|
| Individual User DN | ◆ Only the user can see the Approval activity in his or her task list.<br>◆ Only one approval is needed to complete the activity as Approved. |
| Individual Group DN | ◆ Each member of Group can see the activity in his or her task list.<br>◆ When one member claims the activity, it is removed from task lists of others.<br>◆ Only one approval is needed to complete the activity as Approved. |

| Addressee Value | Description |
|---|---|
| Individual Role DN | ◆ Each member of the Role can see the activity in his or her task list.<br>◆ When role one member claims the activity, it is removed from task lists of others.<br>◆ Only one approval is needed to complete the activity as Approved. |
| Multiple User DNs | ◆ Each user in the virtual group can see the activity in his or her task list.<br>◆ When one user from the virtual group claims the activity, the activity is removed from the task lists of others.<br>◆ Only one approval is needed to complete the activity as Approved. |
| Multiple Group DNs | ◆ Each member in each of the groups can see the activity in his or her task list.<br>◆ When one user from the virtual group claims the activity, the activity is removed from the task lists of others in all of the groups.<br>◆ Only one approval is needed to complete the activity as Approved. |
| Multiple Role DNs | ◆ Each member in each of the roles can see the activity in his or her task list.<br>◆ When one user from the one of the roles claims the activity, the activity is removed from the task lists of others in all of the other roles.<br>◆ Only one approval is needed to complete the activity as Approved. |
| Mixture of Users, Groups, and Roles | ◆ Each user and member of each Group or Role can see the activity in his or her task list.<br>◆ When one of the approvers claims the activity, the activity is removed from the task lists of others.<br>◆ Only one approval is needed to complete the activity as Approved. |

## Multiple Approver Type

The following table describes the workflow behavior when different types of addressee are used with the Multiple **Approver Type**.

***Table 7-10*** *Workflow Behavior with the Multiple Approver Type*

| Addressee Value | Description |
|---|---|
| Individual User DN | ◆ Only the user can see the activity in his or her task list.<br>◆ Only one approval is needed to complete the activity as Approved. |
| Individual Group DN | ◆ Each member of the group can see the activity in his or her task list.<br>◆ When one member claims the activity, the activity is removed from the task lists of others.<br>◆ Only one approval is needed to complete the activity as Approved. |
| Individual Role DN | ◆ Each member of the role can see the activity in his or her task list.<br>◆ When one role member claims the activity, the activity is removed from the task lists of others in the role.<br>◆ Only one approval is needed to complete the activity as Approved. |

| Addressee Value | Description |
|---|---|
| Multiple User DNs | ◆ Each user can see the activity in his or her task list.<br>◆ Each user can claim the activity.<br>◆ Approval of each user is needed to complete the activity as Approved.<br>◆ Any single denial completes the activity as Denied. |
| Multiple Group DNs | ◆ Each member in each of the groups can see the activity in his or her task list.<br>◆ When one member from a group claims the activity, the activity is removed from the task list of others in that Group.<br>◆ Each group must supply one approval to complete the activity as Approved.<br>◆ Any single denial completes the activity as Denied. |
| Multiple Role DNs | ◆ Each member in each of the roles can see the activity in his or her task list.<br>◆ When one member from a role claims the activity, the activity is removed from the task list of others in that role.<br>◆ Each role must supply one approval to complete the activity as Approved.<br>◆ Any single denial completes the activity as Denied. |
| Mixture of Users, Groups, and Roles | ◆ Each user and each member of each group or role can see the activity in his or her task list.<br>◆ Each user can claim the activity, and one member of each group or role can claim the activity (then others in the group or role do not see the task.)<br>◆ Each user and one member of each group or role must approve to complete the activity as Approved.<br>◆ Any single denial completes the activity as Denied. |

## Quorum Approver Type

The following table describes the workflow behavior when different types of addressee are used with the Quorum **Approver Type**.

*Table 7-11 Workflow Behavior with the Quorum Approver Type*

| Addressee Value | Description |
|---|---|
| Individual User DN | ◆ Only the user can see the activity in his or her task list.<br>◆ Only one approval is needed to complete the activity as Approved. |
| Individual Group DN | ◆ Each member of the group can see the activity in his or her task list.<br>◆ When one member claims the activity, the activity is removed from the task lists of others.<br>◆ Only one approval is needed to complete the activity as Approved. |

| Addressee Value | Description |
|---|---|
| Individual Role DN | ◆ Each member of the role can see the activity in his or her task list.<br>◆ When one member claims the activity, the activity is removed from the task lists of others.<br>◆ Only one approval is needed to complete the activity as Approved. |
| Multiple User DNs | ◆ Each user can see the activity in his or her task list.<br>◆ All users can claim the activity simultaneously.<br>◆ An absolute number or specified percentage of Addressees must approve to complete the activity as Approved. |
| Multiple Group DNs | ◆ Each member in each group can see the activity in his or her task list.<br>◆ One member of each group can claim the task (then others in the group do not see the task).<br>◆ An absolute number or specified percentage of Addressees must approve to complete the activity as Approved. |
| Multiple Role DNs | ◆ Each member in each role can see the activity in his or her task list.<br>◆ One member of each role can claim the task (then others in the roles do not see the task).<br>◆ An absolute number or specified percentage of Addressees must approve to complete the activity as Approved. |
| Mixture of Users, Groups, and Roles | ◆ Each user and each member of each Group or Role can see the activity in his or her task list.<br>◆ Each user can claim the activity, and one member of each group or role can claim the activity (then others in the group do not see the task).<br>◆ An absolute number or specified percentage of Addressees must approve to complete the activity as Approved. |

## Troubleshooting Invalid Addressees

If the expression specified in the **Addressee** property of an Approval activity evaluates to a non-existent DN (for example, if the expression was hard-coded incorrectly, calculated incorrectly, or submitted incorrectly by a user selection), no indication is given that the workflow is not processing normally, when it is in fact orphaned. The application server console displays a normal forward message, and the Comment and Flow history shows a normal “assigned” message. To avoid this problem, we recommend that you follow these best practices:

1. Use a Condition activity before the Approval activity and validate the addressee in the Condition activity.
2. Since the addressee could still be deleted after the addressee is validated in the Condition activity, you should specify, for the Approval activity, a timeout interval and a link that performs the desired action in case the workflow times out.

# 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.

**NOTE:** Novell Audit can be configured to send its information to Novell Sentinel for additional logging and reporting features.

During the course of its processing, a workflow can log information about various events that have occurred. Users can then use the reporting tools to look at logged data.

Before you can use logging, you must enable logging in the user application.

**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.

## Properties

The Log activity has the following properties:

*Table 7-12 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. |
| 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>For details on building ECMA expressions, see Chapter 9, "Working with ECMA Expressions," on page 271. For descriptions of the system variables available in a workflow, see "Understanding Workflow Data" on page 163. |
| Comment | Specifies an ECMA expression that defines text that can be displayed in the user comments. You might use it to record the reason for a request or a request's completed approval status. Some examples include:<br>`"Reason for request: "+ flowdata.get('reason')`<br>or<br>`"Process has been " + flowdata.get(IDM_COMPLETED_APPROVAL_STATUS')` |

## Data Item Mapping

Not supported with this activity.

### Email Notification

Not supported with this activity.

## 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.

The Branch activity does not support synchronization between the branches while they are executing. Each branch must not depend on data being updated in another branch. The data synchronization is enforced by the Merge activity. After the Merge activity completes, all of the data set in the branches is available.

### Properties

The Branch activity has the following properties:

***Table 7-13*** *Branch 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.

## 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 properties:

*Table 7-14 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.

# 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” on page 206
- “Data Item Mapping” on page 207
- “Email Notification” on page 207

## Properties

The Condition activity has the following properties:

*Table 7-15 Condition Activity Properties*

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

| Property Name | Description |
|---|---|
| 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>**TIP:** If you need to test whether two objects are equal in a conditional expression, you should 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>`(approval_A.getAction() == "DENIED")`<br>instead of this one:<br>`(approval_A.getAction()).equals("DENIED")`<br>For details on building ECMA expressions, see Chapter 9, "Working with ECMA Expressions," on page 271. For descriptions of the system variables available in a workflow, see "Understanding Workflow Data" on page 163. |

## Data Item Mapping

Not supported with this activity.

## Email Notification

Not supported with this activity.

# Mapping Activity

The Mapping activity allows you to add or manipulate data in a workflow. It evaluates the source expression and saves the result in the target expression of the associated data items. You can use it as a way to combine data from parallel-processed approval forms after their data is moved to flowdata.

For example, in a parallel approval context you might need to collect data from more than one approval form that is dependent on each other or needs to be calculated with each other. To accomplish this, place a Mapping activity after a Merge activity and before any activities that consume the results (for example, Condition, Entity, Provisioning or another Approval activity).

You can also use the Mapping activity to isolate calls to external Java routines that might manipulate data and be resource intensive, thereby not slowing down user-based Approval activities in either their pre-activity or post-activity mapping phase.

- "Properties" on page 208
- "Data Item Mapping" on page 208
- "Email Notification" on page 208

### Properties

The Mapping activity has the following properties:

*Table 7-16 Mapping Activity Properties*

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

### Data Item Mapping

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

*Table 7-17 Mapping Activity Data Item Mappings*

| Setting | Description |
|---|---|
| Source Expression | Specifies a source expression. When you click a cell in the **Source Expression** column, the ECMA Expression Builder displays to help you define your expression. For example,<br>`function list() { s=new java.lang.String(); if (wi.XPath('count(flow-data/groups)' ) > 0) s="There was a group selected"; return s;}; list();` |
| Target Expression | Specifies a target expression. When you click a cell in the **Target Expression** column, the ECMA Expression Builder displays to help you define your expression or you can click the **Map All** button. An example of a target expression is:<br>`flowdata.testexpression` |

### Email Notification

Not supported with this activity

## Workflow Status

The Workflow Status activity lets you specify the approval status (approved or denied) for workflows that do not contain a provisioning activity (an Entitlement or Entity).

- "Properties" on page 209
- "Data Item Mapping" on page 209
- "Email Notification" on page 209

## Properties

The Workflow Status activity has the following properties:

***Table 7-18*** *Workflow Status Activity Properties*

| Property | Description |
|---|---|
| **Name** | Specifies the name of the activity. |
| **Workflow Status** | Specifies the approval status as an expression: either Approved or Denied. |

## Data Item Mapping

Not supported with this activity.

## Email Notification

Not supported with this activity.

# Email Activity

The Email activity provides a way to send an email to interested parties outside of an approval process.

## Properties

The Email activity has the following properties:

***Table 7-19*** *Email Activity Properties*

| Property Name | Description |
|---|---|
| **Name** | Provides a name for the activity. |
| **Notify by E-Mail** | Specifies whether this activity should send email notifications. Set to True to notify by email; otherwise, set to False.<br>You specify the email to send by using the **E-Mail Notification** tab (see "Email Notification" on page 210).<br>To use this feature, the **Notify participants by E-Mail** parameter for the provisioning request definition must be set to true (see Table 4-3, "Overview Properties," on page 86). |

## Data Item Mapping

Not supported with this activity.

# Email Notification

To enable email notification for this activity, you need to specify the email template to use, as well as source expressions for target tokens in the email body.

*Table 7-20 E-Mail Notification Settings for the E-Mail Activity*

<table>
<tr><th>Setting</th><th>Description</th></tr>
<tr><td>E-Mail Template</td><td>Specifies the name of the email template to use. By default, the Approval activity uses the Provisioning Notification template.<br>You can edit an email template in Designer. For more information, see "Editing an email template:" on page 158.</td></tr>
<tr><td>Source/Target</td><td>Specifies the source expressions for target tokens in the email body.<br>The list of target tokens is determined by the selected email template. You cannot add new tokens, but you can assign values to the tokens by building your own source expressions. At runtime, source expressions are evaluated to determine the value of each token.<br>The available target tokens are listed below:<br>◆ TO<br>To specify multiple recipients, use the following syntax:<br>'mail1@domain1.com||mail2@domain1.com'<br>◆ CC<br>◆ BCC<br>◆ REPLYTO<br>◆ TO_DN<br>◆ CC_DN<br>◆ BCC_DN<br>◆ recipientFullName<br>◆ initiatorFullName<br>◆ requestTitle<br>◆ userFirstName<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 abstraction layer (IDVault object). You can modify these expressions to suit your application requirements.<br>For details on building ECMA expressions, see Chapter 9, "Working with ECMA Expressions," on page 271.</td></tr>
</table>

**NOTE**

- Email notification is supported only when the **Notify participants by E-Mail** check box is selected on the **Overview** tab, and the **Notify by E-Mail** property for the activity is set to True.
- When you create a workflow for use with the Resource Request portlet, and you use the "_default_" as the expression for the TO token, the addressee expression must be an IDVault expression.
- If you create an activity using any of the target tokens TO_DN, CC_DN, or BCC_DN, you must specify a user's DN or an expression that resolves to a user's DN as the source expression for the token.
- If you create an activity using both the target tokens TO and TO_DN, the workflow sends out duplicate notification emails to the target users.

# Role Request Binding Activity

The Role Request Binding activity changes the approve or deny attribute in the nrfRequest object. Two such activities are required in any workflow of Flow Type Role Approval or SoD Approval. One Role Request Binding handles the approve condition from an Approval activity and the other handles the deny condition from an Approval activity. The Role Request Binding activities must be completed before the Finish activity or the workflow is considered invalid and cannot be deployed by Designer.

## Properties

The Role Request Binding activity has the following properties:

***Table 7-21*** *Role Request Binding Properties*

| Property | Description |
|---|---|
| **Name** | Provides a name for the activity. |
| **Action** | **approved**: Changes the approve attribute in the nrfRequest object to true. |
| | **denied**: Changes the deny attribute in the nrfRequest object to true. |

## Data Item Mapping

Not supported with this activity.

## Email Notification

Not supported with this activity.

# Role Request Activity

The Role Request activity allows you to automate the granting or revoking of roles to users, groups, or containers. For example, you might write a provisioning request definition that provisions all of the resources and roles a new employee needs on their first day. Using the role request activity, you can automate the approval of that employee for specified roles.

You can also configure the activity to respond to Separation of Duty (SoD) constraint overrides by always approving, or allowing specific cases. You can use the activity to configure the effective and expiration dates for the role, or use it to extend the expiration date of a role.

The Role Request activity runs within the system service security context.

There is no limit on the number of Role Request activities allowed within a workflow.

The Role Request activity fails if the requested role DN or the target DN is invalid, or does not exist.

The result of the role request is written as a system comment to the comment history.

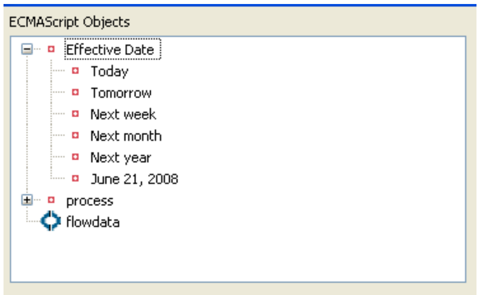
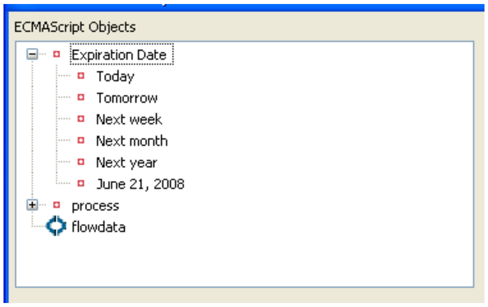
The Role Request activity does not support the ability to set the originator of the request. Use Simple Object Access Protocol (SOAP) calls rather than this activity when you need this information.

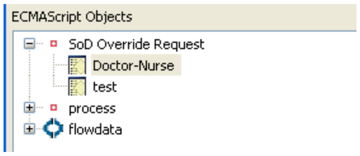
## Properties

The Role Request activity has the following properties:

***Table 7-22*** *Role Request Properties*

| Property Name | Description |
|---|---|
| **Name** | Required. Provides a localizable name for the activity. |
| **Description** | Required. Text that describes the reason for the assignment request. This corresponds to the **Initial Request Description** field of the **Request Roles Assignment** tab. |
| **Action** | Specifies the action the activity should perform. Select a value from the drop-down list. The values are:<br>◆ **grant (default):** Use this value if the role should be granted.<br>◆ **revoke:** Use this value if the role should be revoked.<br>◆ **extend:** Use this value to extend the expiration date of the specified role. The role must already be granted, and the value that you specify in Expiration Date must be later than the one currently specified. |
| **Roles** | Required. An expression that resolves to a list of requested roles. For information on building this expression, see “Specifying the Roles and Targets Properties” on page 215.<br>This is an example of the script to request a specific role:<br>`'CN=Administer Drugs,CN=Level10,CN=RoleDefs,CN=RoleConfig,CN=AppConfig,' + PROVISIONING_DRIVER`<br>In this script example, the value is retrieved from flowdata:<br>`flowdata.get('Start/request_form/role')` |

| Property Name | Description |
|---|---|
| Target Type | Required. Specifies the type of object that the requested role will be assigned. Choose one of the values from the list. The values are:<br>◆ user (default)<br>◆ group<br>◆ container<br>◆ container with subtree |
| Effective Date | The date when the role assignment goes into effect. If no date is specified, the assignment is effective immediately after it is requested. You can use the Expression Builder convenience methods to specify this value.<br> ECMAScript Objects<br>Effective Date<br>Today<br>Tomorrow<br>Next week<br>Next month<br>Next year<br>June 21, 2008<br>process<br>flowdata |
| Targets | Required. An expression that resolves to the DN of the object for whom the role is requested. The target can be users, groups, or containers depending on Target Type value. The targets that you specify must resolve to the Target Type specified.<br>For information on building this expression, see "Specifying the Roles and Targets Properties" on page 215.<br>The following examples show a script for targets:<br>`'cn=ablake,ou=users,ou=medical-idmsample,o=novell'`<br>To retrieve the value from flowdata:<br>`flowdata.get('Start/request_form/group')` |
| Expiration Date | The date when the role assignment expires. If not specified, the assignment remains in effect indefinitely. You can use the Expression Builder's convenience methods to specify this value.<br> ECMAScript Objects<br>Expiration Date<br>Today<br>Tomorrow<br>Next week<br>Next month<br>Next year<br>June 21, 2008<br>process<br>flowdata |
| Correlation ID | An optional string field. If not supplied, it defaults to the process instance ID. This string must be less than or equal to 64 characters. |

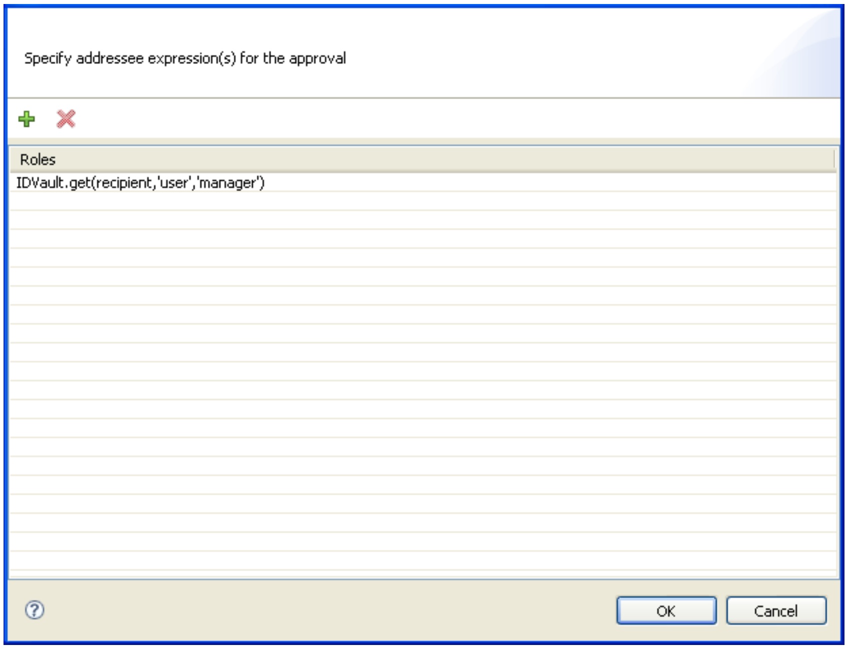
| Property Name | Description |
|---|---|
| **SoD Override Request** | Optional field. Defines how the Role Request activity should handle a request that causes an SoD constraint violation. Values are:<br>- **true:** SoD override is requested for all encountered conflicts.<br>- **false (default):** An SoD override is not requested for all encountered conflicts. Role Request activity uses the list of SoDs in the SoD Overrides property to determine which SoD constraints to override. |
| **Override Justification** | Optional field. Available when **SoD Override Request** is false. Describes why an exception to the SoD constraint is necessary. If no value is specified, the **Description** is used. This example shows how to retrieve the value from flowdata.<br>`flowdata.get('Start/request_form/reason')` |
| **SoD Overrides** | Available when the SoD Override Request is false. It is a list of one or more SoD constraints to override. When an SoD constraint is encountered and the constraint is in this list, the role request activity will request the role. It the SoD is not in this list, the role request activity will stop executing and follow the error link.<br>You can use the Expression Builder’s convenience methods to build the expression. The list contains the local list of SoDs defined for this project. For example:<br> ECMAScript Objects<br>SoD Override Request<br>Doctor-Nurse<br>test<br>process<br>flowdata<br>Selecting the Doctor-Nurse SoD generates an expression like this:<br>`'cn=Doctor-Nurse,cn=SoDDefs,cn=RoleConfig,`<br>`cn=AppConfig,' + PROVISIONING_DRIVER'` |

## Specifying the Roles and Targets Properties

Designer provides a convenient way to build the Roles and Targets expressions using the Expression Builder.

1 Click the button in the property’s **Value** column.

Designer launches this dialog box for adding or removing expressions.



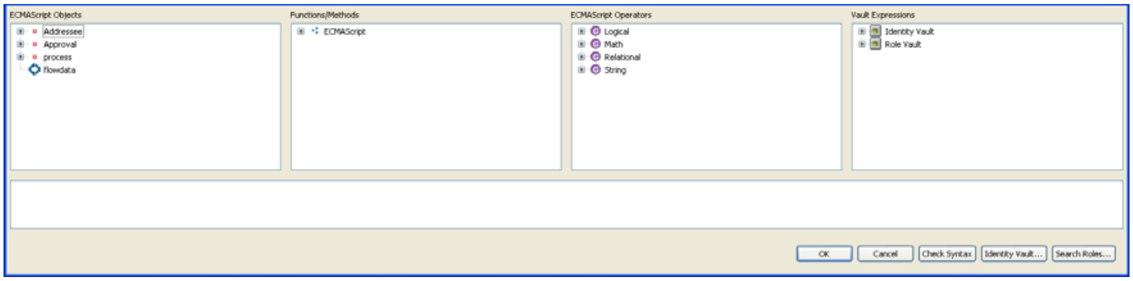
Specify addressee expression(s) for the approval

Roles

IDVault.get(recipient,'user','manager')

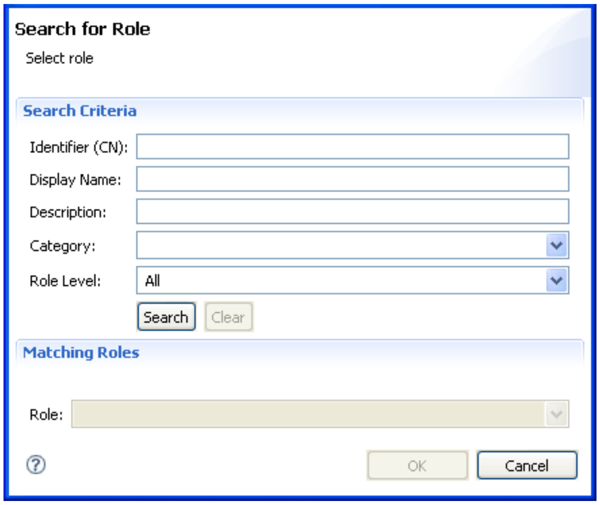
2 Click + to add a new Roles or Targets expression by using the Expression Builder.

The dialog box displayed by Designer varies depending on whether you are specifying Roles or Targets. This dialog shows an example of the dialog box displayed to specify Roles because it includes the **Search Roles** button.



You can choose one of the ECMAScript Objects to build the Roles or Targets expression, or use the **Identity Vault** button to select a specific object. Click **Search Roles** to locate a role.

**2a** To choose specify a Role, click **Search Roles**.



Search for Role

Select role

Search Criteria

Identifier (CN):

Display Name:

Description:

Category:

Role Level: All

Search Clear

Matching Roles

Role:

OK Cancel

**2b** In the dialog box, specify the **CN**, **Display Name**, **Description**, **Role Category**, and **Role Level** on which you want to search.

For **CN**, **Display Name**, and **Description**, you can enter a wildcard (such as S*, *S) or regular expressions (such as [A-Z][a-z]*).

You can enter a value for all of the fields or none of the fields. If you do not supply a value in a particular field, the search returns all of the possible values for that field. If you enter a value in one or more of the fields, the values are ANDed together to create the search filter. The search occurs on the roles defined locally. Roles matching the search criteria are displayed in the **Matching Roles** selection list.

**2c** Select a role from the **Roles** selection list, then click **OK**. The role is added to the expression area.

**3** Click **OK** after you are satisfied with expression. Repeat Step 2 to continue to add more expressions.

## Data Item Mapping

Not supported with this activity.

## Email Notification

Not supported with this activity.

# Resource Request Binding Activity

The Resource Request Binding activity changes the approve or deny attribute in the nrfResourceRequest object. Two such activities are required in any workflow of Flow Type Resource Approval. One Resource Request Binding handles the approve condition from an Approval activity and the other handles the deny condition from an Approval activity. The Resource Request Binding activities must be directly before the Finish activity or the workflow is considered invalid and cannot be deployed by Designer.

## Properties

The Resource Request Binding activity has the following properties:

*Table 7-23 Resource Request Binding Properties*

| Property | Description |
|---|---|
| **Name** | Provides a name for the activity. |
| **Action** | **approved**: Changes the approve attribute in the nrfResourceRequest object to true.<br>**denied**: Changes the deny attribute in the nrfResourceRequest object to true. |

## Data Item Mapping

Not supported with this activity.

## Email Notification

Not supported with this activity.

# Resource Request Activity

The Resource Request activity allows you to automate the granting or revoking of resources to users. For example, you might write a provisioning request definition that provisions all of the resources a new employee needs on their first day. Using the resource request activity, you can automate the approval of that employee for specified resources.

The Resource Request activity runs within the system service security context.

There is no limit on the number of Resource Request activities allowed within a workflow.

The Resource Request activity fails if the requested resource DN or the target DN is invalid, or does not exist.

The result of the resource request is written as a system comment to the comment history.

The Resource Request activity does not support the ability to set the originator of the request. Use SOAP calls rather than this activity when you need this information.

## Properties

The Resource Request activity has the following properties:

*Table 7-24 Role Request Properties*

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

| Property Name | Description |
|---|---|
| **Resources** | Required. An expression that resolves to a list of requested resources. For information on building this expression, see "Specifying the Resource and Targets Properties" on page 218.<br>This is an example of the script to request a specific resource:<br>`'CN=Administer Drugs,CN=ResourceDefs,CN=ResourceConfig,CN=AppConfig,' + PROVISIONING_DRIVER`<br>In this script example, the value is retrieved from flowdata:<br>`flowdata.get('Start/request_form/resource')` |
| **Description** | Required. Text that describes the assignment request. This corresponds to the **Initial Request Description** field of the **Request Resources Assignment** tab. |
| **Action** | Specifies the action the activity should perform. Select a value from the drop-down list. The values are:<br>◆ **grant (default):** Use this value if the resource should be granted.<br>◆ **revoke:** Use this value if the resource should be revoked.<br>◆ **extend:** Use this value to extend the expiration date of the specified resource. The resource must already be granted, and the value that you specify in Expiration Date must be later than the one currently specified. |
| **Correlation ID** | An optional string field. If not supplied, it defaults to the process instance ID. This string must be less than or equal to 64 characters. |
| **Targets** | Required. An expression that resolves to the DN of the object for whom the resource is requested. The target must be an object of the User class only. The targets that you specify must resolve to the Target Type specified.<br>For information on building this expression, see "Specifying the Resource and Targets Properties" on page 218.<br>The following examples show a script for targets:<br>`'cn=ablake,ou=users,ou=medical-idmsample,o=novell'` |
| **Entitlement Params** | Optional. 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. |

## Specifying the Resource and Targets Properties

Designer provides a convenient way to build the Resource and Targets expressions by using the Expression Builder.

1. Click the button in the property's **Targets** or **Entitlement Params** column.

   Designer launches this dialog box for adding or removing expressions.

2. Click + to add a new Resource or Targets expression by using the Expression Builder.

   You can choose one of the ECMAScript Objects to build the Resource or Targets expression, or use the **Identity Vault** button to select a specific resource.

3 Click **OK** after you are satisfied with expression. Repeat Step 2 to continue to add more expressions.

**IMPORTANT:** You cannot specify values to the resource request form fields at the time of resource request activity through Designer.

## Data Item Mapping

Not supported with this activity.

## Email Notification

Not supported with this activity.

# Start Workflow Activity

The Start Workflow activity allows you to invoke a workflow instance from within a provisioning request definition. The workflows you invoke branch; they are not subflows.

You can group these related workflows by using the Correlation ID.

## Properties

The Start Workflow activity has the following properties:

***Table 7-25*** *Start Workflow Properties*

| Property | Description |
|---|---|
| **Activity Id** | Specify a unique string value that identifies the activity. Activity Ids are written to the user application's log file. Specifying a meaningful Activity Id makes it easier to understand the data written to the logs. You can specify letters, numbers, and the underscore (_) character.<br>If you do not specify a value, the Activity Id defaults to `Activity`*`NN`*, where the *NN* represents the order in which the activity was added to the workflow. |
| **Name** | Provides a name for the activity. |
| **Provisioning Request Defn** to start | An expression that resolves to the distinguished name of the provisioning request definition to invoke. You can select the provisioning request from the Expression Builder. The possible selections are provisioning requests whose Status is **Active** and whose Process Type is **Normal**. |
| **Recipient** | An expression that resolves to one or more user or group distinguished names.<br>To specify multiple values, click **Recipient List** (in the ECMAScript Objects panel of the Expression Builder). This generates a pre-built function called multirecipient(). Replace *'Enter recipient'* with the distinguished name of a user or group as needed to include all of the recipients for the workflow.<br>A separate workflow process is created for each recipient that you specify. |

| Property | Description |
|---|---|
| **Correlation ID** | An expression that lets you group the workflows invoked from this activity. If you do not supply a value, the workflow engine supplies a default, which is the request ID of the current workflow. |

## Data Item Mapping

To pass the data needed by the workflow you want to start, you must specify data item mappings. The data items you must specify depend on the workflow that is to be started. The Data Item Mapping view displays the fields required by the **Provisioning Request Defn to start**. Because the Provisioning Request Defn to start is an expression, it might evaluate properly only at runtime. This would be the case in a workflow where the user could choose a workflow to start from a form field, and that value gets mapped to flowdata, and that flowdata expression is used in the Provisioning Request Defn to start property. To account for this, the Data Item Mapping view changes based on whether the workflow to start can be determined at design time. If so, then the data items to start that workflow are shown. However, if the workflow is not known at design time, then the workflow developer must specify them.

The Start Workflow activity has the following data item mappings:

***Table 7-26*** *Start Workflow Data Item Mapping Properties*

| Setting | Description |
|---|---|
| **Source Expression** | An expression used to initialize the data items needed by the **Provisioning Request Defn to Start**. When you click a cell in the **Source Expression** column, the ECMA Expression Builder displays to help you define your expression. |
| **Target Form Field** | A read-only field that specifies target fields for the initialization data specified in the Source Expression. If the workflow cannot be determined at design time, this field is editable. |
| **Data Type** | Specify the data type for this data item. |
| **Multivalued** | A read-only field that specifies if the workflow can accept multiple values for this parameter. If this value is true, you can specify multiple values by using the syntax provided in the Expression Builder. If the workflow cannot be determined at design time, this field is not present. |

## Email Notification

Not supported with this activity.

# Finish Activity

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

- "Properties" on page 221
- "Data Item Mapping" on page 221
- "Email Notification" on page 221

## Properties

The Finish activity has the following properties:

***Table 7-27** Finish Activity Properties*

| Property | Description |
|---|---|
| **Activity Id** | Specify a unique string value that identifies the activity. Activity Ids are written to the user application's log file. Specifying a meaningful Activity Id makes it easier to understand the data written to the logs. You can specify letters, numbers, and the underscore (_) character.<br>If you do not specify a value, the Activity Id defaults to `Activity`*`NN`* where the *NN* represents the order in which the activity was added to the workflow. |
| **Name** | Provides a name for the activity. |
| **Notify by E-Mail** | Provides a method of triggering an email notification when the Finish activity is executed. When this property is set to True, an email notification is sent. When this property is set to False, no email notification is sent.<br>See "Email Notification" on page 221 for information about setting up the email notification. |

## Data Item Mapping

Not supported with this activity.

## Email Notification

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

***Table 7-28** E-Mail Notification Settings for the Finish Activity*

| Setting | Description |
|---|---|
| **E-Mail Template** | Specifies the name of the email template to use. By default, the Finish activity uses the Provisioning Approval Completed Notification template.<br>You can edit an email template in Designer. See "Editing an email template:" on page 158 for more information. |

| Setting | Description |
|---|---|
| **Source**<br>**Target** | Specifies the source expressions for target tokens in the email body.<br>The list of target tokens is determined by the selected email 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>The available target tokens for the Provisioning Approval Completed Notification email template are listed below:<br>◆ TO<br>◆ CC<br>◆ BCC<br>◆ REPLYTO<br>◆ TO_DN<br>◆ CC_DN<br>◆ BCC_DN<br>◆ requestStatus<br>◆ requestSubmissionTime<br>◆ requestID<br>◆ recipientFullName<br>◆ initiatorFullName<br>◆ requestTitle<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 abstraction layer (IDVault object). You can modify these expressions to suit your application requirements.<br>For details on building ECMA expressions, see Chapter 9, “Working with ECMA Expressions,” on page 271. |

**NOTE**

- Email notification is supported only when the **Notify participants by E-Mail** check box is selected on the **Overview** tab.
- When you create a workflow for use with the Resource Request portlet, and you use _default_ as the expression for the TO token, the addressee expression must be an IDVault expression.
- If you create an activity using any of the target tokens TO_DN, CC_DN, or BCC_DN, you must specify a user’s DN or an expression that resolves to a user’s DN as the source expression for the token.
- If you create an activity using both the target tokens TO and TO_DN, the workflow sends out duplicate notification emails to the target users.

# Rest Activity

The Rest activity enables users to call REST endpoints or resources when processing workflow data. The activity allows workflows to exchange data with arbitrary REST services. Data sent to a REST service can integrate a workflow with other systems inside and outside the organization. Data received from a REST service can provide decision support information on approval forms. You create flowdata variables to move data between the workflow and the REST service.

## Properties

The Rest activity has the following properties. Note that most activity properties are ECMAScript expressions, so ensure that you configure each property by clicking the "E" icon in the property field to open the ECMA Expression Builder.

***Table 7-29*** *Rest Activity Properties*

| Property Name | Description |
|---|---|
| **Activity Id** | Specify a unique string value that identifies the activity. Activity Ids are written to the user application's log file. Specifying a meaningful Activity Id makes it easier to understand the data written to the logs. You can specify letters, numbers, and the underscore (_) character.<br>If you do not specify a value, the Activity Id defaults to `Activity`*`NN`* where the *NN* represents the order in which the activity was added to the workflow. |
| **Name** | Provides a name for the activity. |
| **Protocol** | Specifies the protocol the activity uses when calling the REST server. You can specify either **http** or **https**.<br>**NOTE:** If you specify https, you must also configure the **Trust Manager** property for the activity. |
| **Host** | Specifies the REST server the activity calls to request or modify data. |
| **Port** | Specifies the port number the activity uses when calling the REST server. |
| **Path** | Specifies the encoded URL path the activity uses when calling the REST server. If the path includes any reserved characters, you must URL encode the path. For more information about URL encoding, see http://en.wikipedia.org/wiki/Percent-encoding (http://en.wikipedia.org/wiki/Percent-encoding).<br>In the ECMA Expression Builder, select **URL Path Encoding** > **URL encode the path** and modify the path to include the URL path expression to encode. |

| Property Name | Description |
|---|---|
| **Method** | Specifies the method the activity uses to retrieve data from or modify data on the REST server. The choices are:<br>◆ get<br>◆ put<br>◆ post<br>◆ delete<br>◆ head |
| **Authorization Header** | Specifies the Authorization header the activity uses when calling the REST server. In many cases, the REST server expects the header to be in the “Basic Authentication” format. For more information about the “Basic Authentication” format, see http://en.wikipedia.org/wiki/Basic_authentication (http://en.wikipedia.org/wiki/Basic_authentication).<br>If this is the case, in the ECMA Expression Builder, select **Basic Auth Header** and modify the header to include the username and password used to access the REST server. |
| **Accept Header** | Specifies the Accept header the activity uses when receiving data from the REST server. The Accept header tells the REST server the format in which it should return data. The activity can receive data in either XML or JSON format. The choices are:<br>◆ application/json<br>◆ application/xml |
| **Http Headers** | Specifies any additional HTTP headers the activity uses when calling the REST server. Specify both header name and value expressions, as necessary, using quotation marks around each expression. |
| **Timeout** | Specifies a dynamic expression that defines the period of time, in milliseconds, allotted for the Rest activity to complete. The timeout interval applies each time the activity is executed by the addressee.<br>For details on building ECMA expressions, see Chapter 9, “Working with ECMA Expressions,” on page 271. For descriptions of the system variables available in a workflow, see “Understanding Workflow Data” on page 163. |
| **Trust Managers** | If the activity uses the https protocol, this property specifies one or more trust managers used to authenticate the connection to the REST server. The property expression must evaluate to `TrustManager`, `TrustManager[]`, or `List<TrustManager>`.<br>The default choices available in the ECMA Expression Builder are:<br>◆ Trust All Certs<br>◆ Trust Certs in default keystore<br>◆ Trust Certs in specified keystore |

## Data Item Mapping

To bind the data items associated with the Rest activity, you define pre-activity and post-activity mappings. The pre-activity mappings map values retrieved from the flowdata object to attributes in the Input message for the REST server that will be accessed by the Rest activity. The post-activity mappings map the response from the REST server back into the flowdata object.

***Table 7-30*** *Rest Activity Data Item Mappings*

| Setting | Description |
|---|---|
| **Pre-Activity** | If the REST service you want to call requires payload data, you need to specify pre-activity mappings. This is normally the case when you use a PUT or POST request.<br>When this option is selected, you can double-click a cell in the **Source Expression** column to specify where the Rest activity gets data for a particular REST server input field. In particular, you can set a source expression for Content Type and Content, as follows:<br>◆ **Content Type:** The value of this type of expression is normally either `application/json` or `application/xml`, depending on the data type you want to send to the server.<br>◆ **Content:** This type of expression evaluates to a JSON- or XML-formatted string for the expected payload. If the expected format is JSON, look at the helper functions in the ECMA Expression Builder, under **Vault Expressions** > **Script Vault**. In particular, the helper function **Convert object to JSON** may be useful, as it enables you to create a JSON string from an ECMAScript object.<br>While you can build the JSON string using conventional expressions, the **Convert object to JSON** function may be easier and less prone to errors, especially for complex JSON expressions. See the following example for one possible expression using the function:<br>`ScriptVault.JSON.stringify( (function () { var test = {}; test.val = "1"; test.arrayVal = ["one","two"]; return test;} () ))` |

| Setting | Description |
|---|---|
| Post-Activity | If the REST service returns data you want to capture, you need to specify post-activity mappings. This is normally the case for most requests.<br>When this radio button is selected, you can double-click a cell in the Target Expression column to specify where data from a REST server output field should be copied after the form has been processed. In particular, you can set a source expression for Status Code, Content Type, and Content, as follows:<br>- **Status Code:** The value of this expression is the HTTP status code from the REST call.<br>- **Content Type:** The value of this type of expression is normally either `application/json` or `application/xml`, depending on the format of the data the server returns.<br>- **Content:** This type of expression evaluates to a JSON- or XML-formatted string the returned data. If the format of the returned data is JSON, look at the helper functions in the ECMA Expression Builder, under Vault Expressions > Script Vault. In particular, the helper function Convert object to JSON may be useful, as it enables you to create a JSON string from an ECMAScript object.<br>You can use the Map All button to map the Content field into flowdata in the Rest activity, then use the content by extracting fields in a subsequent activity using the Convert object to JSON helper function. For example, in a workflow where you have mapped the content returned from the REST call to flowdata.processInstances, you can use the following expression to obtain the JSON property `totalSize`:<br>`(function () { var processInstances =`<br>`ScriptVault.JSON.parse(`<br>`flowdata.get('processInstances') ); var size =`<br>`processInstances.totalSize; return size;}) ();` |
| Source Expression | Specifies a source expression. When you click a cell in the Source Expression column, the ECMA Expression Builder displays to help you define your expression. It is recommended that you click the Map All button to allow Designer to generate this expression for you. |
| Target Expression | Specifies a target expression. When you click a cell in the Target Expression column, the ECMA Expression Builder displays to help you define your expression. It is recommended that you click the Map All button to allow Designer to generate this expression for you. |

## Email Notification

Not supported with this activity.

# Integration Activity

The Integration activity provides a way to use a Web service to process workflow data. For detailed information about using the Integration activity, see Chapter 8, “Working with Integration Activities,” on page 235.

- “Properties” on page 227
- “Data Item Mapping” on page 228
- “Email Notification” on page 229

## Properties

The Integration activity has the following properties.

***Table 7-31*** *Integration Activity Properties*

| Property Name | Description |
|---|---|
| **Activity Id** | Specify a unique string value that identifies the activity. Activity Ids are written to the user application’s log file. Specifying a meaningful Activity Id makes it easier to understand the data written to the logs. You can specify letters, numbers, and the underscore (_) character.<br>If you do not specify a value, the Activity Id defaults to Activity*NN* where the *NN* represents the order in which the activity was added to the workflow. |
| **Name** | Provides a name for the activity. |
| **WSDL Resource** | Specifies a Web Services Description Language (WSDL) file for the Web service to be used in the Integration activity. After it is specified, the WSDL is incorporated into the provisioning request definition file.<br>When you select a WSDL file, a dialog box is displayed that you use to select the Web service port type and operation that you want to use in the Integration activity.<br>You can also specify a SOAP endpoint for the Integration activity and configure authentication for the SOAP service. For more information about specifying a WSDL file and configuring the Integration activity, see “Adding an Integration Activity” on page 235. |
| **Timeout** | Specifies a dynamic expression that defines the period of time allotted for the Integration activity to complete. The timeout interval applies each time the activity is executed by the addressee.<br>For details on building ECMA expressions, see Chapter 9, “Working with ECMA Expressions,” on page 271. For descriptions of the system variables available in a workflow, see “Understanding Workflow Data” on page 163. |

| Property Name | Description |
|---|---|
| **Retry Count** | Specifies the number of times to retry the activity in the event of a timeout.<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. If the last retry times out, the activity can be marked as success, fault, error, or timed out, depending on the final timeout action specified for the activity. |
| **Final Timeout Action** | Determines the final state of the request in the event that the Integration activity times out. The choices are:<br>◆ success<br>◆ fault<br>◆ error<br>◆ timedout |

## Data Item Mapping

To bind the data items associated with the Integration activity, you define pre-activity and post-activity mappings. The pre-activity mappings map values retrieved from the flowdata object to attributes in the Input message for the Web service that will be accessed by the Integration activity. The post-activity mappings map the response from the Web service back into the flowdata object. For more information about data item mapping for Integration activities, see "Moving Data to and from the Integration Activity" on page 237.

***Table 7-32** Integration 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 Integration activity gets data for a particular Web service input field.<br>**NOTE:** When the **Pre-Activity** option is selected, the cells in the **Web Service Input 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 a cell in the **Target Expression** column to specify where data from a Web service output field should be copied after the form has been processed.<br>**NOTE:** When the **Post-Activity** option is selected, the cells in the **Web Service Output Field** column are not editable. |
| **Source Expression** | Specifies a source expression. When you click a cell in the **Source Expression** column, the ECMA Expression Builder displays to help you define your expression. For example, `flowdata.get('Start/RequestRate/Country1')` for a Web service input, or `flowdata.Start/RequestRate/Country1` for a Web service output field. |

| Setting | Description |
|---|---|
| **Web Service Input Field** | This column displays all of the input fields for the port type and operation specified when the WSDL file was selected. The fields in this column are automatically populated. If you want to remove an input field, click **Mapping**, expand the nodes of the sample document and deselect any input fields that you want to remove. |
| **Web Service Output Field** | This column displays all of the output fields for the port type and operation specified when the WSDL file was selected. The fields in this column are automatically populated. If you want to remove an output field, click **Mapping**, expand the nodes of the sample document and deselect any output fields that you want to remove. |
| **Mapping** | Displays a hierarchical view of the sample document for the inputs to or outputs from the Web service. You can use this feature to deselect input or output fields (by default, all Web service input and output fields are selected). |

## Email Notification

Not supported with this activity.

# 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 7-33*** *Entitlement Activity Properties*

| Property Name | Description |
|---|---|
| **Activity Id** | Specify a unique string value that identifies the activity. Activity Ids are written to the user application's log file. Specifying a meaningful Activity Id makes it easier to understand the data written to the logs. You can specify letters, numbers, and the underscore (_) character.<br>If you do not specify a value, the Activity Id defaults to Activity*NN* where the *NN* represents the order in which the activity was added to the workflow. |
| **Name** | Provides a name for the activity. |
| **Set Workflow Status** | Specifies the approval status of the provisioning request. Set to True for approved; otherwise, set to False. This method of setting workflow status overrides other methods (for example, the **Set Default Completion Status to Approved** parameter (see Table 4-3, "Overview Properties," on page 86) or the **Approval Status** activity (see "Workflow Status" on page 208). |

## Data Item Mapping

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

***Table 7-34*** *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>The DirXML mappings for the Entitlement are described below:<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>`'CN=Groups,CN=GroupEntitlementLoopback,CN=TestDrivers,O=novell'`<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>`'\\MYTREE\\novell\\idmsample-doc\\groups\\Sales'`<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 271.

## Email Notification

Not supported with this activity.

# 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 (see "Working with Entity Activities" on page 233).

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

- "Properties" on page 231
- "Data Item Mapping" on page 232

- "Email Notification" on page 233
- "Working with Entity Activities" on page 233

## Properties

The Entity activity has the following properties:

***Table 7-35*** *Entity Activity Properties*

| Property Name | Description |
|---|---|
| **Activity Id** | Specify a unique string value that identifies the activity. Activity Ids are written to the user application's log file. Specifying a meaningful Activity Id makes it easier to understand the data written to the logs. You can specify letters, numbers, and the underscore (_) character.<br>If you do not specify a value, the Activity Id defaults to Activity*NN* where the *NN* represents the order in which the activity was added to the workflow. |
| **Name** | Provides a name for the activity. |
| **Entity Type** | Specifies the target entity type: User or Group. |
| **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**. |
| **Set Workflow Status** | Specifies the approval status of the provisioning request. Set to True for approved; otherwise, set to False. This method of setting workflow status overrides other methods. For example, the **Set Default Completion Status to Approved** parameter (see Table 4-3, "Overview Properties," on page 86) or the **Approval Status** activity (see "Workflow Status" on page 208). |

## 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 7-36** 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>To create a new object, specify a distinguished name that does not yet exist.<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. After 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>`flowdata.get('groupdn');`<br>For details on using the DNMaker control, see "DNMaker" on page 115. |
| Modify Type | Indicates how the mapping should be performed for an attribute. The choices are<br>◆ Append Value<br>◆ Replace Value<br>◆ Replace All Values<br>For many attributes, **Replace Value** is the only option that makes sense; therefore, this option is selected automatically and cannot be changed.<br>You must specify the **Modify Type** setting before specifying the **Modify Value Expression** setting. |

| Setting | Description |
|---|---|
| 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>Designer automatically inserts a sample ECMAScript expression into this field. The code provided varies depending on the **Operation** property specified in **Properties** and the **Modify Type** selected in **Data Item Mapping**. For example, if you have specified **Create/Modify** for **Operation**, and **Replace All Values** for **Modify Type**, Designer inserts an expression that helps you to create a vector:<br>`function list() { v=new java.util.Vector(); v.add('{Enter Item 1}'); v.add('{Enter Item 2}'); return v; };  list();`<br>In some cases you might be able to create expressions that work as well or better than the sample expression. For example, instead of creating a vector for multiple attribute values, you can create a flowdata variable (see "Understanding Workflow Data" on page 163) to store multiple attribute values, and use the getObject function to retrieve the values of the flowdata variable (see "ECMAScript Objects" on page 273).<br>**NOTE:** The cells in the **Target Attribute** column are not editable. |

## Email Notification

Not supported with this activity.

## 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 so this section includes example procedures for:

- "Adding or Modifying an Entity" on page 233
- "Using an Entity Activity to Delete an Entity" on page 234
- "Using an Entity Activity to Delete an Attribute or Value" on page 234

### Adding or Modifying 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, then 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, then 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 “Working with Entities and Attributes” on page 48 for information about adding entities. If you are adding an entity, you must enter expressions for all required attributes.

## 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, then 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, then 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.

## 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 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 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, then specify an expression that resolves to the value of the attribute that you want to delete.

# 8 Working with Integration Activities

This section provides details about working with Integration activities.

## About the Integration Activity

The Integration activity is an activity that allows workflows to exchange data with arbitrary Web services. Data sent to a Web service can integrate an individual workflow with other systems, inside and outside the organization. Data received from a Web service can provide decision support information on approval forms.

You create flowdata variables to move data from the workflow to the Web service for processing. The Integration activity automatically creates an action model for working with a Web service based on a WSDL document that you specify.

**NOTE:** The action model is a subset of the features available in the Novell Integration Manager product (formerly known as Novell exteNd Composer).

An action model is a visual representation of a set of instructions for processing XML documents and communicating with XML data sources. An action model performs all data mapping, data transformation, and data transfer within an Integration activity. You can edit the action model to manipulate data before and after the data is submitted to the Web service. You then map the data from the Integration activity back to flowdata variables for use in the workflow.

## Adding an Integration Activity

1. Create a provisioning request definition (see Chapter 4, “Configuring Provisioning Request Definitions,” on page 77).
2. Create a workflow for the provisioning request definition (see Chapter 6, “Creating the Workflow for a Provisioning Request Definition,” on page 149).
3. Click the **Workflow** tab.
4. Drag an Integration activity from the palette and place it in the desired location in the workflow.
5. Click the **Properties** tab.
6. Type a name for the activity in the **Name** field.
7. Click the **Value** field for the **WSDL Resource** property, then click the browse button to display a dialog box that you use to locate the WSDL file for the Web service that you want to access with the Integration activity.
8. Use the dialog box to browse your file system to locate the WSDL file for the Web service that you want to use. Click the name of the WSDL file, then click **Open**.

   A dialog box that you use to select a port type and operation for the Web service is displayed.

   The **Select Port Type** list includes a set of port types supported by the Web service. Each port type supports operations that include the input and output messages of the operation.

This window allows you to specify the SOAP endpoint, the user ID used to access the SOAP endpoint, and the password used to access the SOAP endpoint. These options are all ECMAScript expressions and can be GCV values. Use the ECMA Expression Builder to configure the expressions you want to use.

**NOTE:** The **Use new WSDL Generation type** option enables Designer to build the Integration activity using an XML Interchange action instead of a WS Interchange action. It is recommended that you leave this option selected, because the WSDL parsing is more robust, and the resulting Integration activity does not require the WSDL document be maintained as part of the provisioning request definition and is smaller in size.

**9** Select a port type from the **Select Port Type** list.

**10** Select an operation from the **Select Operation** list.

**11** ***If you want to specify a SOAP endpoint,*** specify a SOAP endpoint URL, either by selecting the URL from the **Select Soap Service Endpoint Expression** list or by clicking the “E” icon and using the ECMA Expression Builder to configure an expression that resolves to the SOAP endpoint URL.

**NOTE:** Ensure that you specify the private key in DER format (PKCS8) and without a password. You can convert the private key from PKS12 to PKCS8 format by running an OpenSSL command. For example, `openssl pkcs8 -topk8 -inform PEM -outform DER -in key.pem -nocrypt > key`

**12** ***If you want to use basic authentication for a SOAP endpoint,*** complete the following steps:

**12a** Select **SOAP Service requires Basic Authentication**.

**12b** In the **User ID Expression** field, click the “E” icon and use the ECMA Expression Builder to specify an expression that resolves to the user ID used to access the SOAP endpoint.

**12c** In the **Password Expression** field, click the “E” icon and use the ECMA Expression Builder to specify an expression that resolves to the password used to access the SOAP endpoint.

**NOTE:** As a best practice, it is recommended that you use password-ref GCV for passwords. For information about additional best practices about managing passwords, see Managing Passwords in the *NetIQ Identity Manager Security Guide*. To retrieve a named password over LDAP from a workflow, see Allowing a Named Password to be Retrieved over LDAP in the *NetIQ Identity Manager - Administrator’s Guide to the Identity Applications*.

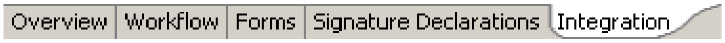
**13** Click **OK**.

The Integration activity creates an action model based on the WSDL document.You use the action model at design time to test the input to the Web service, test the response from the Web service, and map and transform data, if necessary, before returning the data to the workflow.

For many Web services, you don’t need to concern yourself with the action model. You simply create data item mappings for the Integration activity. After the action model is created, a new tab, **Integration**, is added to the provisioning request definition editor. You use this tab to access the action model.

**14** Specify the **Timeout**, **Retry Count**, and **Final Timeout Action** properties (see “Integration Activity” on page 227).

**15** If you want to view or edit the action model, click the **Integration** tab.



# Moving Data to and from the Integration Activity

1 Create form fields to allow users to provide input to the Web service accessed by the Integration activity (see Chapter 5, “Creating Forms for a Provisioning Request Definition,” on page 95). For example, if you are working with a Web service that provides stock quotes, you need a field for the user to specify a stock symbol.

2 To move user input from the form to the workflow, create a flowdata variable in an activity that precedes the Integration activity in the workflow.

  See “Understanding Workflow Data” on page 163 for information about creating flowdata variables.

  For example, if you have created a form field called “symbol” to accept a stock symbol for input to the Web service, you would go to the post-activity data item mapping for the activity associated with the form that contains the **symbol** field, then you would map the **symbol** field to a flowdata variable (for example, flowdata.symbol).

3 In the **Workflow** tab, right-click the icon for the Integration activity, then choose **Show Data Item Mapping**.

  The **Data Item Mapping** tab is displayed.

4 In the **Data Item Mapping** view, select **Pre-Activity**.

  In the **Web Service Input Field** grid, you should see fields that match all of the input fields associated with the port type and operation specified in Step 9 and Step 10 on page 236.

  The integration activity automatically selects all of the input field associated with the port type and operation. You can remove the input fields or modify properties of the input fields by following this procedure:

  4a Click **Mapping**.

    The **Sample Document** dialog box is displayed.

  4b Expand the nodes of the sample document and deselect any input fields that you want to remove.

  4c If an input field is an unbounded element, right-click and select **Occurs**.

  4d Type the maximum number in the **Enter Repeats** dialog, and click **OK**

  4e Click **OK** to return to the **Data Item Mapping** view.

5 For each **Web Service Input Field**, click in the **Source Expression** field, then click the ECMA Expression Builder button.

  The ECMA Expression Builder is displayed.

6 Expand the **flowdata** node in the **ECMAScript Objects** pane of the ECMA Expression Builder, then double-click the flowdata variable for the user input to the Web service.

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

8 Select **Post Activity**.

  In the **Web Service Output Field** grid, you should see fields that match all of the output fields associated with the port type and operation specified in Step 9 and Step 10 on page 236.

9 The Integration activity automatically selects all of the output fields associated with the port type and operation. If you want to remove some of the output fields, follow these steps:

  9a Click **Mapping**.

    The **Sample Document** dialog box is displayed.

- **9b** Expand the nodes of the sample document and deselect any attributes that you want to remove.
- **9c** Click **OK** to return to the **Data Item Mapping** view.

**10** Click **Map All** to automatically create flowdata variables for each **Web Service Output Field**.

Alternatively, for each **Web Service Output Field**, click in the **Source Expression** field, then click the ECMA Expression Builder button.

**11** Expand the **flowdata** node in the **ECMAScript Objects** pane of the ECMA Expression Builder, then double-click the flowdata variable that will receive data from the Web service.

**12** If you want to configure the Integration activity to provide more detailed information about any potential SOAP faults that might be encountered during the SOAP call of the activity, select **Fault Maps** in the **Data Item Mapping** view and click **Map All**. If Identity Manager encounters a SOAP fault, the Integration activity executes the fault maps to provide further details.

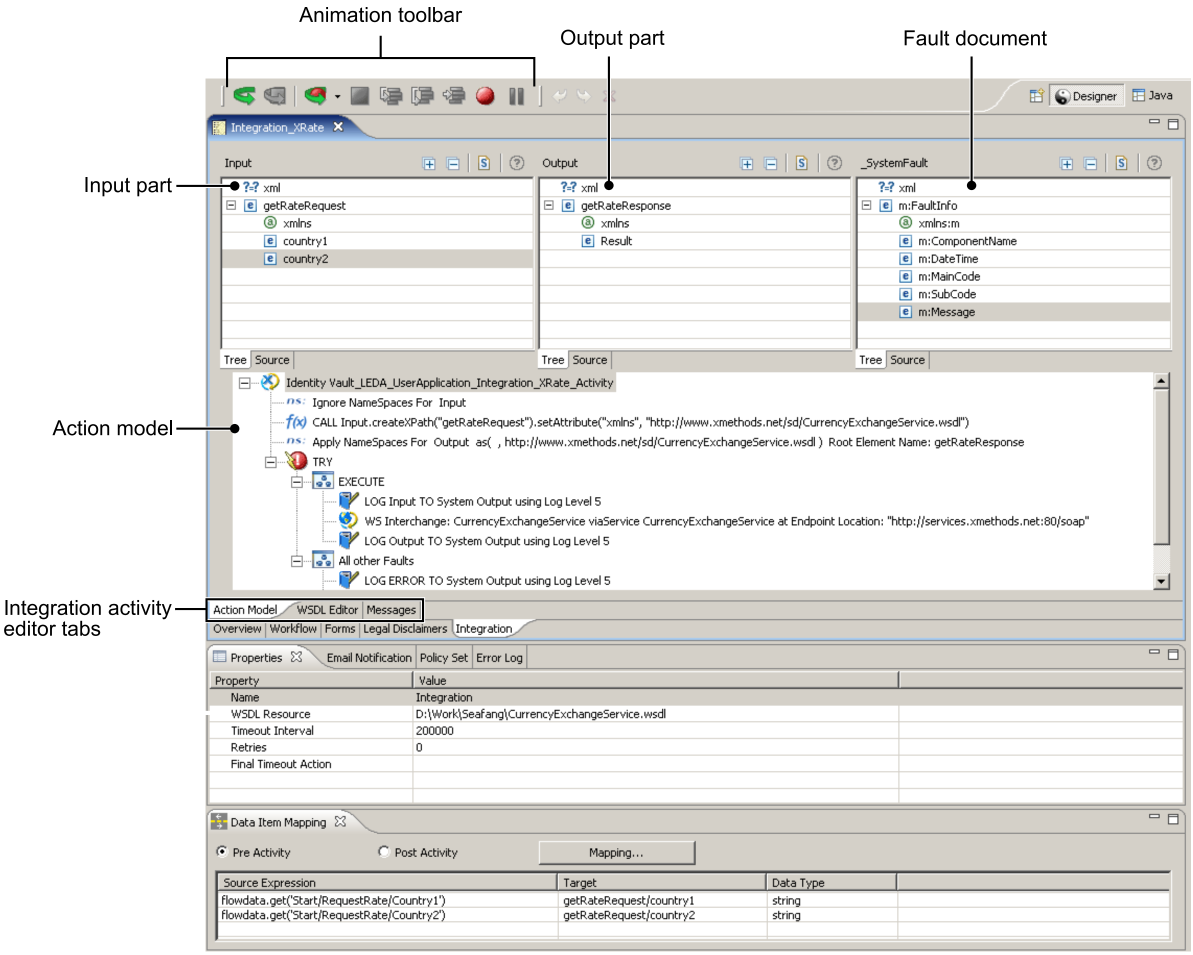
**13** Click **OK** to close the ECMA Expression Builder.

Now you can work in the Integration view to test and refine the interaction with the Web service.

# Using the Integration Activity Editor Interface

The Integration activity editor provides a working environment for the input, output, and actions of the Integration activity. The Integration activity editor is composed of three views: Action Model, WSDL Editor, and Messages.

*Figure 8-1 Integration Activity Interface*



```
Identity Vault_LEDA_UserApplication_Integration_XRate_Activity
    ns: Ignore NameSpaces For  Input
    f(x) CALL Input.createXPath("getRateRequest").setAttribute("xmlns", "http://www.xmethods.net/sd/CurrencyExchangeService.wsdl")
    ns: Apply NameSpaces For  Output  as(  , http://www.xmethods.net/sd/CurrencyExchangeService.wsdl )  Root Element Name: getRateResponse
    TRY
        EXECUTE
            LOG Input TO System Output using Log Level 5
            WS Interchange: CurrencyExchangeService viaService CurrencyExchangeService at Endpoint Location: "http://services.xmethods.net:80/soap"
            LOG Output TO System Output using Log Level 5
        All other Faults
            LOG ERROR TO System Output using Log Level 5
```

| Property | Value |
|---|---|
| Name | Integration |
| WSDL Resource | D:\Work\Seafang\CurrencyExchangeService.wsdl |
| Timeout Interval | 200000 |
| Retries | 0 |
| Final Timeout Action | |

● Pre Activity ○ Post Activity

| Source Expression | Target | Data Type |
|---|---|---|
| flowdata.get('Start/RequestRate/Country1') | getRateRequest/country1 | string |
| flowdata.get('Start/RequestRate/Country2') | getRateRequest/country2 | string |

# XML Views

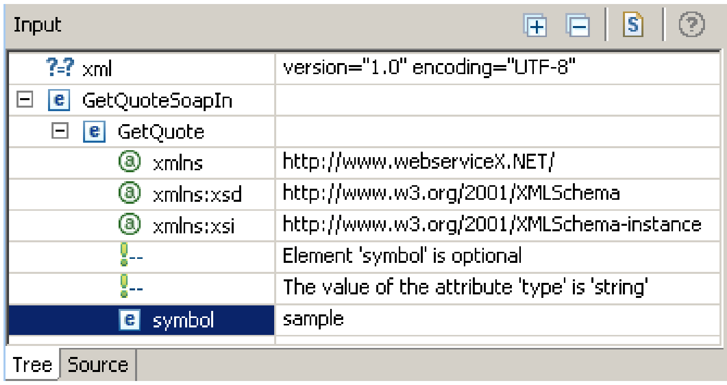
The Integration activity provides a number of XML views (for example, Input and Output messages, WSDL Editor, Messages) derived from the WSDL document. These views use a common interface.

- "Tree View" on page 239
- "Source View" on page 243

## Tree View

You use the Tree view to work with a hierarchical view of an XML document. You display the Tree view by clicking the **Tree** tab.

*Figure 8-2 Tree View*



| | |
|---|---|
| ?-? xml | version="1.0" encoding="UTF-8" |
| ⊟ e GetQuoteSoapIn | |
| ⊟ e GetQuote | |
| ⓐ xmlns | http://www.webserviceX.NET/ |
| ⓐ xmlns:xsd | http://www.w3.org/2001/XMLSchema |
| ⓐ xmlns:xsi | http://www.w3.org/2001/XMLSchema-instance |
| !-- | Element 'symbol' is optional |
| !-- | The value of the attribute 'type' is 'string' |
| e symbol | sample |

- "Tree View Editing Features" on page 240
- "Tree View Menu" on page 240
- "Tree View Toolbar" on page 241
- "Attaching a Schema or DTD" on page 242

## Tree View Editing Features

The Tree view provides the following editing features:

- You can edit attribute values, attribute name, namespace names and values, text, and comments.
- You can insert new nodes by using the menu that is displayed when you right-click within the Tree view. The menu allows you to insert nodes as children before or after the selected node. If the node is an element, you can insert attributes. The submenus for **Add Child**, **Add After**, and **Add Before** contain the node that can be legally added. If no schema or DTD is associated with the document, the submenus contain **New Attribute** or **New Element**.
- You can delete a node by right-clicking a node and selecting **Remove**.
- You can drag and drop items between Tree views (for example, between views of the Input and Output messages) to create Map actions (see "Map" on page 266 for information about Map actions).
- You can undo, redo, cut, copy, and paste.

## Tree View Menu

When you right-click an item in the Tree view, a menu is displayed that you use to perform operations on the XML document. The menu is context-sensitive and only displays the commands that are appropriate for the item you clicked.

*Table 8-1 Tree View Menu*

| Item | Description |
|---|---|
| Remove | Removes the selected item. |
| Add DTD Information | Displays a dialog box that you use to add DTD information. You can edit the **Root Element Name**, **Public ID,** and **System ID**. |
| Edit Namespaces | Displays a dialog box that you use to add namespace declarations. |
| Add Attribute | Displays a dialog box that you use to define a new attribute. |
| Add Child<br>Add Before<br>Add After | Displays a submenu with the following options:<br>Comment<br>Add Processing Instruction<br>#PCDATA<br>CDATA Section<br>New Element |
| Replace with | Replaces the selected item with an item selected from the menu. |

## Tree View Toolbar

Tree view toolbars provide the following features:

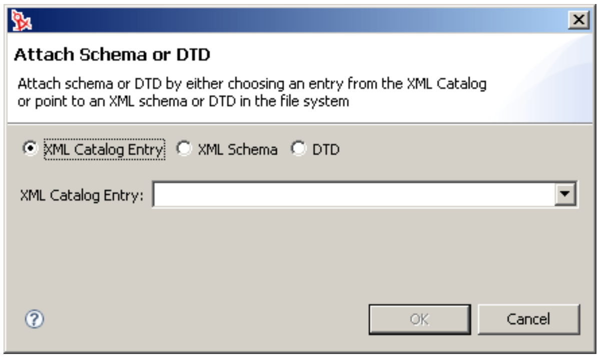
*Table 8-2 Tree View Toolbar*

| Button | Description |
|---|---|
| ⊞ | Expands all nodes in the document. |
| ⊟ | Collapses all nodes in the document. |
| S | Attaches a schema or DTD (see “Attaching a Schema or DTD” on page 242). |
| ? | Displays online help. |

## Attaching a Schema or DTD

You can attach a schema or DTD to the current XML document when you are using the Tree view.

1 Click S in the Tree view toolbar. The **Attach Schemas or DTD** dialog box is displayed.



Attach Schema or DTD

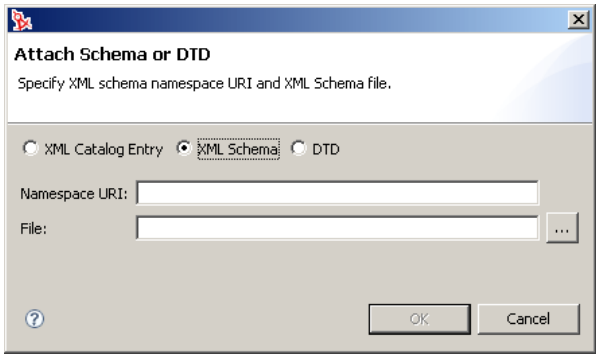
Attach schema or DTD by either choosing an entry from the XML Catalog or point to an XML schema or DTD in the file system

● XML Catalog Entry ○ XML Schema ○ DTD

XML Catalog Entry:

2 To choose from a list of entries in the XML catalog, choose an entry from **XML Catalog Entry** list.

3 To specify an XML schema on disk, click **XML Schema**.



Attach Schema or DTD

Specify XML schema namespace URI and XML Schema file.

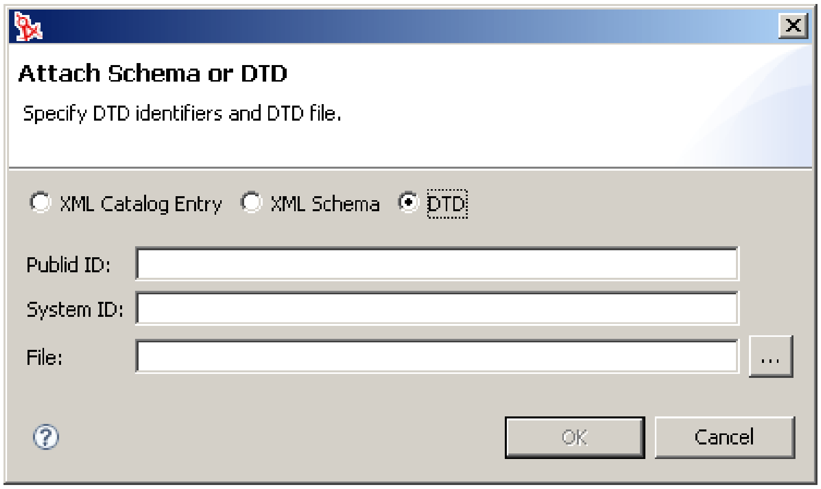
○ XML Catalog Entry ● XML Schema ○ DTD

Namespace URI:

File:

4 Type a **Namespace URI**, then use the browse button in the **File** field to select an XML schema on disk.

5 To specify a DTD on disk, click **DTD**.



Attach Schema or DTD

Specify DTD identifiers and DTD file.

○ XML Catalog Entry ○ XML Schema ● DTD

Publid ID:

System ID:

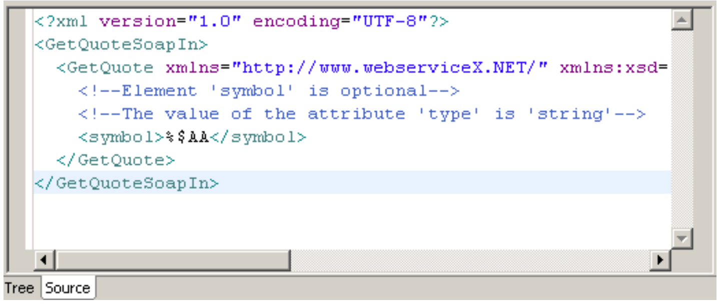
File:

6 Type a **Public ID** and **System ID**, then use the browse button in the **File** field to select the DTD file on disk.

## Source View

You use the Source view to view the XML source of the document. You display the Source view by clicking the **Source** tab.

***Figure 8-3*** *Source View*



```
<?xml version="1.0" encoding="UTF-8"?>
<GetQuoteSoapIn>
  <GetQuote xmlns="http://www.webserviceX.NET/" xmlns:xsd=
    <!--Element 'symbol' is optional-->
    <!--The value of the attribute 'type' is 'string'-->
    <symbol>%$AA</symbol>
  </GetQuote>
</GetQuoteSoapIn>
```

- “Source View Features” on page 243
- “Source View Menu” on page 244

### Source View Features

The source view supports the following features:

- Syntax highlighting.
- Context-sensitive code-completion based on DTD or XML Schema.
- Validation as you type. If the XML is invalid (for example, the closing bracket is omitted from a tag), the editor indicates the error.
- General text editing operations such as undo, redo, cut, copy, paste, and select all.

### Source View Menu

When you right-click an item in the Source view, a menu is displayed that you use to perform operations on the XML document.

*Table 8-3 Source View Menu*

| Item | Description |
|---|---|
| **Undo** | Reverses the last action. |
| **Redo** | Reverses an undo operation. |
| **Cut** | Cuts the selected text to the clipboard. |
| **Copy** | Copies the selected text to the clipboard. |
| **Paste** | Pastes the clipboard contents at the insertion point. |
| **Delete** | Deletes the selected text. |
| **Select All** | Selects all of the text in the document. |
| **Find** | Displays a dialog box that you use to find and replace text within the document. |

# Action Model

The action model includes the Action Model view and views for displaying message parts. The Action Model view displays actions that operate on the contents of the message parts. The message parts display the XML for the Web service Input and Output messages.

## About the Action Model Views

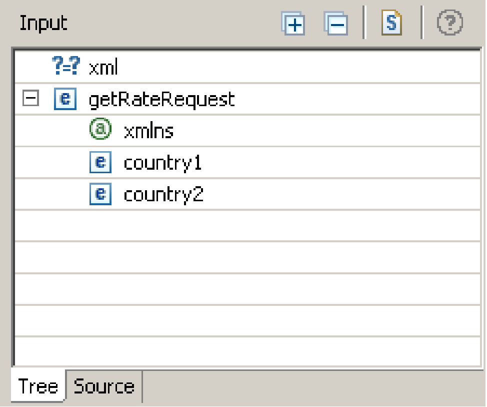
The action model views are used at design time to test the interaction with the Web service. You edit actions in the Action Model view. You can enter test data to be input to the Web service in the Input view, examine the response from the Web service in the Output view, and see any error messages returned from the Web service in the _SystemFault view. The Integration activity has the following message panes:

- Input view
- Output view
- _SystemFault view
- Action Model view

## About the Input View

The Input view displays the input message derived from the WSDL document for the Web service. You can resize the view by dragging the right border.You can resize columns within the view by dragging the column border. You can specify a value to use in testing the action model directly in the Input part, in which case the value is discarded after executing the action model. You can also specify a value using the **Messages** tab (see "Messages" on page 250), in which case the value persists until you delete the value or you regenerate the action model (see "Regenerating Code for the Action Model" on page 251).

*Figure 8-4 Input View*

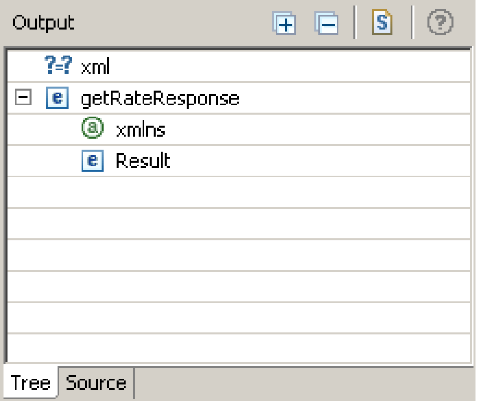


## About the Output View

The Output view displays the output message derived from the WSDL document for the Web service. When you execute the action model, you use the Output view to view the values returned from the Web service.

You can resize the view by dragging the left border.You can resize columns within the view by dragging the column border. You can specify a value directly in the Output part for modeling purposes, in which case the value is discarded after executing the action model. You can also specify a value by using the **Messages** tab (see "Messages" on page 250), in which case the value persists until you delete the value.

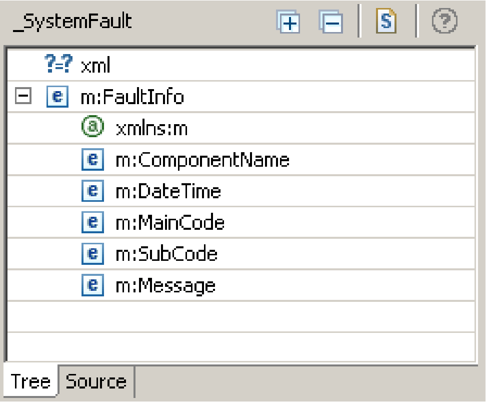
*Figure 8-5 Output View*



## About the _SystemFault View

The _SystemFault view displays any error messages produced when you execute the action model. The XML information contained in _SystemFault is also written to a global object called ERROR.

***Figure 8-6** _SystemFault View*



Beneath the FaultInfo root are the following elements:

- **DateTime** contains the Date and Time at which the fault occurred.
- **ComponentName** contains the name of the component that threw the fault.
- **MainCode** contains the main code number for the error.
- **SubCode** contains a sub-code number for the error.
- Message contains the error message defined when you set up a Throw Fault action (see "Throw Fault" on page 253). If you do not specify an error message in your Throw Fault action, the following message is displayed: `"A user-defined Fault occurred!"`. If the error occurred within a Try/On Fault action, and you did not specify a Fault, this element is populated with an Exception message.

## About the Action Model Pane

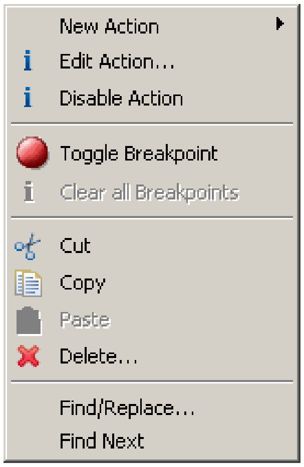
The Integration activity has a single action model. The action model represents the mappings, transformations, and other actions that is performed on the Web service input and output messages. The Action Model view is resizable. Most of your activity that takes place in the Action Model view involves adding and editing actions.

- "Action Model Context Menu" on page 247
- "Finding and Replacing Text in the Action Model" on page 247

## Action Model Context Menu

If you right-click in the action model, a menu is displayed.

***Figure 8-7*** *Action Model Menu*



From this menu, you can add or edit actions (see “Actions” on page 251), toggle breakpoints in the action model (see “Animation” on page 247) and perform other tasks.

### Finding and Replacing Text in the Action Model

You can replace a word or string by using the **Replace** command on the action model menu.

1. Right-click in the action model, then select **Replace**.
2. Specify the search text.
3. If you want to replace the search text, click **Replace with**, then type a string to replace the search string.
4. If you want to find the search text regardless of the capitalization of the text, click **Ignore case**.
5. If you want to find the search text in whole words only, click **Whole word**.
6. Click **OK**

   The Integration activity finds the first occurrence of the search text. If the operation is a find and replace operation, the Integration activity asks you to confirm the replacement. You can then replace the next or all occurrences of the search text.

## Animation

The action model provides animation tools that you can use to test and troubleshoot actions interactively within the Integration activity. You can execute the action model step by step and watch the result of each action. Not only do you see any errors as they happen, but you can verify, in real time, that connections and data behaved as you planned.

The animation tools allow you to toggle one or more breakpoints. You can use this feature to concentrate on a particular section of an action model. When used in conjunction with the run-to-breakpoint tool, breakpoints allow you to quickly run through action model sections that work properly, coming to a stop at a particular action. From there, you can step through each action in sequence. You can also step over loops and other code blocks that would otherwise be tedious to execute step-by-step.

## The Basic Animation Tools

The animation tools are available on the Designer toolbar.

***Figure 8-8*** *Animation Toolbar*

***Table 8-4*** *Animation Tools*

| Animation Toolbar Button | Name | Description |
|---|---|---|
| | **Execute** | Executes the action model. |
| | **Execute Current Action** | Executes the currently selected action. |
| | **Start Animation** | Starts the animation process. Enables **Step Into**, **Step Over**, and **Run to Breakpoint/End**. |
| 🟥 | **End Animation** | Stops the animation process. |
| | **Step Into** | Executes the currently selected action and highlights the next sequential action.<br>For a Repeat Loop action, clicking **Step Into** executes each action in the loop and iterates through each loop.<br>For a Decision Action, **Step Into** processes the next action in the True or False branch.<br>For the Try/On Fault action, **Step Into** processes the next action inside the execute branch, and possibly the On Fault branch. |
| | **Step Over** | Executes the currently selected action and highlights the next sequential action. Unlike the Step Into button, clicking this button does not highlight and execute the details of Repeat, Decision, or Try/On Fault actions. |
| | **Run To Breakpoint/End** | Runs the animation to the next breakpoint or to the end of the action model if there are no breakpoints. |
| ● | **Toggle Breakpoint** | Sets the selected action in the action model as a breakpoint. You may set more than one breakpoint. Another way to toggle a breakpoint is to right-click the desired action and select **Toggle Breakpoint** from the menu. |
| ⏸ | **Pause Animation** | Pauses the animation. |

## Starting Animation

When you first open an Integration activity, **Start Animation** and **Toggle Breakpoint** are the only enabled buttons. When you click **Start Animation**, the rest of the animation buttons are enabled. If you want to halt the animation temporarily, you can use the **Pause Animation** button. If you want to abort the animation, you can do so at any time by clicking **End Animation**.

Although Copy, Paste, and action editing operations (including adding new actions) are all available at animation time, we recommend that you do not edit the action model during animation. If you do, exceptions or unpredictable behavior can occur. If you need to edit the action model, use **End Animation** to stop the animation first. Then apply your edits and begin the animation again.

1. Open an Integration activity.
2. Click **Start Animation** button in the Designer toolbar. All of the animation buttons become active except the Start Animation button, which is now dimmed.
3. Follow the instructions in the following sections to perform the desired Animation activity.

## Toggling a Breakpoint

You use the Toggle Breakpoint tool to set a breakpoint in the action model where you want the animation process to stop. This is helpful if you have a lengthy action model with long sections that work properly. You can set the breakpoints for each action that is causing a problem and then step through the action to troubleshoot it.

1. In the Action pane, select the action for which you want to set a breakpoint. This is where the animation will stop.
2. Click **Toggle Breakpoint** on the Designer tool bar, or right-click the action and select **Toggle Breakpoint**. A dot appears in the left border of the action model to indicate the breakpoint.
3. If desired, repeat the previous steps to select additional breakpoints.

## Stepping Into an Action

**Step Into** runs the selected action in the action model and then moves to the next action in the sequence. You can use the **Step Into** tool to step through each action in the entire action model, or you can use it in conjunction with the Run to Breakpoint tool. Execution stops at the next breakpoint or when the action model ends, whichever comes first.

A possible scenario for using a breakpoint would be if you have ten actions that you know work properly but have doubts about the eleventh. You could set the eleventh action as a breakpoint, execute the Run to Breakpoint tool, and then step through the eleventh (and subsequent) actions by executing the **Step Into** tool.

1. Start the animation (see “Starting Animation” on page 249).
2. Click **Step Into**. The first action in the action model is selected.
3. Click **Step Into** again. The selected action executes and the next action is selected.
4. Continue to work through the action model by clicking **Step Into** after each action executes and the subsequent action is selected.

### Stepping Over an Action

You use **Step Over** when you don't want to step into the details of the Repeat, Decision, or Try/On Fault actions. You can execute an entire block of code without stepping individually through each action.

You can use **Step Over** in conjunction with Run to Breakpoint. For example, you can toggle a Breakpoint, execute Run to Breakpoint, and then use **Step Over** to execute the action designated as the breakpoint. **Step Over** can save a great deal of time when testing lengthy action models, because you can avoid tediously stepping through individual actions.

1. Start the animation (see "Starting Animation" on page 249).
2. Step through the action model with **Step Into** until you reach a loop or other line of code that precedes an indented code block.
3. Click **Step Over**. The first action after the block of indented code is selected. All of the indented code executes normally and you are taken straight to the next block of actions, without stepping through the indented actions.
4. Continue to work through the action model by clicking **Step Over** as needed.
5. Continue to click **Step Into** and **StepOver** to execute all of the actions in the action model.

### Pausing Animation

You use **Pause Animation** to pause the execution of an action in the action model. This is especially helpful in cases in which the action model contains lengthy loops.

1. During the execution of an action, click **Pause Animation**.
2. To resume the animation process, click **Step Into**, **Step Over**, or **Run to Breakpoint** (if a breakpoint has been set).

### Aborting Animation

You use **Stop Animation** to stop the animation process. After you stop the animation, you cannot restart from the place where you left off. You must restart from the beginning of the action model.

## WSDL Editor

The WSDL Editor displays the WSDL document for the Web service. You can edit the WSDL by using the Tree view and Source view editing features (see "Tree View" on page 239and "Source View" on page 243).

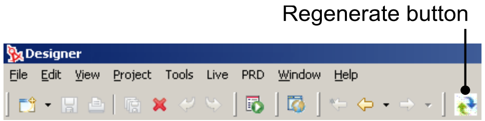
## Messages

The Messages view displays the messages derived from the WSDL document for the Web service. You can edit the messages using the Tree view and Source view editing features (see "Tree View" on page 239 and "Source View" on page 243). You can use this feature for entering test data that is used when you execute the action model at design time. Data that you enter in the Messages view persists across executions of the action model.

## Regenerating Code for the Action Model

When working in the WSDL Editor view, you can regenerate all code for the action model and regenerate messages by clicking the **Regenerate** button. When working in the Messages view, you can regenerate all actions in the action model. The **Regenerate** button is located in the Designer toolbar:

***Figure 8-9*** *Regenerate Button*



## Adding Actions to the Action Model

Actions are the processing steps that take place within the Integration activity. A collection of actions is referred to as an action model. An action in the action model is displayed as a line and contains an icon for the action type along with an abbreviated definition of the action. Some actions are subordinate to other actions. For example, you can create a Repeat action that controls loop processing, then add actions inside the loop. The actions inside the loop are subordinate to the Repeat action and appear indented beneath the Repeat action. Subordinate actions process as long as the Repeat action is True.

To add an action to the action model, click the line in the action model that is one line above the position in which you want to insert an action. Add an action by using any of following methods. The new action is inserted below the line you selected.

- Drag and drop. You can add Map actions by dragging and dropping elements from the **Input** view to the **Output** view.
- Copy and Paste. You can copy an action in the **Action Model** view and paste it somewhere else in the view.
- Right-click the line in the action model that is one line above the position in which you want to add the action, then select the desired action from the menu.

**NOTE:** You can reorder actions in the action model by dragging actions to a new position within the action model.

After you have created the action model, you should test the action model. Perform testing by using the Animation tools. With the Animation tools, you can set breakpoints, start an animation, step into and over actions, and pause the animation. See “Animation” on page 247.

# Actions

This section describes the actions that are available for use within an action model. An action is similar to a programming statement in that it takes input in the form of parameters and performs specific tasks. An action model is a set of instructions for processing XML documents and communicating with XML data sources. An action model performs all data mapping, data transformation, and data transfer within an Integration activity. All actions within an action model work together.

At runtime, every action is converted to an executable form of ECMAScript and processed. At design time, many actions accept ECMAScript expressions as parameters, adding great flexibility and control to the action model. The Function action provides you with the most flexibility and control by giving you access to the full functionality of the ECMAScript language.

# Advanced

This section includes descriptions of the following actions:

- "Apply Namespaces" on page 252
- "Throw Fault" on page 253
- "Try/On Fault" on page 255

## Apply Namespaces

### About the Apply Namespaces Action

Ideally, an Integration activity always receives valid XML documents (that is, the documents validate against their schema, map and transform data appropriately, and send valid XML documents). However, this might not always the case. In other cases you might want to ignore namespaces altogether. It is important to have some means of validating XML documents. These and many other XML processing cases require a method of modifying or overriding the prefix and namespace handling provided by the Integration activity.

The Apply Namespaces action provides a mechanism for managing namespaces and namespace prefixes in effect for input and output messages within an action model. The action allows you to consolidate namespace and prefix declarations for a Web service in one place, to override those declared in the input and output messages, or to ignore namespaces altogether.

The Apply Namespaces action can be applied to input and output messages. You can also have multiple Apply Namespaces actions for an individual message part, effectively changing namespaces based on conditions specified in the action model. The namespaces declared are in effect until the end of the action model is reached or another Apply Namespaces action is executed. In other words, only the most recent Apply Namespaces action is in effect.

When creating a new Integration activity, an Apply Namespaces action is created automatically for the Output message if the WSDL declares any namespaces. After component creation, you can manually create additional Apply Namespaces actions.

### Creating an Apply Namespaces Action

1 Right-click the line in the action model that is one line above the position in which you want to place the Apply Namespaces action (the new action is inserted below the line that you selected).

2 Select **New Action > Advanced > Apply Namespaces**.

3 Select the message (Input, Output, _SystemFault, or Project) to which you want to apply the namespace from the **Apply the following namespaces for Part** list.

4. Click the plus (+) icon to add a new row, then click the **Namespace** column and type a namespace URI.

   The table displays all the Namespace declarations for the selected message part. After creating a new Apply Namespaces action, the table might or might not contain a list of declarations for a selected part. The list of declarations is initially constructed from the declarations defined in the WSDL.

   Within the declaration list for a message part, prefixes must be unique. However, you can have duplicate namespace URIs if the URIs have unique prefixes. This allows you to declare multiple prefixes that are associated with the same namespace URI.

5. If desired, click **Ignore Namespaces when document is used in an XPath expression.**

   Use this option when you want Map action source XPaths to find elements by their XML local name only.

   This provides for a less restrictive method of specifying Map actions (see “Map” on page 266) and is useful when Map actions contain the wrong prefix or no prefix in their Source specifications. This allows you to place the Apply Namespaces action inside a Decision action (see “Decision” on page 264) that tests whether the Input message contains prefixes or not, yet still have one set of Map actions to map the input to another part. In other words, the Integration activity normally expects the input to contain prefixes, so you design all your Map actions with prefix names. For the occasional input that has no prefixes, the Decision action activates the Apply Namespaces action defined to ignore namespaces for input, allowing the Map actions to work in either case.

6. When you want to declare a set of namespaces in the root element of an output message built by your action model, click **Declare these namespaces in the part**.

   This option is almost always checked for output to ensure that prefixed elements created in the output, as a result of Map actions, resolves to the proper namespaces.

   This allows a recipient of the output to validate the document properly. The Apply Namespaces action with this option checked could also be used to add new declarations to an existing document that already contains declarations.

   The **Target document Root Element Name** is used to specify the name of the root element to contain the Namespace declaration attributes. The Integration Activity automatically fills in this value based on the information in the WSDL document and the message part specified in the **Apply the following namespaces** for Part list.

7. Click **OK**. The new action is added to the action model.

# Throw Fault

## About the Throw Fault Action

You use the Throw Fault action to do the following:

- Write information to an XML message on failure of an action
- Perform any number of actions before throwing the fault
- Halt execution of a component

Throw Fault is only executed when a condition that you specify is true. The message part that is written when a Throw Fault action is executed is called a Fault document, and the XML within this message is contained in a global object called ERROR.

Throw Fault actions can be used in a number of ways:

- Using a Throw Fault Action by itself. You can specify a Fault Condition and an error message within the Throw Fault Action dialog. When the action is executed, the Fault Condition is evaluated. If the condition evaluates as True, the following occurs:
  - Any Before Throw actions that you specify are executed. This can be very useful as a way to leave your application in a particular state before halting execution. For example, you can send a mail message stating that the execution did not complete.
  - The contents of the error message are written to the Fault document in a node that you specify, as well as to the global object ERROR.
  - The action model execution is halted.
- Using a Throw Fault Action within a decision expression in the Decision action. You can specify a fault condition by entering it in the decision expression of a Decision action. Then put a Throw Fault statement in the True branch of the Decision action. Here you can either specify additional conditions in the Throw Fault fault condition or leave it blank and simply specify the fault document to which the fault information should be written. When the action is executed and all your conditions are True, the Throw Fault action is executed. If the fault condition in the Decision action or Throw Fault action is False, the next action in the action model is executed.
- Using a Throw Fault inside a Try /On Fault action (see "Try/On Fault" on page 255. By putting either of the above methods inside the execute branch of a Try /On Fault action, you prevent the Integration activity from halting execution and have an opportunity to respond or recover from the fault. You create your fault condition by using one of the previous two methods inside the execute branch of a Try/On Fault action after other actions the output of which you want to test have worked correctly. You can specify any number of unique faults so that the Integration activity can branch into several different directions, depending on which fault occurs. When the Throw Fault action for the given fault is triggered, instead of halting execution of the component, control passes into the appropriate branch of the Try/On Fault action. Here you can specify other actions to remedy or respond to the error.

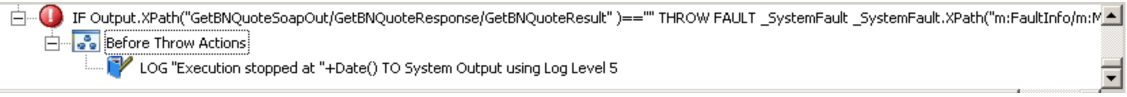
## Adding a Throw Fault Action

1. Right-click the line in the action model that is one line above the position in which you want to place the Throw Fault action (the new action is inserted below the line that you selected).
2. Select **New Action > Advanced > Throw Fault**.
3. In the **Fault Condition** field, type a valid ECMAScript expression that, when True, causes the action to throw a fault.

   You can also click the ECMA Expression Builder button and build an expression (see Chapter 9, "Working with ECMA Expressions," on page 271).
4. Select **Throw {System}{Fault}** to write your error message to the _SystemFault document.

   By default, the message that you type in the **Error Message** field is placed in the Fault/FaultInfo/Message node of that document. Specify a different node if desired. You can also click the ECMA Expression Builder button and build an expression.
5. Select **Throw Defined Fault** to select a fault document that is one of the message parts associated with the Integration activity.
6. Click **OK**.

   The new action is added to the action model. Place any actions that you wish to execute before the application stops in the **Before Throw Actions** branch.



```
IF Output.XPath("GetBNQuoteSoapOut/GetBNQuoteResponse/GetBNQuoteResult" )=="" THROW FAULT _SystemFault _SystemFault.XPath("m:FaultInfo/m:M
    Before Throw Actions
        LOG "Execution stopped at "+Date() TO System Output using Log Level 5
```

# Try/On Fault

## About the Try/On Fault Action

The Try/On Fault action executes a set of actions when a fault occurs within the Execute branch of the Try/On Fault action. Any number of defined faults can be specified within the Execute branch. You can use the Try/On Fault action to trap anticipated errors and run other actions to remedy or report on the fault. For instance, you can use Try/On Fault to respond to an XML Interchange action that fails to find a file.

When you add a Try/On Fault action, several lines are added to the action model:

- The beginning of the Try action
- The **Execute** branch
- A branch for each Fault that you specified
- An **All other Faults** branch

When you are aware of potential faults an action can produce, you put those actions in the Execute branch. You then put error handling actions under each On Fault branch to handle unique situations. If a fault does occur, the actions in the On Fault branch execute.

Using the example given previously, if you anticipate a fault with the XML Interchange action, you put the action under the Execute branch. In one On Fault branch, you might add another XML Interchange action that attempts to read the file from an alternate location. In another On Fault branch, you might add another XML Interchange action that looks for a file with a different extension.

## Adding a Try/On Fault Action

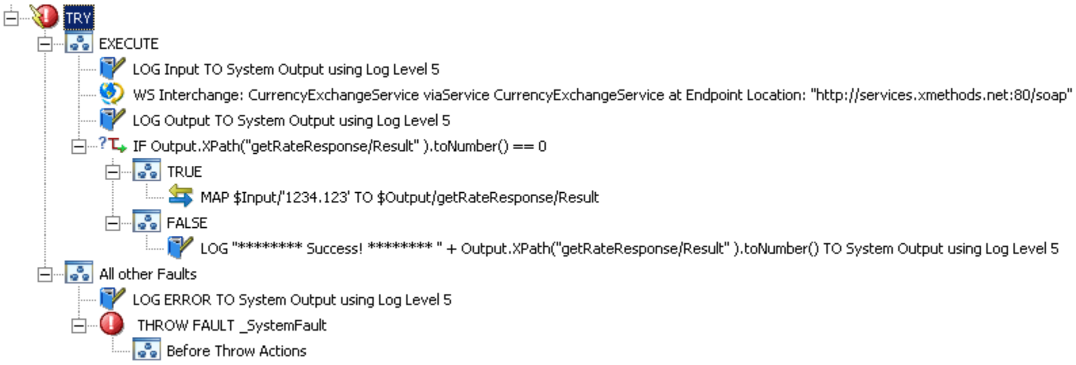
1. Right-click the line in the action model that is one line above the position in which you want to place the Try/On Fault action (the new action is inserted below the line that you selected).
2. Select **New Action > Advanced > Try/On Fault**.

   Use the + icon to add a new fault part to the **Fault Part Name** list. Use the red - icon to remove fault parts from the list. Use the up-arrow and down-arrow icons to change the order of the faults.

   If you don’t specify a fault part, corrective actions can be placed in the default All Other Faults branch of the Try/On Fault action.
3. Click **OK**.

   The following appears in the Action Model Viewer: the **Try On Fault** action icon, with an Execute, one or more On Fault branches, and an All Other Faults branch.
4. Add any actions that might cause errors to the Execute branch.
5. In the On Fault branch, add actions that resolve the errors specified in the Execute branch.

The following illustration shows a complete Try/On Fault action in the action model.



```
TRY
  EXECUTE
    LOG Input TO System Output using Log Level 5
    WS Interchange: CurrencyExchangeService viaService CurrencyExchangeService at Endpoint Location: "http://services.xmethods.net:80/soap"
    LOG Output TO System Output using Log Level 5
    IF Output.XPath("getRateResponse/Result" ).toNumber() == 0
      TRUE
        MAP $Input/'1234.123' TO $Output/getRateResponse/Result
      FALSE
        LOG "********* Success! ********* " + Output.XPath("getRateResponse/Result" ).toNumber() TO System Output using Log Level 5
  All other Faults
    LOG ERROR TO System Output using Log Level 5
    THROW FAULT _SystemFault
      Before Throw Actions
```

# Data Exchange

This section includes descriptions of the following actions:

- "WS Interchange" on page 256
- "XML Interchange" on page 257

## WS Interchange

### About the WS Interchange Action

The WS (Web Service) Interchange action is the most important action in the Integration activity and allows the Integration activity to invoke a Web service according to calling conventions specified in a WSDL file. The Integration activity automatically creates a WS Interchange action when it creates the action model.

In most cases there is no need to add another WS Interchange action to the action model. However, there might be situations in which you need to do so. The following procedure describes how to add a WS Interchange action.

### Adding a WS Interchange Action

1. Right-click the line in the action model that is one line above the position in which you want to place the WS Interchange action (the new action is inserted below the line that you selected).
2. Select **New Action > Data Exchange > WS Interchange**.

   The **WSDL Resource**, **Service Name**, **Port**, and **Operation** fields are filled in automatically based on the information in the WSDL specified for the Integration activity.
3. If desired, modify the information in the **Endpoint Location** field (usually a URL pointing at a servlet) for the Web service that you wish to use, wrapped in quotation marks. Alternatively, enter an ECMAScript expression that will evaluate to an Endpoint Location at runtime.
4. Click the **Messages** tab.

   The **Message**, **Part**, and **Type/Element** fields are filled in automatically based on the information in the WSDL specified for the Integration activity.
5. If desired, click the **Expression** column for a message, then use the ECMA Expression Builder to create an ECMAScript expression that describes the source and target for the message. Usually, this is an expression that specifies an XPath location in an Input part or Output part.

6 Click the **Connection** tab.

   You use this tab to specify connection parameters for HTTP servers that require authentication.

7 Type a user ID to use for the connection in the **User ID** field, and a password for the user in the **Password** field.

   The user ID and password are not actually submitted during the establishment of a connection. They are simply defined here. The password is encrypted. You will have access to UserID and Password variables in ECMAScript, allowing you to map the user ID and password as values into the screen. This way, no one ever sees the passwords.

8 If the connection requires a client certificate, choose a client certificate by clicking the browse button in the **Client Certificate** field and selecting the certificate file you want to use for this connection.

9 If the connection requires a client private key, choose a client private key by clicking the browse button in the **Client Private Key** field and selecting the client private key file.

10 Type the password for the client private key in the **Private Key Password** field.

11 Specify a connection timeout value, in seconds, in the **Connection Timeout** field.

12 Click **Apply** to test the WS Interchange action in real time, or click **OK** to close.

# XML Interchange

## About the XML Interchange Action

The XML Interchange action reads external XML documents into a DOM and writes data from the DOM as XML files. There are four types of XML Interchange actions:

- GET
- PUT
- POST
- POST with Response

When using the GET interchange, fill in the **Interchange URL Expression** field with a URL that points to the XML document that you want to bring into the Integration activity. In the **Response Part** field, you select the DOM (Input, Output, _SystemFault, or Project) that is to receive the XML.

When using the PUT interchange, enter a URL that points to the location to which you want to write the XML document in the **Interchange URL Expression**. In the **Request Part** field, you select the name of the DOM from which you want to send data as XML.

When using the POST interchange, enter a URL that points to the location to which you want to write the XML document in the **Interchange URL Expression** field. In the **Request Part** field, you select the name of the DOM from which you want to send data as XML.

When using the POST with Response interchange, you supply the same parameters as for Post, with one additional parameter. You must also specify a Response Part DOM to receive the Response XML document from the Post with Response interchange. The difference between the two interchanges is that Post with Response expects a response XML object back from the origin server.

## Adding an XML Interchange Action

1. Right-click the line in the action model that is one line above the position in which you want to place the XML Interchange action (the new action is inserted below the line that you selected).
2. Select **New Action > Data Exchange > XML Interchange**.
3. Select an **Interchange Type** (Get, Put, Post, or Post with Response).
4. In the **Interchange URL Expression** field, type an expression that defines a fully qualified URL for an XML document, using any of the following supported protocols:

   - file
   - FTP
   - HTTP
   - HTTPS

   Depending on the **Interchange Type** selected, this URL is the source or the destination of the XML file for the XML Interchange action. For example:

   ```
   file:///g:/xmldata/invoicebatch1.xml
   ftp://accounting:password@123.456.789.987:21/invoices/inv1.xml
   ```

   Because this is an ECMAScript expression, the URL string must be enclosed in quotation marks.
5. If you need to specify HTTP header parameters, click **HTTP Header Parameters**.
6. Click the plus (+) icon to add new header parameters, then type a **Parameter** name and a corresponding **Value**. Common HTTP header parameters include “Content-Type,” “Content-Length,” and “Keep-Alive.” You can add any number of Parameter-Value pairs.
7. Click **OK** to return to the **XML Interchange** dialog box.
8. In the **Request Part** field (which is enabled if the **Interchange Type** is Put, Post, or Post with Response), select the name of the DOM from which you want to send data as XML.
9. In the **Response Part** field (which is enabled if the **Interchange Type** is Get or Post with Response), select the name of the DOM tree that will receive the XML.
10. If you want to filter the incoming XML document to improve performance, select the check box next to the **Filter Document** button, then select the **Filter Document** button.

    The document displayed is the document selected in the **Response Part** in the **XML Interchange** dialog box.

    You use this dialog box to specify the individual nodes to retain (rather than discard) from the incoming XML document in real time to improve performance and reduce RAM overhead.
11. Select the nodes that you want to keep in the document.

    Nodes that are not selected are discarded before parsing the DOM.
12. When you have selected nodes that you want to keep, click **OK** to return to the **XML Interchange** dialog box.
13. Click **OK**.

    Alternatively, you can click **Apply** to see the affect of the XML Interchange action without closing the dialog box. This allows you to make repetitive edits to an XML Interchange action and quickly see the results.

# Repeat

This section includes descriptions of the following actions:

- "Break" on page 259
- "Continue" on page 259
- "Declare Group" on page 260
- "Repeat for Element" on page 261
- "Repeat for Group" on page 262
- "Repeat While" on page 263

## Break

### About the Break Action

The Break Action stops the execution of a Repeat for Element, Repeat for Group, or Repeat While loop. The action model continues execution with the next action outside the loop.

The use of Break is appropriate when, for example, you are using a loop to search a node list for one particular item. When the target item is found, there is no need to continue iterating. A Break can be used to terminate the loop immediately.

**NOTE:** A Break action is usually used in one branch of a Decision action (within a loop). You place the Break action in either the True or False branch of the Decision action.

### Adding a Break Action

1. Within a Repeat action that you want to modify to include a Break action, select a position inside the loop where you want to place the Break action.

   Generally, this is in one leg or the other of a Decision action.

2. Select **New Action > Repeat > Break**.

   The Break action is inserted into the action model.

## Continue

### About the Continue Action

The Continue action causes execution of the current iteration of a Repeat for Element, Repeat for Group, or Repeat While loop to stop and execution to begin at the top of the loop, with the next iteration. The Continue action provides a way to pass over downstream actions inside the loop while allowing the loop to continue on to the next iteration.

A Continue action is appropriate in a situation where, for example, one item in a list should be skipped for some reason, yet execution of the loop must continue.

**NOTE:** A Continue action usually occurs in one branch of a Decision action within a loop. You place the Continue action in either the True or False branch of the Decision action, as appropriate.

### Adding a Continue Action

1. Within a Repeat action that you want to modify to include a Continue action, select a position inside the Loop actions where you want to place the Continue action.

   This is usually inside one fork or the other of a Decision action.

2. From the Action menu, select **New Action > Repeat > Continue**.

   A Continue action appears in the action model.

## Declare Group

### About the Declare Group Action

You use the Declare Group action to create two special lists, each in reference to a DOM. These group lists can then be used as the basis for a loop in the Repeat for Group action. To create the lists, you supply a Group Name and specify an XPath. The Integration activity then creates the lists as follows:

- A Group list is created that contains one entry for each unique value found in all the elements that match the XPath. The Group list is referred to by the Group Name that you supply.
- A Detail list is created for each unique entry in the Group list that contains as many entries as there are members in the Group. The Detail list is referred to by the group name that you supply, post-fixed with the label (Detail).

Using Groups allows you to select a repeating element in your Input DOM and create fewer elements based on the unique values across all siblings of that repeating element. Instead of having multiple elements, you have one element for each unique element value in your Output DOM.

### Adding a Declare Group Action

1. Right-click the line in the action model that is one line above the position in which you want to place the Declare Group action (the new action is inserted below the line that you selected).
2. Select **New Action > Repeat > Declare Group**.
3. Type a name for the group in the **Group Name** field.
4. If you want to create multiple group levels, select a group from the **Parent Group** list, which lists groups that you have previously defined.
5. Click Add. The **Add Element** dialog box is displayed.
6. Select a part name and an element.
7. Click **OK**.
8. Repeat Step 5 through Step 7 to add more elements to the group.
9. When you have all the elements that you want in the group, click **OK**.

# Repeat for Element

## About the Repeat for Element Action

The Repeat action creates looping structures within an action model. Loops give you the ability to repeat a set of one or more actions. There are three types of loops: Repeat for Element, Repeat for Group, and Repeat While.

XML allows multiple instances of an element in a document (analogous to multiple records in a database table). The number of instances can vary from document to document and is defined in the document schema (DTD or XML Schema). For example, you might receive an XML document containing line items for an invoice on a daily basis. Each day the XML document has a different number of line items. Not knowing how many instances of “line item” are in the XML document poses a problem if you want to transfer these item numbers from the input XML document to an output XML document programatically. The Repeat for Element action solves this problem.

The Repeat for Element action allows you to mark an element that occurs multiple times. The action then sets up a processing loop that executes one or more actions for each instance of the marked element until no more instances exist. In the previous example, the processing loop would contain a single Map action to transfer the line item number and this action would be repeated until all line items had been mapped.

The Repeat for Element action also uses the concept of an alias. An alias performs two functions. It is an alternate name or shorthand for the marked repeating element, which saves you the work of re-specifying long XPath expressions. In some cases, the repeating element might be several levels down in the document hierarchy. When you create Map actions in the Repeat loop that transfers child elements of the marked element, using the alias is quicker than re-typing a long XPath expression. An alias is also an indicator to Map actions within the Repeat loop to use the next instance of the repeating element each time the loop processes. A Map action within a Repeat for Element loop that does not use the alias always refers to the first instance of the element in the source message.

The Repeat for Element action allows you to process more than one action within the loop. In the simplest case, the repeat loop might only contain one Map action that transfers the value of the current element instance from the input Part to the output Part. You can also define multiple actions in the processing loop. For example, a Map action to transfer the current value, and a Log action that writes to a file, creating an audit of each transfer.

## Adding a Repeat for Element Action

**1** Right-click the line in the action model that is one line above the position in which you want to place the Repeat for Element action (the new action is inserted below the line that you selected).

**2** Select New **Action > Repeat > Repeat for Element**.

**3** Specify the **Source** information.

**3a** Type an alias name in the Source **Alias** field.

A good naming convention for an alias is to use the element name with a prefix indicating source or target and the type of repeat action, such as “S1Lineitem.”

**3b** Type an XPath expression, or click the ECMA Expression Builder button and build an XPath expression for the repeating element.

4 Specify the Target information.

   4a Type an alias name in the Target **Alias** field.

   4b Select **Always create new output elements** if you have repeating actions that should add new elements rather than updating existing elements.

   4c Specify an XPath expression, or click the ECMA Expression Builder button and build an XPath expression for the repeating element.

5 Click **OK**. The Repeat for Element loop is added to the action model.

6 Click **Loop Actions** in the action model to begin adding actions to be performed within the loop.

## Repeat for Group

### About the Repeat for Group Action

The format of an XML document that you receive is not always the format that meets the requirements of your business process. A Repeat for Group action allows you to restructure data and establish a framework to calculate aggregates on your data. Grouping allows you to select a repeating element in your input part and create fewer elements based on the unique values across all siblings of that repeating element.

The Repeat for Group action sets up a processing loop based on one of two lists created by the Declare Group action. The loop executes as many times as there are entries in the list you use (either the Group list or Detail list). By combining a Repeat for Group with Map commands, you can create a new XML document that has a different structure and data from the original.

Similar to the Repeat for Element action, a Repeat for Group action also uses the concept of an alias. The values for the source group used in the Repeat for Group dialog boxes are the list names created by the Declare Group action. The list names perform two functions. They are an alternate name or shorthand for the XPath source of any Map actions within the loop. This saves you the work of re-specifying long XPath expressions. The group list name, when used in place of a DOM name in a Map action source, is also an indicator to the Map action within the Repeat loop to use the next instance in the group list each time the loop processes. A Map action within a Repeat for Group loop that does not use the group name always refers to the first instance of the element in the source part.

The target aliases created in the Repeat for Group action also serve two functions. They are an alternate name or shorthand for the XPath target of any Map actions within the loop. This saves you the work of re-specifying long XPath expressions. The target alias, when used in place of a part name, is also an indicator to Map actions within the Repeat loop to create a new instance of the Source in the target message part. A Map action within a Repeat for Group loop that does not use a target alias always overwrites the first instance created in the target message part with subsequent instances from the source group list.

Creating a Repeat for Group action consists of three tasks:

- Create a Declare Group action to create the group lists.
- Create a Repeat for Group action specifying which group list to use.
- Create Map actions inside the loop.

### Adding a Repeat for Group Action

1 Right-click the line in the action model that is one line above the position in which you want to place the Repeat for Group action (the new action is inserted below the line that you selected).

2 Select **Action > New Action > Repeat > Repeat for Group**.

3 In the Source section, select a Group name from the **Where** list on which to base the Repeat for Group action loop.

4 Optionally, type a Where clause in the **Where** field to filter the group list, or click the ECMA Expression Builder icon and create a Where expression.

5 If you know the alias for the Target element, type the name in the **Alias** field.

6 If you do not know the alias, select either the **XPath** button and select an element from the list, or select the **Expression** button and type an expression (or click the ECMA Expression Builder button and build an expression).

7 Click **OK**.

# Repeat While

## About the Repeat While Action

The Repeat While action repeats one or more actions as long as a condition that you specify remains True. The target alias that you create in the Repeat While action serves two functions. It is an alternate name or shorthand for the XPath target of any Map actions within the loop. This saves you the work of respecifying long XPath expressions. The target alias, when used in place of a DOM name in a Map action, is also an indicator to Map actions within the Repeat loop to create a new instance of the source in the target DOM. A Map action within a Repeat for Group loop that does not use a target alias always overwrites the first instance created in the target DOM with subsequent instances from the source.

**NOTE:** Unlike the Repeat for Element and Repeat for Group, Repeat While does not need to be based on data in a DOM tree. The loop can operate independently of data in the DOM tree.

## Adding a Repeat While Action

1 Right-click the line in the action model that is one line above the position in which you want to place the Repeat While action (the new action is inserted below the line that you selected).

2 Select **New Action > Repeat > Repeat While**.

3 In the **While** field, type an expression that tests the While loop, or click the ECMA Expression Builder button and build an expression.

4 In the **Index Variable** field, type a name for a variable to keep track of the condition of the loop.

5 If you know the alias for the Target element, type the name in the **Alias** field.

6 If you do not know the alias, select either the **XPath** button and select an element from the list, or select the **Expression** button and type an expression (or click the ECMA Expression Builder button and build an expression).

7 Click **OK**.

# Comment

## About the Comment Action

You can use the Comment action to document the action model and clarify the processing that takes place. You can add comments anywhere within an action model. Comments perform no processing of their own.

## Adding a Comment Action

1. Right-click the line in the action model that is one line above the position in which you want to place the comment (the new action is inserted below the line that you selected).
2. Select **New Action > Comment**.
3. Type your comment.
4. Click **OK**.

# Decision

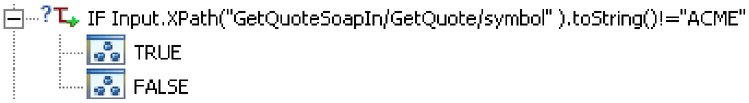
## About the Decision Action

The Decision action creates an *if. . . then* construct between actions or a group of actions. You use a Decision action to select a branch, based on a condition that you supply. The condition must use an ECMAScript comparison operator, such as = =, <, >, !, >=, <=, (&), OR (||), or <>. The expression must resolve to a Boolean True or False statement.

## Adding a Decision Action

1. Right-click the line in the action model that is one line above the position in which you want to place the Decision action (the new action is inserted below the line that you selected).
2. Select **New Action > Decision**.
3. Type an ECMAScript expression, or click the ECMA Expression Builder button and create a Decision script that will evaluate to *true* or *false* at runtime.
4. Click **OK**.

   A Decision action similar to the following is displayed.



```
IF Input.XPath("GetQuoteSoapIn/GetQuote/symbol" ).toString()!="ACME"
    TRUE
    FALSE
```

5. In the action model pane, select the **TRUE** branch.
6. Add one or more actions that will execute if the expression is True.
7. Select the **FALSE** branch.
8. Add one or more actions to execute if the expression is False.

   You can nest other Decision actions inside the TRUE or FALSE branches of the Decision action.

# Function

- "About the Function Action" on page 265
- "Adding a Function Action" on page 265

## About the Function Action

The Function action executes an ECMAScript function. To manipulate a DOM element, the script that you call in the Function action must reference a fully qualified DOM element name in the current Integration activity.

Custom Script functions that you create and add to an action model can act upon any XML tree element. For example, you can create a function that changes the format of a date element. You can create a function that performs a math function on the contents of an element. You can also perform file system, database, or URL functions that have no interaction with a message part.

## Adding a Function Action

1. Right-click the line in the action model that is one line above the position in which you want to place the Function action (the new action is inserted below the line that you selected).
2. Select **New Action > Function**.
3. Type the function in the **Function Expression** field or click the ECMA Expression Builder button to build an ECMAScript expression.
4. Click **OK**.

# Log

- "About the Log Action" on page 265
- "Adding a Log Action" on page 265

## About the Log Action

Log actions provide customizable reporting capabilities (design-time as well as runtime) for Integration activities. You can specify log level settings to control the degree of reporting.

Some Log usage examples include the following:

- Writing certain error information to the operator console when a Try On Fault condition is reached.
- Using ECMAScript expressions to aid in debugging by logging information about variables or DOM contents, the values of which are known only at runtime.
- Capturing specific information from each cycle of a Repeat for Element loop.

## Adding a Log Action

1. Right-click the line in the action model that is one line above the position in which you want to place the Log action (the new action is inserted below the line that you selected).
2. Select **New Action > Log**.

3 Select the type of log that you want to produce from the **Log to** group.

The Log action writes information to locations specified in the action. There are three locations for log output: System Output, System Log, and User Log.

| Log Location | Description |
|---|---|
| **System Output** | Select System Output to write messages that you specify in the **Log Expression** field to the operating system console at design time, or the application server console at runtime.<br>**NOTE:** To view messages on the operating system console, start Designer using the Eclipse `-debug` and `-consoleLog` startup parameters. |
| **System Log** | Select System Log to write messages that you specify in the **Log Expression** field to the application server log file. |
| **User Log** | Select User Log to write messages that you specify in the Log Expression field to a file that you specify in the **User Log File** field. |

4 Use **Log Level** to select a priority level (1 to 10) for this Log action.

The default priority level is 5. You should assign a number from 5 to 10 to messages that you want to appear in the log file. The priority you assign here is compared to the threshold number (which is set to 5 internally and cannot be changed). If the priority is equal to or greater than the threshold, the message is logged; otherwise it is not.

5 Check **Clear the Log File** if you want the data in the log file to be cleared each time the component is executed.

6 If **User Log** is selected in the **Log to** group, type the path to the log file in the **User Log File** field, or use **Browse** to specify a log file.

If you specify a file that doesn't exist, the file is created. On Windows* systems, if you type the path, you must add an extra backslash character wherever a backslash character occurs in a path (for example, C:\Windows becomes C:\\Windows).

7 Create the message that you want to record to the log in the **Log Expression** field.

You can type a message in the field or use the ECMA Expression Builder to build an expression.

# Map

## About the Map Action

The Map action performs DOM-node input and output mapping. It can transfer and transform data from one document context to another document context. A context has two parts. The first part usually identifies a DOM and the second part identifies a location within the DOM. The basic document context in an Integration activity is expressed as a DOM name combined with an element location identified through an XPath expression. The DOM name is usually Input, Output, _System Fault, or Project. The XPath expression identifying a location in a DOM has the path elements delimited by “/”.

**NOTE:** A context in an Integration activity can also be a Group name that itself is simply an alias or shorthand for an XPath expression.

- “Default Mapping Behavior” on page 267
- “Leaf Elements That Contain Markup” on page 267

## Default Mapping Behavior

When you use the Map action to map elements and attributes within XML documents, certain default behaviors occur. The following table lists those default behaviors.

***Table 8-5*** *Default Mapping Behavior*

| Map Type | Default Behavior |
|---|---|
| Leaf Element to Leaf Element | Transfers only the element data. |
| Leaf Element to Branch Element | Transfers only the element data. |
| Branch Element to Leaf Element | Transfers the entire branch, including all child elements and attribute data under the branch. |
| Branch Element to Branch Element | Transfers the entire branch, including all child elements and attribute data under the branch after removing the target’s current branch. |
| A particular Leaf Element in a list of Leaf Elements, to Element | Transfers the element data from the selected leaf (or element instance) to the target element. |
| Attribute to Attribute | Transfers only the attribute data. |
| Element to Attribute | Transfers element data to attribute data. |
| Attribute to Element | Transfers only the attribute data. |

Many of these behaviors can be altered, on an action-by-action basis, through the use of the Advanced mapping dialog box (see “Advanced Mapping Options” on page 268).

## Leaf Elements That Contain Markup

A problem can occur when an element is populated at runtime by a Java or ECMAScript operation. The element might receive data that contains markup (strings with illegal characters, such as < and >). If the Integration activity were to map the contents of such an element to a node in the Output DOM, the output document would be malformed. The Integration activity resolves this issue by mapping any data that contains markup to a new CDATA section in the target document.

**NOTE:** When markup is entered at design time, the behavior is different. If you type markup into a node, and you examine the raw XML in the Source view, you’ll see that markup characters typed into a node are converted to entities. For example, a “<” character is converted to `&lt;`.

## Adding a Map Action

1. Right-click the line in the action model that is one line above the position in which you want to place the Map action (the new action is inserted below the line that you selected).
2. Select **New Action > Map**.
3. In the **Source** section, select **XPath**.
4. Select a part (Input, Output, _SystemFault, or Project) from the list, then type the appropriate XPath expression, or use the ECMA Expression Builder to locate the element that you want.

   Together, the part name and XPath specify the Source context for the Map action.
5. Repeat Step 3 and Step 4 for the **Target** section.
6. If you want more control over mapping, select the **Advanced** (see "Advanced Mapping Options" on page 268) or **Content Editor** (see "Transforming Elements with the Content Editor" on page 270) options.

   You can click **Apply** to see the effect of the Map action without closing the dialog box. This allows you to make repetitive edits to a Map action and quickly see the results.
7. Click **OK**.

## Advanced Mapping Options

When you select the **Advanced** option in the **Map** dialog box, the **Advanced** dialog box is displayed. Options that you set in the Advanced dialog box only affect the current Map action.

The options in this dialog box give you fine control over how input part nodes are mapped to the output part.

### Copy Attributes

You use **Copy Attributes** to specify how attributes are mapped. **Copy Attributes** has the following options:

***Table 8-6*** *Copy Attributes Options*

| Option | Description |
|---|---|
| **For Non-leaf Root Nodes and Dependents** | Specifies that when a non-terminal (non-leaf) element is mapped to output, the element (minus its attributes) and its children are mapped to output. Attribute data for the children is included, but not for the original (parent) element. |
| **Never** | Specifies that no attribute data (whether for parent or leaf nodes) is carried over during mapping. |
| **Always** | Specifies that all attribute data, for all nodes, is mapped to output. |

### Deep Copy

The default Integration activity behavior is to map whole branches at a time (the target node plus all of its children). This is referred to as a *deep copy*. In some cases, you might want to turn off this behavior so that you can copy just the parent element without its children. Deselect **Map the Dependents** if you want to disable deep copy.

## Create Target

You use **Create Target** to create the destination node that you specified in the **Target** group in the Map action (see "Adding a Map Action" on page 268), based on whether or not the source node is present in the source DOM. By default the Integration activity always creates the target, whether or not the runtime source DOM contains the nodes specified in the Source XPath for mapping.

For example, in the Map action, you might have specified a Source XPath that looks like

```
$Input/Root/MySourceElement
```

In the Target XPath, you might have specified

```
$Output/Root/MyParentNode/SomeOtherElement
```

If the incoming Input document does not have a node corresponding to `Root/MySourceElement`, the Integration activity by default creates an empty `Root/MyParentNode/SomeOtherElement` node in the output DOM. In some cases, this might not be what you want. Using **Create Target** mapping, you can change the default behavior.

***Table 8-7*** *Create Target*

| Option | Description |
|---|---|
| **Only if Source exists** | Specifies that the Map Action is skipped (no target nodes are created in the output DOM) if the node specified in the Source XPath doesn't exist in the input message. |
| **Raise Error** | Specifies that if the input document doesn't contain the node specified in the Source XPath, it is considered an error at runtime. You should plan accordingly by wrapping your Map action in a Try/OnError block so that you can handle the error. |
| **Always** | Specifies that the target node should always be created (the default behavior). When **Always** is selected, you can use the **Default Value** field to specify a default data value for the target element. |

## Create Target as CDATA Section

You use **Create Target as CDATA** to control the way element data gets mapped into CDATA sections.

***Table 8-8*** *Create Target as CDATA Section*

| Option | Description |
|---|---|
| **Only if source contains markup** | Specifies that if the source data contains XML, HTML, or other types of markup in which illegal (in this context) characters are used, the data is placed, unmodified, in a CDATA section in the target DOM. This is the default behavior. |
| **Never** | Specifies that source data will not be wrapped in a CDATA section for output. Illegal characters that occur in the source data are converted to escaped entities, such as &gt; for >, on the output side. |
| **Always** | Specifies that whatever form the source data takes, it will get wrapped in a CDATA section on output. |

## Transforming Elements with the Content Editor

You use the Content editor to change the format and content of the input element to match that required by the output element. Using the Content Editor, you can slice the input data into small parts, move the parts to different locations relative to one another, add new parts, omit some parts, and apply functions to individual parts.

1 In the Action model, select two elements from different parts (for example, from the Input and Output parts) to map.

2 Select **New Action > Map**.

3 In the Map action dialog box (see "Adding a Map Action" on page 268), select the **Content Editor** check box, then select the **Content Editor** button.

4 If desired, click **New Sample** and type a sample string.

5 Click **OK** to return to the **Content Editor** dialog box.

6 In the **Sample** section, move the slider that is above the sample to the position where you want the first cut to take place, then move the slider that is below the sample to the position where you want the end cut to take place.

   The sliders determine how to take a substring from the input data.

7 Click **Apply**.

   The substring is copied to the **Result** field as a separate object.

8 Repeat Step 6 and Step 7 for each part of the sample in the order that you want.

   Using this method, you can build a new string out of substrings of the original input.

9 To change the format of an object in the Result field:

   9a Select an object.

   9b Click **Modify**.

   The **Start Cut at Characters** field displays the character in the string where the first cut will take place. The first **Occurrence** field displays when the cut will take place. In the previous illustration, the first cut will take place at the first occurrence of the letter l. The **End Cut at Characters** field displays that character in the string where the last cut will take place. The second **Occurrence** field displays when the cut will take place. The Offset field displays the number of characters from the beginning of the original string where the object will start. The Length field displays the length of the object.

   9c If desired, you can write an ECMAScript expression in the **Script Expression** field to modify the content region.

   The **%r** shown in the **Script Expression** field is a local variable representing the content region to which you want to apply a function. For example, to apply the `.toUpperCase()` function to the content region, you would write the Script Expression: `var test='%r'; test.toUpperCase()`.

   9d To assign a constant to an object, select **Constant**, then type a constant string.

   9e When you are finished mapping string formats with the Content Editor, click **OK** to save the changes and close the Content Editor.

10 Click **OK** to return to the action model.

# 9 Working with ECMA Expressions

This section provides details on using the ECMA Expression Builder.

## 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 email addresses and content.
  - Set entitlement grant/revoke rights and parameters.
  - Evaluate any past activity data to conditionally follow a workflow path by using the Condition activity.
  - Write different log messages that are conditionally triggered by using a single Log activity.

This section describes 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 Chapter 7, “Workflow Activity Reference,” on page 187. For descriptions of the system variables available in a workflow, see “Understanding Workflow Data” on page 163.

### 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).

# 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-related 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.

# 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 is 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.

The ECMA Expression Builder provides pick lists of available objects, methods, and properties in the top panes (all of which are resizable), with rollover tooltips 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 Objects** 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.

## Checking Syntax

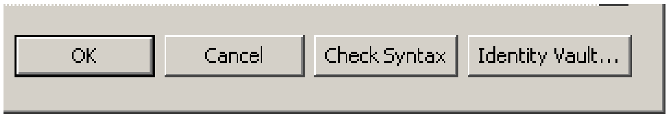
The ECMA Expression Builder includes a **Check Syntax** button. Clicking the button causes the ECMAScript interpreter to check the syntax of the expression. If there are problems involving ECMAScript syntax, an error message is displayed. You can then edit the expression and validate again as needed. 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 that might require selecting a DN from the Identity Vault (for example, Start, Approval, and Entitlement activities).

*Figure 9-1 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 dimmed (to indicate that it is unavailable) if you are not connected to the Identity Vault.

## ECMAScript Objects

This pane displays the names of objects 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 "Understanding Workflow Data" on page 163.

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. |

## Vault Expressions

This pane allows you to insert Entity definitions (see "Working with Entities and Attributes" on page 48) 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 Vault Expressions Pane 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).

## Using Special Characters

You can use special characters in literal strings in the ECMA Expression Builder by using escape sequences. Escape sequences begin with a backslash character ( \ ). The following table contains some commonly used escape sequences:

*Table 9-2 Escape Sequences*

| Escape Sequence | Character |
|---|---|
| \b | Backspace |
| \f | Form feed |
| \n | New line |
| \r | Carriage return |
| \t | Horizontal tab |
| \” | Double quote |
| \\ | Backslash (\) |
| \’ | Apostrophe |

You also can specify any Unicode character by using `\u` followed immediately by four hexadecimal digits. Here are some examples:

*Table 9-3 Escape Sequence Examples for Unicode Characters*

| Escape Sequence | Character |
|---|---|
| \u00A3 | Pound symbol (£) |
| \u20AC | Euro symbol (€) |

# 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`).

**NOTE:** Unlike ECMAScript that is available in other parts of the provisioning request definition editor, form action scripts are executed on the browser, not the server. All directory access from within a form action script is handled by AJAX calls from the browser to the server. See “Form Action Script Methods” on page 280.

- “Adding the Java Class to the User Application WAR” on page 275
- “Accessing Java from ECMAScript” on page 275

## 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\` 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, use ECMAScript syntax to 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");
```

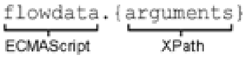
The ECMA Expression Builder is built on Mozilla Rhino. Rhino is an open source implementation of JavaScript written entirely in Java. For more information about accessing Java from ECMAScript, see Scripting Java (http://www.mozilla.org/rhino/ScriptingJava.html).

# About XPath Integration

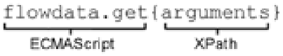
## XPath in Workflows

A provisioning request definition workflow supports a special object called *flowdata* (see “Understanding Workflow Data” on page 163). 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.parent/child[1]`<br>`flowdata.reason` |

To get an object from flowdata:

| Syntax | Examples |
|---|---|
|  | `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 273.

## XPath in Integration Activities

When you are working with an Integration activity, the ECMAScript interpreter recognizes a custom method called XPath(). This method allows expressions such as:

```
Input.XPath("GetBNQuoteSoapIn/GetBNQuote/sISBN")
```

When you use the ECMA Expression Builder, this type of expression is built for you automatically when you select nodes in ECMA Expression Builder pick lists.

The Integration activity uses the XPath addressing syntax adopted by W3C. The XPath syntax is similar to URL address syntax but includes many subtle and powerful features for addressing and manipulating XML. Some of the more common syntax rules are listed in the following table.

***Table 9-4*** *XPath Syntax*

| XPath Syntax | Description |
|---|---|
| / | The single forward slash represents an absolute path to an element. For example, `/ABC` selects the root element ABC. |
| // | Double slashes represent all elements in a path. //ABC selects all occurrences of ABC. For example, `//ABC//DEF` selects all DEF elements that are children of ABC. |
| * | The asterisk selects all elements located by the preceding path. For example, `*ABC/DEF` selects all elements enclosed by elements ABC/DEF. `//*` selects all elements. |
| [] | Square brackets specify a particular element. For example `/ABC[3]` selects the third element in ABC. This can also be used as a filter (similar to a Where clause in SQL). `//ABC["Table"]` selects all elements that have the content “Table.” |
| @ | The At sign selects elements with a specified attribute. For example, `/ABC@name` selects all elements in ABC that have an attribute called name. |
| \| | The vertical bar allows you to specify multiple paths. For example, `//ACB\|//DEF` selects all elements in ACB and in DEF. |
| $ | The dollar sign allows you to reference other documents in addition to the current one: `INVOICEBATCH/INVOICE[SELLER/NAME= $PROJECT/USERCONFIG/ COMPANYNAME]` |
| function() | XPath has numerous functions that you can add to your XPath addresses. For example, `//*[count(*)=2]` selects all elements that have two children. |
| math operator() | XPath has numerous math operators that you can add to your XPath addresses. For example, `/ABC\|position() mod 2 = 0]` selects all even elements in ABC. |

You can find the complete list of operators in the W3 Recommendation XML Path Language (XPath) (http://www.w3.org/TR/xpath.html).

## About Global Configuration Values Integration

Global configuration values (GCVs) are settings that are similar to driver parameters. Global configuration values can be specified for a driver set, as well as for an individual driver. If a driver does not have a GCV, the driver inherits the value for that GCV from the driver set.

GCVs are integrated with the workflow process through the ECMA Expression Builder, which provides a bridge to the GCVs from the driver and the GCV resource objects. To access a GCV through the ECMA Expression Builder, go to the variables pane and select any GCV from the GCV list. For GCV resource objects to be available on the variable pane, ensure that they are linked to the User Application driver. These GCVs are available to you in all activities on the workflow and not on the forms.

**NOTE**

- When you create a GCV of the type password-ref (Named Password), ensure that you create the GCV on the specific driver itself and *not* on the driver set.
- If you include a password-ref GCV in a workflow, you must also create and include a separate Boolean GCV named `allow-fetch-named-passwords` and set the value of that GCV to `true` If you do not include the second Boolean GCV, the workflow fails with a scripting error.

## About Global ECMAScripts Integration

ECMAScripts that are globally available on the User Application drivers or within the Identity Vault libraries can be accessed through the Expression Builder. Ensure that ECMA Scripts are imported into the provisioning request definition from the Overview page. For more information about creating provisioning request definitions, see Chapter 4, “Configuring Provisioning Request Definitions,” on page 77.

The global ECMAScripts are available throughout all your activities and on the forms. For making them available, you should include them as necessary on workflow activities, start activity, and forms.

## 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 consequences.

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

- Consider whether a task can be accomplished by using a custom Java class (that 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 and 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, such as 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 creating 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 the following:

```
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.

# ECMAScript Examples

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

## General Examples

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();
```

## Flowdata Examples

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

### Getting the Value of a Flowdata Variable

Suppose 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 Objects pane of the ECMA Expression Builder and double-clicking the `ApprovalStatus` attribute.

### Creating an XML Element with Child Element and Adding it to the Flowdata

You can add 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
```

# Form Control Examples

This section presents several examples of scripting with form controls.

## Retrieving the Value of a Form Field

Suppose you have a form field named FirstName, which you have mapped in the Post Activity of the Start (Request) to `flowdata.FirstName`. To get the value of this field in another activity, you would use the following expression:

```
flowdata.get('FirstName')
```

You can use the Expression Builder to create this expression. In the ECMAScript Objects pane, click the arrow next to **flowdata** and select **FirstName**.

## 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 the following:

```
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 Check Box Item

To populate list controls (for example, PickList or MVEditor) or the MVCheckbox control by 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();
```

## Comparing DNs

To compare DNs to find out if they are equal, use an expression like this:

```
if ( IDVault.dnCompare(flowdata.get('Activity3/CardRequest/Candidate'),recipient
)) true; else false ;
```

This comparison is case-insensitive. For example, the following DNs, when compared with dnCompare, returns True:

```
CN=jdoe,ou=users,ou=idmsample,o=acme
cn=JDOE,ou=users,ou=idmsample,o=acme
```

**WARNING:** Prior to Identity Manager 3.7, the name of the function used to compare DNs was “DNcompare.” With the 3.7 release, however, the name was changed to “dnCompare.” Please note that if you create one or more workflows that use “DNcompare” in a pre-3.7 Identity Manager environment and then migrate to version 3.7 or later, you must change all instances of “DNcompare” to “dnCompare” for your workflows to function properly.

### Error Handling

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();
```

## User Application API

This section describes the User Application API.

### Form Action Script Methods

Unlike the ECMAScript that runs in other components of the workflow, form script executes on the Web browser, not the server. All directory access from within form script is handled by AJAX calls from the browser to the server.

This section lists all form action methods and properties supported by the ECMA Expression Builder.

#### Form

Lets you work with Form methods.

##### focus(fieldname)

```
form.focus(fieldname)
```

Sets the focus to the specified field. For list-based or choice-based controls, sets the focus to either the selected choice or when no selection is made, it sets the focus to the first choice. If a fieldname parameter is passed and that field is list or choice based, it sets the focus on the choices corresponding to the values parameter. If the value is an array, only the first value is used to determine on which check box or radio button to set focus. If the specified field is invisible or disabled, this method has no effect.

### select(fieldname)

```
form.select(fieldname)
```

If no values parameter is passed in, this method sets the focus to the underlying text field. For list-based or choice-based controls, this method sets the focus to the selected choice or if no selection was made, to the first choice. If a values parameter is passed and the field is list-based or choice-based it sets the focus on the choices corresponding to the fieldname parameter. This method has no effect on disabled or invisible fields.

### activate(fieldname)

```
form.activate(fieldname)
```

A combination of setFocus() and select().

### setRequired(fieldname, is-required)

```
form.setRequired("fieldname", is-required)
```

Sets the field to required if is-required is True; otherwise, the field is optional. A field that is required blocks the form submission if the field is empty.

### InterceptAction(actionname, order, function)

```
form.interceptAction("actionname", "order", "function")
```

Allows you to intercept the script attached to an action button. The function passed in is executed based on the order parameter.

Valid actionname values are `SubmitAction` and `CancelAction`.

The choices for actionname for an approval form are: ApprovalAction, RefusalAction, DenyAction, UpdateAction, CancelAction and CommentAction.Valid values for the advice parameter are:Before: The function is called before the script attached to the button executes.after: The function is called after the script attached to the button executes.around: The function is passed a parameter that allows you to decide whether to execute the script attached to the button The following example shows the submit action intercepted. The form is only submitted if the user replies Yes.

```
window.inv=function (invocation) { if (confirm( "Are you sure you want to
submit?")) { var result = invocation.proceed(); return result; }; };
```

```
form.interceptAction("SubmitAction", "around", window.inv);
```

### getLocale()

```
form.getLocale()
```

Returns the current locale. Can be used as input for all methods that support a locale parameter.

### getRBMessage()

```
form.getRBMessage(key)
```

```
form.getRBMessage(key, value[s])
```

```
form.getRBMessage(key, value[s], bundle)
```

This method tries to find an entry with key in the resource bundle with ID bId. The resourcebundle Java class should extend the java.util.ListResourceBundle.Parameter. The parameter can be used to pass in replacements for parameters ({0}, {1}, etc) in message "msg"; for example:

```
var msg = frm.getRBMessage ("mykey", ["value0", "value1"], "mybundle");
```

### stringToDate()

```
form.stringToDate(date)
```

```
form.stringToDate(date, include-time)
```

Converts a date string to a Date. The format must correspond to the dateform for the current locale, as used in the DatePicker. The value of a DatePicker control can be converted with this method. Example:

```
form.showMsg("Date="+form.stringToDate(d,true));
```

### dateToString()

```
form.dateToString(date)
```

```
form.dateToString(date, include-time)
```

Converts a date to a string that can be stored in the DatePicker, for example:

```
var d = form.dateToString(new Date(), true);
```

```
form.setValues("hireDate", d);
```

**NOTE:** dateToString() is not available in environments running Identity Manager Home and Provisioning Dashboard.

### isValidDate(date)

```
form.isValidDate(date)
```

Use this to validate the correct format for a date string.

### isValidDate(date,include-time)

```
form.isValidDate(date, include-time)
```

Use this to validate the correct format for a date string.

### alert(string)

```
form.alert("msg")
```

Displays a message in an alert box.

### showMsg(string)

```
form.showMsg(msg, param, bId)
```

Adds a message to the status portion of the form. The msg string parameter can either contain the text of the message itself or it can contain a key pointing to an entry in the resource bundle bId. This method always tries to find an entry with the key `msg` in the resource bundle with the id `bId`. The `param` parameter can be used to pass in replacements for stakeholders ({0}, {1}, etc) in msg.

**NOTE:** If you want to add debugging messages to your script, it is better practice to use form.showDebugMsg().

Example:

```
form.showMsg("my message" {0},{1}", ["value0","value1"]);
```

### showWarning(string)

```
form.showWarning(msg, param, bId)
```

Adds a warning to the status portion of the form.

The msg string parameter can either contain the text of the warning itself or it can contain a key pointing to an entry in the resource bundle bId. This method always tries to find an entry with the key msg in the resource bundle with the id bId. The `param` parameter can be used to pass in replacements for stakeholders ({0}, {1}, etc) in msg.

**NOTE:** If you want to add debugging messages to your script, it is better practice to use form.showDebugMsg().

Example:

```
form.showWarning("my warning" {0},{1}", ["value0","value1"]);
```

### showError(string)

```
showError(msg, param, bId);
```

Adds an error message to the status portion of the form.

The msg string parameter can either contain the text of the error itself or it can contain a key pointing to an entry in the resource bundle bId. This method always tries to find an entry with the key msg in the resource bundle with the id bId. The `param` parameter can be used to pass in replacements for stakeholders ({0}, {1}, etc) in msg.

**NOTE:** If you want to add debugging messages to your script, it is better practice to use form.showDebugMsg().

Both normal and fatal errors block form submission. The distinction between a normal error and a fatal error is that normal errors get reset just before form validation occurs (because of a form submission). Fatal errors are remembered and therefore block the form submission unless you restart. A normal error only blocks submission if it is generated during the validation phase. If you add normal errors during onload or custom events, they are lost when the form is submitted.

**NOTE:** If you want to add debugging messages to your script, it is better practice to use form.showDebugMsg().

Example:

```
form.showError("my error" {0},{1}", ["value0","value1"]);
```

### showFatal(string)

```
form.showFatal("my fatal" {0},{1}", ["value0","value1"]);
```

Adds a fatal error message to the status portion of the form.

The msg string parameter can either contain the text of the fatal error itself or it can contain a key pointing to an entry in the resource bundle bId. This method always tries to find an entry with the key msg in the resource bundle with the id bId. The param parameter can be used to pass in replacements for stakeholders ({0}, {1}, etc) in msg.

Both normal and fatal errors block form submission. The distinction between a normal error and a fatal error is that normal errors get reset just before form validation occurs (because of a form submission). Fatal errors are remembered and therefore block the form submission unless you restart. A normal error only blocks submission if it is generated during the validation phase. If you add normal errors during onload or custom events, they are lost when the form is submitted.

**NOTE:** If you want to add debugging messages to your script, it is better practice to use form.showDebugMsg().

Example:

```
form.showFatal("my fatal" {0},{1}", ["value0","value1"]);
```

### enable(fieldname)

```
form.enable("fieldname")
```

Enables a field on a form.

### disable(fieldname)

```
form.disable("fieldname")
```

Disables a field on a form.

**NOTE:** A disabled field still sends data back to the workflow engine. The content of a disabled field is validated when submitting the form or when calling the field.validate() method.

### getValue(fieldname)

```
form.getValue("fieldname")
```

Returns the first value for the field. The type returned is always string, independent of the data type of the field. If the field does not have a value, the method returns an empty string if text can be entered into the field (like Text, TextArea, DatePicker, DNLookup) or it returns “undefined” if the control is choice-based (for example, StaticList, radio buttons, check boxes). For DN type controls, this method always returns the DN and never the display expression.

### getValues(fieldname)

```
form.getValues("fieldname")
```

Returns a string array containing the values. If no values are found, the array is empty (size = 0). For DN type controls, this method always returns the DN and never the display expression.

### setValues(fieldname)

```
form.setValues("fieldname", data-values, display values, KeepOldValues)
```

Sets a value. Supports multiple values. This method allows changing the available entries for list-based controls (for example, StaticList, MVCheckbox, PickList). By default, existing values are deleted unless the KeepOldValues parameter equals True. For non-list-based controls, the display values parameter is ignored.

## Field

Lets you work with Field methods.

### activate()

```
field.activate(value[s])
```

This method is a combination of field.focus() and field.select().

### disable()

```
field.disable()
```

Disable the field.

**NOTE:** A disabled field still sends data back to the workflow engine. The content of a disabled field is validated when submitting the form or when calling the field.

### enable()

```
field.enable()
```

Enable the field.

### fireEvent()

```
field.fireEvent("eventname")
```

Fires a custom event. Passes the name of the custom event that is fired. To get the values of the event that is fired, use form.getValues(event.getOrigin()).

### focus()

```
field.focus(value[s])
```

If no values parameter is passed in, this method sets the focus to the underlying text field. For list-based or choice-based controls, this method sets the focus to the selected choice if no selection was done to the first choice. If a values parameter is passed and if the field is list-based or choice-based, this method sets the focus on the choices corresponding to the values parameter. If the values parameter is an array, only the first value is used to determine the check box or radio button to set focus. This method has no effect on disabled or invisible fields.

### getLabel()

```
field.getLabel()
```

Gets the label associated with the field. If no label is found, this method returns the name of the field.

### getName()

```
field.getName()
```

Gets the name of the field.

### getValue()

```
field.getValue()
```

Returns the first value for the field. The type returned is always a string, independent of the data type of the field. If the field does not have a value, the method returns an empty string if text can be entered into the field (like Text, TextArea, DatePicker, DNLookup) or it returns “undefined” if the control is choice-based (for example, StaticList, radio buttons, check boxes). For DN type controls, this method always returns the DN and never the display expression.

### hide()

```
field.hide()
```

Hides this field.

### getValues()

```
form.getValues()
```

Returns a string array containing the requested values. If no values are found, the array is empty (size = 0). For DN type controls, this method always returns the DN and never the display expression.

### show()

```
field.show()
```

Shows this field.

### select()

```
field.select(value[s])
```

If no values parameter is passed in, this method sets the focus to the underlying text field. For list-based or choice-based controls, it sets the focus to either the selected choice or if no selection was made, it sets the focus to the first choice. If a values parameter is passed and if the field is list-based or choice-based, it sets the focus on the choices corresponding to the values parameter. If the value parameter is an array, only the first value is used to determine which check box or radio button to set focus on. This method has no effect on disabled or invisible fields.

### setRequired()

```
field.setRequired(is-required)
```

Sets the field to `required` if `isRequired` is `True` or `optional` otherwise. A field that is required blocks the form submission if it is empty.

### setValues(fieldname)

```
field.setValues(data-values, display-values, KeepOldValues)
```

Sets a value. Supports multiple values. This method allows changing the available entries for list-based controls (for example, StaticList, MVCheckbox, PickList). By default, existing values are deleted unless the `KeepOldValues` parameter equals `True`. For non-list-based controls, the display values parameter is ignored.

If you want to set or change the initial value of a field, you should do so in an onload event.

**NOTE:** This method triggers the onchange event for the field.

Examples:

```
field.setValues("cn=jdoe,ou=users,ou=mysample,o=novell"); // for a DNLookup
field.setValues(["jdoe@novell.com", "test@novell.com"]) // for an MVEditor
field.setValues(["W","B"],["White","Black"],true); // for a StaticList
```

### validate()

```
field.validate()
```

Triggers browser validation for the field. To validate the data entered in this field as soon as the user navigates to another field, call this method in the onchange event. Returns `True` if validation errors were detected; otherwise, returns `False`.

## Event

Lets you work with events.

### getEventName()

```
event.getEventName()
```

Returns the name of the event.

### getOrigin()

```
event.getOrigin()
```

Returns the name of the field from which the event was triggered.

### getValue()

```
event.getValue()
```

Returns a string that contains the first value in the event.

You should not use the value that is returned by this method, because it is possible that a user might have modified the data in the field after the event was triggered. Instead, you should use the value returned by the `form.getValue` method. For example, form.getValue(event.GetOrigin()). This ensures that you get the current value of the field. If you select event.getValue() from the pick list in the ECMA Expression Builder, form.getValue(event.GetOrigin()) is inserted.

### getValues()

```
event.getValues()
```

Returns an array of strings that contains all values in the event.

You should not use the value that is returned by this method, because it is possible that a user might have modified the data in the field since the event was triggered. Instead, you should use the value returned by the form.getValues method. For example, form.getValue(event.GetOrigin()). This ensures that you get the current value of the field. If you select event.getValues() from the pick list in the ECMA Expression Builder, form.getValues(event.GetOrigin()) is inserted.

# IDVault Functions

This section lists functions that are used with IDVault data.

## dnCompare

```
dnCompare(String dn1, String dn2)
```

Performs a case-insensitive comparison of DNs from the Identity Vault. Returns True if the DNs are the same.

A DN encapsulates a distinguished name (an LDAP name with context). The syntax of the DNs must conform to that specified in RFC 2253, which describes the String representation of DNs. The following DNs are all valid for use with dnCompare (and would return True if compared):

```
cn=jdoe,ou=users,ou=idmsample,o=acme
CN=jdoe,ou=users,ou=idmsample,o=acme
cn=JDOE,ou=users,ou=idmsample,o=acme
```

For more information about RFC 2253, see RFC 2353 (http://www.ietf.org/rfc/rfc2253.txt).

Example:

```
if ( IDVault.dnCompare(flowdata.get('Activity3/CardRequest/Candidate'),recipient
)) true; else false ;
```

## get()

```
get()
```

```
get(fieldname)
```

```
get(fieldname,dn)
```

```
get(fieldname,dn,entity-type)
```

```
get(fieldname,dn,entity-type,attribute)
```

This corresponds to the IDVault.get() function of the workflow script engine. Retrieves the values of the attribute for the given entity. The result is an array of values. If the field parameter is specified the result of the query result is used to refresh the content of the field. If not, the result is up to the form developer to use the result of the query. Example:

```
IDVault.get("assetProp",dn,"user","LastName");
```

`getUserName(dn,locale)`

Retrieves the user distinguished name formatted for the specified locale. The `FullName` DAL entity definition is used to retrieve the user name. The name is resolved based on the attribute key that matches the best fit for the `locale` argument. If you do not specify a locale, the attribute key default will be used.

`getUserName(dn)`

Retrieves the user distinguished name. The `FullName` DAL entity definition is used to retrieve the user name. The name is resolved based on the entity definition attribute key named `default`.

`getObjectName(dn,locale)`

Retrieves the resolved name of a user, role, resource, group, or container formatted for the specified locale. User names are resolved based on the attribute key that matches the best fit for the `locale` argument.

`getObjectType(dn)`

Retrieves the distinguished name of a user, role, resource, group, or container.

## execService()

`IDVault.execService(service)`

`IDVault.execService(service, param)`

`IDVault.execService(service, param, locale)`

Executes an AJAX service and the result is used to refresh the content of the field. The service must be registered in the UI control registry. The first column of the result list result is used for the display value, the second one for the data value. Example :

```
var r=IDVault.execService("dnlookup2",params);
var res=r?r["_data"]["raw"][dn]["value"]:"error encountered";

field.setValues("IDVault.execService(\"dnlookup2\") :"+res);
```

## Form Action Script Methods

Unlike the ECMAScript that runs in other components of the workflow, form script executes on the Web browser, not the server. All directory access from within form script is handled by AJAX calls from the browser to the server.

This section lists all form action methods and properties supported by the ECMA Expression Builder.

### Form

Lets you work with Form methods.

### *focus(fieldname)*

```
form.focus(fieldname)
```

Sets the focus to the specified field. For list-based or choice-based controls, sets the focus to either the selected choice or when no selection is made, it sets the focus to the first choice. If a fieldname parameter is passed and that field is list or choice based, it sets the focus on the choices corresponding to the values parameter. If the value is an array, only the first value is used to determine on which check box or radio button to set focus. If the specified field is invisible or disabled, this method has no effect.

### *select(fieldname)*

```
form.select(fieldname)
```

If no values parameter is passed in, this method sets the focus to the underlying text field. For list-based or choice-based controls, this method sets the focus to the selected choice or if no selection was made, to the first choice. If a values parameter is passed and the field is list-based or choice-based it sets the focus on the choices corresponding to the fieldname parameter. This method has no effect on disabled or invisible fields.

### *activate(fieldname)*

```
form.activate(fieldname)
```

A combination of setFocus() and select().

### *setRequired(fieldname, is-required)*

```
form.setRequired("fieldname", is-required)
```

Sets the field to required if is-required is True; otherwise, the field is optional. A field that is required blocks the form submission if the field is empty.

### *InterceptAction(actionname, order, function)*

```
form.interceptAction("actionname", "order", "function")
```

Allows you to intercept the script attached to an action button. The function passed in is executed based on the order parameter.

Valid actionname values are `SubmitAction` and `CancelAction`.

The choices for actionname for an approval form are: ApprovalAction, RefusalAction, DenyAction, UpdateAction, CancelAction and CommentAction.Valid values for the advice parameter are:Before: The function is called before the script attached to the button executes.after: The function is called after the script attached to the button executes.around: The function is passed a parameter that allows you to decide whether to execute the script attached to the button The following example shows the submit action intercepted. The form is only submitted if the user replies Yes.

```
window.inv=function (invocation) { if (confirm( "Are you sure you want to
submit?")) { var result = invocation.proceed(); return result; }; };
```

```
form.interceptAction("SubmitAction", "around", window.inv);
```

### *getLocale()*

```
form.getLocale()
```

Returns the current locale. Can be used as input for all methods that support a locale parameter.

***getRBMessage()***

```
form.getRBMessage(key)
```

```
form.getRBMessage(key, value[s])
```

```
form.getRBMessage(key, value[s], bundle)
```

This method tries to find an entry with key in the resource bundle with ID bId. The resourcebundle Java class should extend the java.util.ListResourceBundle.Parameter. The parameter can be used to pass in replacements for parameters ({0}, {1}, etc) in message "msg"; for example:

```
var msg = frm.getRBMessage ("mykey", ["value0", "value1"], "mybundle");
```

***stringToDate()***

```
form.stringToDate(date)
```

```
form.stringToDate(date, include-time)
```

Converts a date string to a Date. The format must correspond to the dateform for the current locale, as used in the DatePicker. The value of a DatePicker control can be converted with this method. Example:

```
form.showMsg("Date="+form.stringToDate(d,true));
```

***dateToString()***

```
form.dateToString(date)
```

```
form.dateToString(date, include-time)
```

Converts a date to a string that can be stored in the DatePicker, for example:

```
var d = form.dateToString(new Date(), true);
```

```
form.setValues("hireDate", d);
```

**NOTE:** dateToString() is not available in environments running Identity Manager Home and Provisioning Dashboard.

***isValidDate(date)***

```
form.isValidDate(date)
```

Use this to validate the correct format for a date string.

***isValidDate(date,include-time)***

```
form.isValidDate(date, include-time)
```

Use this to validate the correct format for a date string.

***alert(string)***

```
form.alert("msg")
```

Displays a message in an alert box.

***showMsg(string)***

```
form.showMsg(msg, param, bId)
```

Adds a message to the status portion of the form. The msg string parameter can either contain the text of the message itself or it can contain a key pointing to an entry in the resource bundle bId. This method always tries to find an entry with the key `msg` in the resource bundle with the id `bId`. The `param` parameter can be used to pass in replacements for stakeholders ({0}, {1}, etc) in msg.

---

**NOTE:** If you want to add debugging messages to your script, it is better practice to use form.showDebugMsg().

---

Example:

```
form.showMsg("my message" {0},{1}", ["value0","value1"]);
```

***showWarning(string)***

```
form.showWarning(msg, param, bId)
```

Adds a warning to the status portion of the form.

The msg string parameter can either contain the text of the warning itself or it can contain a key pointing to an entry in the resource bundle bId. This method always tries to find an entry with the key msg in the resource bundle with the id bId. The `param` parameter can be used to pass in replacements for stakeholders ({0}, {1}, etc) in msg.

---

**NOTE:** If you want to add debugging messages to your script, it is better practice to use form.showDebugMsg().

---

Example:

```
form.showWarning("my warning" {0},{1}", ["value0","value1"]);
```

***showError(string)***

```
showError(msg, param, bId);
```

Adds an error message to the status portion of the form.

The msg string parameter can either contain the text of the error itself or it can contain a key pointing to an entry in the resource bundle bId. This method always tries to find an entry with the key msg in the resource bundle with the id bId. The `param` parameter can be used to pass in replacements for stakeholders ({0}, {1}, etc) in msg.

---

**NOTE:** If you want to add debugging messages to your script, it is better practice to use form.showDebugMsg().

---

Both normal and fatal errors block form submission. The distinction between a normal error and a fatal error is that normal errors get reset just before form validation occurs (because of a form submission). Fatal errors are remembered and therefore block the form submission unless you restart. A normal error only blocks submission if it is generated during the validation phase. If you add normal errors during onload or custom events, they are lost when the form is submitted.

---

**NOTE:** If you want to add debugging messages to your script, it is better practice to use form.showDebugMsg().

---

Example:

```
form.showError("my error" {0},{1}", ["value0","value1"]);
```

***showFatal(string)***

```
form.showFatal("my fatal" {0},{1}", ["value0","value1"]);
```

Adds a fatal error message to the status portion of the form.

The msg string parameter can either contain the text of the fatal error itself or it can contain a key pointing to an entry in the resource bundle bId. This method always tries to find an entry with the key msg in the resource bundle with the id bId. The param parameter can be used to pass in replacements for stakeholders ({0}, {1}, etc) in msg.

Both normal and fatal errors block form submission. The distinction between a normal error and a fatal error is that normal errors get reset just before form validation occurs (because of a form submission). Fatal errors are remembered and therefore block the form submission unless you restart. A normal error only blocks submission if it is generated during the validation phase. If you add normal errors during onload or custom events, they are lost when the form is submitted.

**NOTE:** If you want to add debugging messages to your script, it is better practice to use form.showDebugMsg().

Example:

```
form.showFatal("my fatal" {0},{1}", ["value0","value1"]);
```

***enable(fieldname)***

```
form.enable("fieldname")
```

Enables a field on a form.

***disable(fieldname)***

```
form.disable("fieldname")
```

Disables a field on a form.

**NOTE:** A disabled field still sends data back to the workflow engine. The content of a disabled field is validated when submitting the form or when calling the field.validate() method.

***getValue(fieldname)***

```
form.getValue("fieldname")
```

Returns the first value for the field. The type returned is always string, independent of the data type of the field. If the field does not have a value, the method returns an empty string if text can be entered into the field (like Text, TextArea, DatePicker, DNLookup) or it returns “undefined” if the control is choice-based (for example, StaticList, radio buttons, check boxes). For DN type controls, this method always returns the DN and never the display expression.

***getValues(fieldname)***

```
form.getValues("fieldname")
```

Returns a string array containing the values. If no values are found, the array is empty (size = 0). For DN type controls, this method always returns the DN and never the display expression.

***setValues(fieldname)***

```
form.setValues("fieldname", data-values, display values, KeepOldValues)
```

Sets a value. Supports multiple values. This method allows changing the available entries for list-based controls (for example, StaticList, MVCheckbox, PickList). By default, existing values are deleted unless the KeepOldValues parameter equals True. For non-list-based controls, the display values parameter is ignored.

## Field

Lets you work with Field methods.

***activate()***

```
field.activate(value[s])
```

This method is a combination of field.focus() and field.select().

***disable()***

```
field.disable()
```

Disable the field.

---

**NOTE:** A disabled field still sends data back to the workflow engine. The content of a disabled field is validated when submitting the form or when calling the field.

---

***enable()***

```
field.enable()
```

Enable the field.

***fireEvent()***

```
field.fireEvent("eventname")
```

Fires a custom event. Passes the name of the custom event that is fired. To get the values of the event that is fired, use form.getValues(event.getOrigin()).

***focus()***

```
field.focus(value[s])
```

If no values parameter is passed in, this method sets the focus to the underlying text field. For list-based or choice-based controls, this method sets the focus to the selected choice if no selection was done to the first choice. If a values parameter is passed and if the field is list-based or choice-based, this method sets the focus on the choices corresponding to the values parameter. If the values parameter is an array, only the first value is used to determine the check box or radio button to set focus. This method has no effect on disabled or invisible fields.

***getLabel()***

```
field.getLabel()
```

Gets the label associated with the field. If no label is found, this method returns the name of the field.

***getName()***

```
field.getName()
```

Gets the name of the field.

### *getValue()*

```
field.getValue()
```

Returns the first value for the field. The type returned is always a string, independent of the data type of the field. If the field does not have a value, the method returns an empty string if text can be entered into the field (like Text, TextArea, DatePicker, DNLookup) or it returns “undefined” if the control is choice-based (for example, StaticList, radio buttons, check boxes). For DN type controls, this method always returns the DN and never the display expression.

### *hide()*

```
field.hide()
```

Hides this field.

### *getValues()*

```
form.getValues()
```

Returns a string array containing the requested values. If no values are found, the array is empty (size = 0). For DN type controls, this method always returns the DN and never the display expression.

### *show()*

```
field.show()
```

Shows this field.

### *select()*

```
field.select(value[s])
```

If no values parameter is passed in, this method sets the focus to the underlying text field. For list-based or choice-based controls, it sets the focus to either the selected choice or if no selection was made, it sets the focus to the first choice. If a values parameter is passed and if the field is list-based or choice-based, it sets the focus on the choices corresponding to the values parameter. If the value parameter is an array, only the first value is used to determine which check box or radio button to set focus on. This method has no effect on disabled or invisible fields.

### *setRequired()*

```
field.setRequired(is-required)
```

Sets the field to `required` if `isRequired` is `True` or `optional` otherwise. A field that is required blocks the form submission if it is empty.

### *setValues(fieldname)*

```
field.setValues(data-values, display-values, KeepOldValues)
```

Sets a value. Supports multiple values. This method allows changing the available entries for list-based controls (for example, StaticList, MVCheckbox, PickList). By default, existing values are deleted unless the `KeepOldValues` parameter equals `True`. For non-list-based controls, the display values parameter is ignored.

If you want to set or change the initial value of a field, you should do so in an onload event.

**NOTE:** This method triggers the onchange event for the field.

Examples:

```
field.setValues("cn=jdoe,ou=users,ou=mysample,o=novell"); // for a DNLookup
field.setValues(["jdoe@novell.com", "test@novell.com"]) // for an MVEditor
field.setValues(["W","B"],["White","Black"],true); // for a StaticList
```

***validate()***

```
field.validate()
```

Triggers browser validation for the field. To validate the data entered in this field as soon as the user navigates to another field, call this method in the onchange event. Returns `True` if validation errors were detected; otherwise, returns `False`.

## Event

Lets you work with events.

***getEventName()***

```
event.getEventName()
```

Returns the name of the event.

***getOrigin()***

```
event.getOrigin()
```

Returns the name of the field from which the event was triggered.

***getValue()***

```
event.getValue()
```

Returns a string that contains the first value in the event.

You should not use the value that is returned by this method, because it is possible that a user might have modified the data in the field after the event was triggered. Instead, you should use the value returned by the `form.getValue` method. For example, form.getValue(event.GetOrigin()). This ensures that you get the current value of the field. If you select event.getValue() from the pick list in the ECMA Expression Builder, form.getValue(event.GetOrigin()) is inserted.

***getValues()***

```
event.getValues()
```

Returns an array of strings that contains all values in the event.

You should not use the value that is returned by this method, because it is possible that a user might have modified the data in the field since the event was triggered. Instead, you should use the value returned by the form.getValues method. For example, form.getValue(event.GetOrigin()). This ensures that you get the current value of the field. If you select event.getValues() from the pick list in the ECMA Expression Builder, form.getValues(event.GetOrigin()) is inserted.

## globalList(fieldname, key, locale)

```
IDVault.globalList("fieldname", "key", "locale")
```

Retrieves a global list from the directory abstraction layer, identified by the key of the global list. If the field name is specified, the result of the query is used to refresh the content of the field. To retrieve a list without storing the result in a field, use a null value for the fieldname parameter.The locale is optional. If locale is not specified, the locale in the HTTP request is used.

You can call this function only in the client.

Example:

```
IDVault.globalList("dallist", "departments", "en");
```

**NOTE:** Designer's ECMAScript Expression Builder does not support this function for workflow script engine expressions.

## globalQuery(fieldname, key, param)

```
globalQuery(fieldname, key, param)
```

Executes the predefined directory abstraction layer query key (see "Queries General Properties" on page 72). If the field name is specified, the result of the query is used to refresh the content of the field. To retrieve a list without storing the result in a field, use a null value for the `fieldname` parameter.The `param` parameter is used as input to the query. The parameter has the form `{`*`parname1`*`:`*`value`*`,`*`parname2`*`:`*`value`*`}`, in which the value can be an individual value or an array. The first column of the result list (always a DN) is used for the data value, and the second column is used for the display label.

You can call this function in the client or in the workflow script engine. Ensure that the first parameter (fieldname) is omitted if the function is used in the Identity Manager engine scripts. This function has the following syntax in workflows: `IDVault.globalQuery(key, param)`

Example (client):

```
IDVault.globalQuery("canchangepwd", "getsites");  // query without a parameter
```

```
IDVault.globalQuery("building", "getbuildings", {site:form.getValue("site")}); //
query with one parameter
```

```
IDVault.globalQuery("room", "getrooms", {site:form.getValue("site"),
building:form.getValue("building")}); // query with two parameters
```

Example (workflow):

```
IDVault.globalQuery("getbuildings", {}); // query without a parameter
```

```
IDVault.globalQuery("getbuildings", {site: "mysite"}); // query with one parameter
```

**NOTE:** Designer's ECMAScript Expression Builder does not support this function for workflow script engine expressions.

## containers(fieldname, rootdn, Search scope, Show DN)

```
IDVault.containers("test", rootdn, SearchScope, ShowDN)
```

Gets a list of containers, with the scope equal to "subtree" or the same level. The method returns an array with two entries, the first an array with the resulting DNs; the second entry an array with the display labels.

***Table 9-5*** *Container Parameters*

| Parameter | Description |
|---|---|
| `fieldname` | If the field name is specified, the result of the query is used to refresh the content of the field. To retrieve a list without storing the result in a field, use a null value for the `fieldname` parameter. |
| `rootdn` | If the `rootdn` parameter is empty, the root container for the default entity is used. |
| `scope` | If the `scope` parameter is empty, one-level is used. Valid choices for scope are “o” (onelevel) and “s” (subtree). |
| `showdn` | If the parameter `showDN` is true, the full DN is used for the display label. Otherwise the naming part (for example, ou, dc) is displayed. |

You can call this function in the client or in the workflow script engine. Ensure that the first parameter (fieldname) is omitted if the function is used in the Identity Manager engine scripts. This function has the following syntax in workflows: `IDVault.containers(rootdn, SearchScope, ShowDN)`

Example (client):

```
IDVault.containers("assetProp2", null, "o", true);  // set the entries in a
StaticList to all containers directly under the root DN of the default entity
```

Example (workflow):

```
IDVault.containers("o=system", "o", true); // get containers under the specified
root DN
```

**NOTE:** Designer’s ECMAScript Expression Builder does not support this function for workflow script engine expressions.

# nrfRequest Properties and Methods

This section lists functions used with the nrfRequest object in Roles Based provisioning request definitions.

## Role Request Properties

***Table 9-6*** *Role Request Properties*

| Property | Code | Description |
|---|---|---|
| NEW_REQUEST | 0 | Set by the User Application on a newly created nrfRequest object. |
| SOD_APPROVAL_START_PENDING | 2 | The Role Service driver attempts to start the SoD workflow again. This is used for requests in the SOD_APPROVAL_START_SUSPENDED mode. |
| SOD_APPROVAL_START_SUSPENDED | 3 | Occurs when the Role Service driver is not able to start an SoD workflow. A driver task then resets these requests to SOD_WORKFLOW_START_PENDING to retry the starting of the workflow. |

| Property | Code | Description |
|---|---|---|
| SOD_EXCEPTION_APPROVAL_PENDING | 5 | Set by the Role Service driver after successfully initiating an SoD exception workflow. |
| SOD_EXCEPTION_APPROVED | 10 | Set by the SoD exception workflow when the exception is approved. |
| APPROVAL_START_PENDING | 12 | The Role Service driver attempts to start the workflow. The request must be in APPROVAL_START_SUSPENDED mode. |
| APPROVAL_START_SUSPENDED | 13 | Occurs when the Role Service driver is not able to start the approval workflow. A driver task then resets these requests to APPROVAL_START_PENDING to try to start the workflow again. |
| APPROVAL_PENDING | 15 | Set by the Role Service driver after successful role assignment workflow. |
| APPROVED | 20 | Set by the role assignment workflow when the exception is approved. |
| ACTIVATION_TIME_PENDING | 25 | Set by the Role Service driver after obtaining all necessary approvals and the activation time has not yet been reached. |
| PROVISION | 30 | Set by the Role Service driver after all the necessary approvals have been approved and the role activation time has been reached. |
| PROVISIONED | 50 | Set by the Role Service driver after a role has been provisioned. |
| PROVISIONING_ERROR | 80 | Set by the Role Service driver when an error occurred during provisioning/deprovisioning. |
| SOD_EXCEPTION_DENIED | 90 | Set by the SoD exception workflow when the exception is denied. |
| DENIED | 95 | Set by the role assignment workflow when the exception is approved. |
| CLEANUP | 100 | Set when nrfRequest workflow should be cleaned up (deleted). This is intended to be triggered by a batch process some configurable amount of time after the request has either been fulfilled or denied. |

## Role Request Is Methods

- “isAddOperation” on page 300
- “isRemoveOperation” on page 300
- “isTargetDNaUserDN” on page 300
- “isTargetDNaRoleDN” on page 300
- “isTargetDNaContainerDN” on page 300
- “isTargetDNaGroupDN” on page 300

### isAddOperation

Returns True if this is an add operation. Occurs when the `AddedDN` attribute is not null.

```
public boolean isAddOperation() throws ActivityException
```

### isRemoveOperation

Returns True if this is a remove operation. Occurs when the `RemovedDN` attribute is not null.

```
public boolean isRemoveOperation() throws ActivityException
```

### isTargetDNaUserDN

Returns True if the target DN is a user DN.

```
public boolean isTargetDNaUserDN() throws ActivityException
```

### isTargetDNaRoleDN

Returns True if the target DN is a role DN

```
public boolean isTargetDNaRoleDN() throws ActivityException
```

### isTargetDNaContainerDN

Returns True if the target DN is a container DN

```
public boolean isTargetDNaContainerDN() throws ActivityException
```

### isTargetDNaGroupDN

Returns True if this is target DN is a group DN.

```
public boolean isTargetDNaGroupDN() throws ActivityException
```

## Role Request Get Methods

### getCN

Returns the CN.

```
public String getCn() throws ActivityException
```

### getCategoryLocaleString

Returns the category type localized string.

```
public String getCategoryLocaleString() throws ActivityException
```

### getCompletedWFEmailAddress

Gets the completed work flow email address. This is a convenience method for the NrfRequest ECMA script object.

```
public String getCompletedWFEmailAddress()
```

### getCorrelationId

Returns the Correlation ID.

```
public String getCorrelationId() throws ActivityException
```

### getDecisionDate

Returns the decision date.

```
public Date getDecisionDate() throws ActivityException
```

### getDescription

Returns the description.

```
public String getDescription() throws ActivityException
```

### getEndDate

Returns the end date.

```
public Date getEndDate() throws ActivityException
```

### getEntityKey

Returns the entity key.

```
public String getEntityKey()
```

### getLocale

Returns the preferred locale.

```
public Locale getLocale()
```

### getOperation

Returns either the Add operation or the Remove operation.

```
 public String getOperation() throws ActivityException
```

### getRequestDate

Returns the request date.

```
public Date getRequestDate() throws ActivityException
```

### getRequester

Returns the requester.

```
public String getRequester() throws ActivityException
```

### getStatusLocaleString

Returns the status localized string.

```
public String getStatusLocaleString() throws ActivityException
```

### getStartDate

Returns the start date.

```
public Date getStartDate() throws ActivityException
```

### getStatusValue

Returns the status int value.

```
public int getStatusValue() throws ActivityException
```

### getSourceDN

Returns the source DN.

```
public String getSourceDN() throws ActivityException
```

### getSourceDNDisplayName

Returns the source DN display name. This is a Role DN.

```
public String getSourceDNDisplayName() throws ActivityException
```

### getTargetDN

Returns the target DN affected by this operation.

```
public String getTargetDN() throws ActivityException
```

### getTargetDNDisplayName

Returns the Target DN display name. If it is:

- If it is a user, it returns first name + last name.
- If it is a group, it returns the description.
- If it is a role, it returns the description.
- If it is a container, it returns the target DN.

```
 public String getTargetDNDisplayName()
```

### getCategoryValue

Returns the category type int value.

```
public int getCategoryValue() throws ActivityException
```

## Role Vault API

This section describes the Role Vault API.

# About the Role Vault API

The Role Vault API allows you to programmatically access role assignments. It includes a set of methods for reporting on role assignments by container, user, group, or role, and for determining whether a user is in a particular role. You might use this API in conjunction with the Role Request activity to write your own workflow that can:

- Display the current role assignments for a particular user on a form.
- Allow the user to request a new role assignment.
- Verify whether the requested roles have any Separation of Duty (SoD) constraints then perform custom branching based on the existing SoD constraints. If the conflicts are allowed, you could invoke the Role Request activity to complete the assignment. Or, you can build in logic before allowing the user to make a role assignment request.

## Accessing the API

The Role Vault API is available from both forms and provisioning requests. The method signatures and return values are the same regardless of where they are used.You access the API by using the Expression Builder.

- In a workflow, you can access the Role Vault API from an activity (such as the Role Request Activity) through the Vault Expressions panel of the Expression Builder.
- From a form, you access the Role Script API by creating an event on the form and launching the Expression Builder from the event's action expression property. The supported script expressions are available under the Vaults Node in the ECMAScript Objects pane.

## Locale Handling

Some methods take a locale as a parameter. If you do not specify a locale, the User Application uses one of the following:

- The authenticated user's preferred locale when run from a form.
- The User Application's default locale when run in a workflow.

## Security Context

The Role Vault methods run in the following security context:

- On a form, the security context is that of the currently logged in user.
- On a workflow, the security context is the LDAP administrator's security context. Because this might return more data than an end user typically has access to, be careful how you display it.

## Working with the Role Script API

The Role Script API methods typically return one of four Role Vault Beans objects (IdentityBeans, RoleAssignmentBeans, RoleBeans, and SodBeans), or one of four Role Vault Bean objects (IdentityBean, RoleAssignmentBean, RoleBean, and SodBean). A Bean object is a specific entry in the Role Subsystem; for example, IdentityBean can represent a specific user in the Identity Vault. A Beans object is a collection or array of Bean objects; for example, IdentityBeans might contain one or more user objects represented as individual Bean objects. You iterate through the Beans, extracting each Bean and working with it as a specific object. The Beans classes implement the Java Iterable interface, so they allow you to obtain member values directly out of the list of Bean objects as arrays.

## Getting the Role

This example shows how to use the Beans methods to return a list of member values for the Bean. The expression is used to address the Approval activity to all the user DNs that are assigned to the role. The components of this workflow and their responsibilities are summarized in Table 9-7, “Sample Workflow for Roles,” on page 304.

*Figure 9-2 Sample Workflow for Roles*

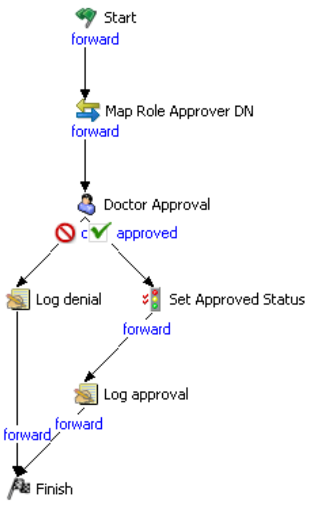


*Table 9-7 Sample Workflow for Roles*

| Activity | Activity Type | Description |
|---|---|---|
| Start | Start | Logical starting point for all workflows. |
| Map Role Approver DN | Mapping | The data item mapping source expression<br>`'cn=Doctor-east,cn=Level30,cn=RoleDefs,cn=RoleConfig,cn=AppConfig,' + PROVISIONING_DRIVER`<br>is mapped to the target:<br>`flowdata.roledn` |

| Activity | Activity Type | Description |
|---|---|---|
| Doctor Approval | Approval | This is where the Role Script API is used to define the addressee for the approval activity. The Addressee property uses this expression:<br>`java.util.Arrays.asList(RoleVault.getUsersToRoleAssignments(flowdata.get('roledn'), true).getTargetDn())`<br>- The expression<br>`RoleVault.getUsersToRoleAssignments(flowdata.get('roledn'),true)`<br>returns the RoleAssignmentBeans.<br>- The method call<br>`getTargetDn()`<br>is the RoleAssignmentBeans method that used to return an array of user DN strings.<br>- To convert the array to a list so it can be used by the workflow, use the<br>`java.util.Arrays.asList(...)` |
| Log Denial/Log Denial | Log | Used to write messages to the log to indicate the result of the request (approved or denied). |
| Finish | Finish | Logical end point of all workflows. |

## Retrieving SoD Violations

This example shows the methods to use to either array-like methods or a list iterator to walk through the individual RoleAssignmentBean objects contained in the RoleAssignmentBeans object. These methods are common to all of the Beans classes.

***Figure 9-3*** *Sample Workflow for Retrieving SoDs*

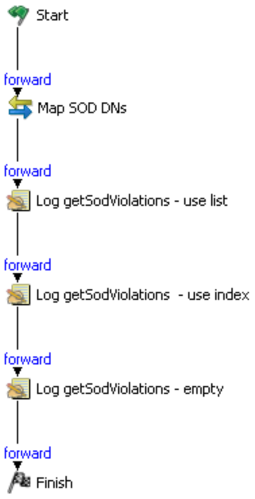

Start
forward
Map SOD DNs
forward
Log getSodViolations - use list
forward
Log getSodViolations - use index
forward
Log getSodViolations - empty
forward
Finish

***Table 9-8*** *Sample Workflow for Retrieving SoDs*

| **Activity Name** | **Activity Type** | **Description** |
|---|---|---|
| Start | Start | Logical starting point for all workflows. |
| Map SoD Dns | Mapping | |
| Log getSodViolations - Use List | Logging | Illustrates how to use an iterator to walk through the list of identityBean objects contained in the IdentityBeans returned by the RoleVault method getSodViolations().<br>The size() method is used to determine if any violations were returned.<br>`identitybeans.size()==0`<br>To return an iterator to walk the list, use this method:<br>`iterator=identityBeans.iterator()` |
| Log getSodViolations - use index | Logging | Illustrates how to use the index to access the array IdentityBean members returned from IdentityBeans using the Role Vault method getSodViolations(). This is similar to the list processing above, except that it uses the a For loop and a reference by index.<br>To loop through all the members in the array:<br>`for (i = 0; i < identityBeans.size(); i++ )`<br>To get the bean at position i in the array:<br>`identityBean = identityBeans.get(i);`<br>All beans support a getBean method that takes a dn string as the input paramter and returns the bean if there is one contained in the array for that dn.<br>All Beans classes support a getBean() that takes a DN string as the input parameter. It returns the bean if the array contains one for that DN. |
| Finish | Finish | Logical end point for all workflows. |

# Role Script API Reference

The Role Script API includes the methods available in the ECMA Expression Builder. These are the methods available for forms and workflows.

## Container and Group Methods

- "getContainersToRoleAssignments" on page 306
- "getGroupsToRoleAssignments" on page 307

### getContainersToRoleAssignments

```
RoleVault.getContainersToRoleAssignments(roleDN)
```

Returns a RoleAssignmentBeans object that contains a list of RoleAssignmentBean objects. The RoleAssignmentBean objects include the container DN(s) assigned to the specified roleDN.

### getGroupsToRoleAssignments

```
RoleVault.getGroupsToRoleAssignments(roleDN)
```

Returns a RoleAssignmentBeans object that contains a list of RoleAssignmentBean objects. The RoleAssignmentBean objects include the Group DNs assigned to the specified roleDN.

## Role Methods

### getRoleAssignmentCause

```
RoleVault.getRoleAssignmentCause(identityDn, roleDn)
```

Returns an IdentityBeans object that contains a list of IdentityBean objects. The IdentityBeans object shows the cause hierarchy for the role assignment for the specified `identityDn` and `roleDn`. For explicit assignments, it includes the DN of the user who made the request.

### getRoleInfo

```
RoleVault.getRoleInfo(roleDN, locale)
```

A role lookup method that returns a RoleBean.

### getRolesToContainerAssignments

```
RoleVault.getRolesToContainerAssignments(containerDN)
```

Returns a RoleAssignmentBeans object that contains a list RoleAssignmentBean. The RoleAssignmentBean objects contain the role DNs assigned to the specified `containerDN`.

### getRolesToGroupAssignments

```
RoleVault.getRolesToGroupAssignments(groupDN)
```

Returns a RoleAssignmentBeans object that contains a list of RoleAssignmentBean objects. They include the role DNs for the specified `groupDN`.

### getRolesToRoleAssignments

```
RoleVault.getRolesToRoleAssignments(roleDN)
```

Returns a RoleAssignmentBeans object that contains a list of RoleAssignmentBean objects. The RoleAssignmentBean objects include the child role DNs assigned to the specified `roleDN`.

### getRolesToUserAssignments

```
RoleVault.getRolesToUserAssignments(userDN)
```

Returns a RoleAssignmentBeans object that contains a list of RoleAssignmentBean objects. These beans include the role DN(s) assigned to the specified `userDN`.

### getRolesUserIn

```
RoleVault.getRolesUserIn(userDN)
```

Return a list of role DNs where the specified `userDN` is a member.

### getRoleOwners

```
RoleVault.getRoleOwners(roleDN)
```

Returns the IdentityBeans object that contains a list of IdentityBean objects. The IdentityBeans object shows the owners of the specified role DN.

### getRoleApprovers

```
RoleVault.getRoleApprovers(roleDN)
```

Returns the IdentityBeans object that contains a list of IdentityBean objects. The IdentityBeans object shows the approvers of the specified role DN.

## SoD Methods

- "getSodInfo" on page 308
- "getSodViolations" on page 308

### getSodInfo

```
RoleVault.getSodInfo(sodDN, locale)
```

Returns a SodBean.

### getSodViolations

```
RoleVault.getSodViolations(sodDn)
```

Returns an IdentityBeans object that contains a list of IdentityBean objects. They represent the users, groups, containers, and roles in violation of the specified sodDN.

## User Methods

### getUsersInRole

```
RoleVault.getUsersInRole(roleDN)
```

Returns a list of user DNs who are members of the specified roleDn.

### getUsersToRoleAssignments

```
RoleVault.getUsersToRoleAssignments(roleDN, direct)
```

Returns RoleAssignmentBeans object that contains a list of RoleAssignmentBean objects. The beans include the user DNs assigned to the specified roleDN. Specifying the direct argument means that only explicitly assigned to roles should be returned.

### isUserAppAdmin

```
RoleVault.isUserAppAdmin(userDN)
```

Returns True if the current user is a Global Administrator.

### isUserAttestationManager

```
RoleVault.isUserAttestationManager(userDN)
```

Returns True if the current user is an Attestation Officer.

### isUserComplianceAdmin

```
RoleVault.isUserComplianceAdmin(userDN)
```

Returns True if the current user a Compliance Administrator.

### isUserInRole

```
RoleVault.isUserInRole(roleDN, userDN)
```

Returns True if role is currently assigned to the specified user. The role can be assigned either explicitly or implicitly.

### isUserProvAdmin

```
RoleVault.isUserProvAdmin(userDN)
```

Returns True if the current user is a Provisioning Administrator.

### isUserRoleAdmin

```
RoleVault.isUserRoleAdmin(userDN)
```

Returns True if the current user is a Role Administrator.

## Hidden Methods

The following methods are part of the Role Vault API, but helper methods are not provided in the Expression Builder in Designer. You must manually type the method. The Expression Builder supports the following methods:

### findRoles

```
RoleVault.findRoles(String attributeKey, String relationalOp, String filterValue, int roleLevel, String locale)
```

| Parameter | Description |
|---|---|
| attributeKey | |
| relationalOp | Valid values are: less, not-less, less-or-equal,<br>not-less-or-equal, greater, not-greater, greater-or-equal, not-greater-or-equal, not-equals, equals, contains, not-contains, ends-with, not-ends-with, starts-with, not-starts-with |
| filterValue | |
| roleLevel | optional. |
| locale | optional. |

Returns a RoleBeans object. You can use then access a list of roleBeans based on the attributeKey and its relation to the filterValue. Use the relationalOp such as STARTWITH or CONTAINS. When roleLevel is specified, additional scoping is performed based on the roleLevel.

## findSods

```
RoleVault.findSods(String attributeKey, String relationalOp, String filterValue,
String locale)
```

Returns a SodBeans object that contains a list of sodBeans based on the attributeKey relation to the filterValue based on the relationalOp.

***Table 9-9*** *Enter Table Title Here*

| Parameter | Description |
|---|---|
| attributeKey | |
| relationalOp | Valid values are<br>less<br>not-less<br>less-or-equal<br>greater<br>not-greater<br>greater-or-equal<br>not-equals<br>equals<br>contains<br>not-contains<br>ends-with<br>not-ends-with<br>starts-with<br>not-starts-with |
| filterValue | |
| locale | Optional. |

## findSodsByRoles

```
findSodsByRoles(String roleDNs, String locale)
```

Returns a SodBeans object containing SodBean objects for the specified role DNs. Locale is an optional parameter.

## findSodsByRoles

```
findSodsByRoles(/*arraylist*/roledns, String locale)
```

Returns a SodBeans object containing a list of SodBean objects that include any of the specified roledns. Locale is optional.

## findSodsByRoles

```
findSodsByRoles(String role1, String role2, String locale)
```

Returns a SodBeans object containing a list of SodBean objects that have a conflict between the two roles specified. Locale is optional.

## getContainerSodViolations

```
getContainerSodViolations(String containerdn)
```

Returns a SodBeans object containing a list of SodBean objects where the specified container has roles assigned in violation of existing SoD contraints.

## getGroupSodViolations

```
getGroupSodViolations(String groupdn )
```

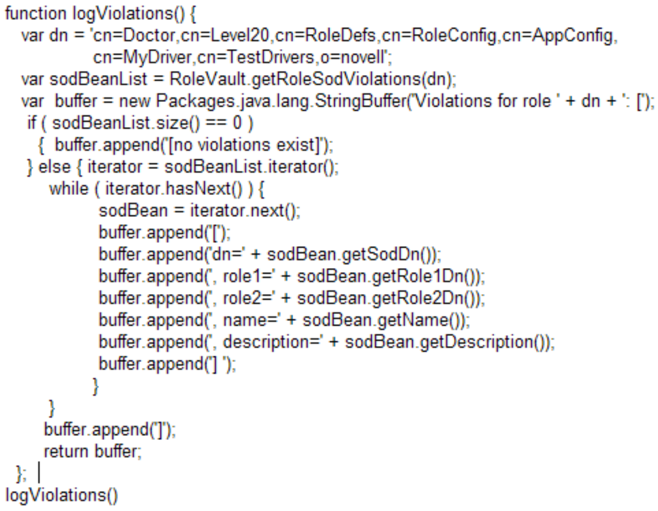
Returns a SodBeans object that contains a list of sodBeans where the specified group has roles assigned in violation of existing SoD contraints.

## getRoleSodViolations

```
getRoleSodViolations(String roledn )
```

Returns a SodBeans object that contains a list of SodBean objects where the specified roleDN has roles assigned in violation of existing SoD contraints.

Example:



```
function logViolations() {
   var dn = 'cn=Doctor,cn=Level20,cn=RoleDefs,cn=RoleConfig,cn=AppConfig,
              cn=MyDriver,cn=TestDrivers,o=novell';
   var sodBeanList = RoleVault.getRoleSodViolations(dn);
   var  buffer = new Packages.java.lang.StringBuffer('Violations for role ' + dn + ': [');
    if ( sodBeanList.size() == 0 )
      {  buffer.append('[no violations exist]');
    } else { iterator = sodBeanList.iterator();
        while ( iterator.hasNext() ) {
                 sodBean = iterator.next();
                 buffer.append('[');
                 buffer.append('dn=' + sodBean.getSodDn());
                 buffer.append(', role1=' + sodBean.getRole1Dn());
                 buffer.append(', role2=' + sodBean.getRole2Dn());
                 buffer.append(', name=' + sodBean.getName());
                 buffer.append(', description=' + sodBean.getDescription());
                 buffer.append('] ');
                }
        }
       buffer.append(']');
       return buffer;
  };
logViolations()
```

# Role Vault Bean API Reference

There are four Bean classes that represent the data returned by the Role Vault API. They are IdentityBean, RoleAssignmentBean, RoleBean, and SodBean. In many cases, multiple instances of these beans are returned. If a list is used to return multiple beans, you need to iterate through the list to retrieve the required data. Methods that are invoked from the form script that return multiple beans return a list of bean objects. To make it easier to manipulate data from a script, four other Beans classes are provided. They are IdentityBeans, RoleAssignmentBeans, RoleBeans, and SodBeans. These classes make it easier to retrieve data from individual bean classes without iterating through a list.

## IdentityBean

The IdentityBean class includes methods for retrieving a DN and an identity type. It includes the following methods:

- "getDn" on page 312
- "getType" on page 312

### getDn

`public java.lang.String getDn()`

Returns the DN of the identity.

### getType

`public java.lang.String getType()`

Returns the type of the identity. Valid types are:

- C: Container
- G: Group
- R: Role
- U: User

## IdentityBeans

The IdentityBeans class includes methods for manipulating one or more IdentityBeans objects or a list of IdentityBeans objects.

### getDns

```
public String[] getDns()
```

Returns a String Array of DNs.

### getType

```
public String[]getType()
```

Returns a String Array of identity types. Values are:

- U: Indicates User
- G: Indicates Group
- C: Indicates Container
- R: Indicates Role

### getIdentityBean

```
public IdentityBean getIdentityBean(Stringdn);
```

Returns the Identity Bean with the specified DN.

### size()

```
public int size()
```

Returns the number of Identity Beans.

### getBean()

```
public IdentityBean getBean(int n)
```

n is the index of the required bean.

Returns the IdentityBean at the specified index.

## RoleAssignmentBean

The RoleAssignmentBean class includes methods for manipulating a single RoleAssignmentBean. The methods include:

### getEffectiveTime

```
public long getEffectiveTime()
```

Returns the role’s effective time. (java.util.Date.getTime()).

### getExpirationTime

```
public long getExpirationTime()
```

Returns the role's expiration time.

### getTargetDn

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

Returns the DN. The type of DN is based on the context of the method returning the bean. It can be a DN for a user, group, container, or role.

### getType

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

Returns the role's assignment type. Values can be:

- G: Assignment was made through membership in a group.
- C: The assignment was made through membership in a container.
- E: The assignment was explicit.
- R: The assignment was inherited through the role hierarchy.

## RoleAssignmentBeans

The RoleAssignmentBeans class includes methods for manipulating one or more RoleAssignmentBeans objects as well as a list of RoleAssignmentBeans.

### getEffectiveTimes

public Long[] getEffectiveTimes()

Returns the role's effective time.

### getExpirationTimes

```
public long[] getExpirationTimes()
```

Returns the role's expiration times.

### getTargetDn

```
public String[] getTargetDn()
```

Returns target DNs. This could be a user, group, container, or role DNs based on the context of the method that returns the bean.

### getType()

```
public String[] getType()
```

Returns the assignment types. Values are:

- G: Assignment was derived from group membership.
- C: Assignment was derived from Container.
- E: Assignment was explicit.
- R: Assignment was through role hierarchy.

### getRoleAssignmentBean

```
public RoleAssignmentBean getRoleAssignmentBean(String targetDN)
```

Returns the role assignment bean with the corresponding DN.

### size

```
public int size()
```

Returns the number of role assignment beans.

### getBean

```
public RoleAssignmentBean getBean(int n)
```

Returns the Role assignment bean at the specified index.

## RoleBean

The RoleBean class includes methods for manipulating a single RoleBean.

### getDescription

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

Returns the localized role description.

### getName

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

Returns the localized role name.

### getRoleDn

```
public java.lang.String getRoleDn(String roleDN)
```

Returns the role’s DN.

### getRoleLevel

```
public long getRoleLevel()
```

Returns the role level.

### getRoleOwner

```
public java.lang.String getRoleOwner(String roleDN)
```

Returns the role’s owner.

### getRoleApprover

```
public java.lang.String getRoleApprover(String roleDN)
```

Returns the role’s approver.

## RoleBeans

The RoleBeans class includes methods for manipulating one or more RoleBeans as well as a list of RoleBeans.

### getDescription

```
public String[]getDescriptions()
```

Returns the localized role description.

### getNames

```
public String[] getNames()
```

Returns the localized role names.

### getRoleDns

```
public String[] getRoleDns()
```

Returns the role DNs.

### getRoleLevels

```
public long[] getRoleLevels()
```

Returns the role levels

### getRoleBean

```
public RoleBean getRoleBean(String roleDN)
```

Returns the RoleBean with the specified role DN.

### size

```
public int size()
```

Returns the number of RoleBeans in the list.

### getBean

```
public RoleBean getBean(int n)
```

Returns the RoleBean at the specified index (n).

### findRoles

Returns the roles based on filter values. It has four methods. These methods must be mentioned separately because different parameters are passed in each of these methods.

- findRoles

  ```
  public RoleBeans findRoles(String attributeKey, String relationalOp, String
  value, String locale, int roleLevel)
  ```

  Returns the roles based on filter values.

- findRoles

```
public RoleBeans findRoles(String attributeKey, String relationalOp, String
value, int roleLevel)
```

Returns the roles based on filter values.

- findRoles

```
public RoleBeans findRoles(String attributeKey, String relationalOp, String
value, String locale)
```

Returns the roles based on filter values.

- findRoles

```
public RoleBeans findRoles(String attributeKey, String relationalOp, String
value)
```

Returns the roles based on filter values.

## SodBean

The SodBean class includes methods for manipulating a single SodBean.

### getDescription

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

Returns the SoD's localized description.

### getName

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

Returns the SoD's localized name.

### getRole1Dn

```
public java.lang.String getRole1Dn()
```

Returns a role included in the SoD conflict. No special considerations are made between Role1Dn and Role2Dn.

### getRole2Dn

```
public java.lang.String getRole2Dn()
```

Returns a role included in the SoD conflict. No special consideration is made between Role1Dn and Role2Dn.

### getSodDn

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

Returns the SoD DN.

## SodBeans

The SodBeans class includes methods for manipulating one or more SoDBeans objects along with a list of SodBeans.

### getDescriptions

```
public String [] getDescriptions()
```

Returns the localized description of the SoD.

### getNames

```
public String [] getNames()
```

Returns the localized names of the SoD.

### getRole1Dns

```
public String[] getRole1Dns()
```

Returns the first role in the SoD conflict. No special consideration is made for Role1Dn and Role2Dn.

### getRole2Dns

```
public String[] getRole2Dns()
```

Returns the second role in the SoD conflict.

### getSodDns

```
public String[] getSodDns()
```

Returns SoD DNs.

### getSodBean

```
public SodBean getSodBean(String sodDn)
```

Returns the SodBean with the specified SodDn.

### size

```
public int size()
```

Returns the number of SodBeans.

### getBean

```
public SodBean getBean(int n)
```

Returns the SodBean at the specified index (n)

### findSodsByRoles

findSodsByRoles has six methods. These methods must be mentioned separately because different parameters are passed in each of these methods.

- findSodsByRoles

  ```
  public SodBeans findSodsByRoles(List<String> roleDns, String locale)
  ```

  Returns the SoDs based on a list of role DNs.

- findSodsByRoles

```
public  SodBeans findSodsByRoles(List<String> roleDns)
```

Returns the SoDs based on a list of role DNs by using the default application locale.

- findSodsByRoles

  ```
  public SodBeans findSodsByRoles(String targetRoleDn, String sourceRoleDn,
  String locale)
  ```

  Returns the SoDs based on source and target DNs.

- findSodsByRoles

  ```
  public SodBeans findSodsByRoles(String targetRoleDn, String sourceRoleDn)
  ```

  Returns the SoDs based on source and target DNs.

- findSodsByRoles

  ```
  public SodBeans findSodsByRoles(String[] roleDns)
  ```

  Returns the SoDs based on an array of role DNs.

- findSodsByRoles

  ```
  public SodBeans findSodsByRoles(String[] roleDns, String locale)
  ```

  Returns the SoDs based on an array of role DNs.

### findSods

findSods has two methods. These methods must be mentioned separately because different parameters are passed in each of these methods.

- findSods

  ```
  public SodBeans findSods(String attributeKey, String relationalOp, String
  value, String locale)
  ```

  Returns the SoDs based on the DAL attribute filter.

- findSods

  ```
  public SodBeans findSods(String attributeKey, String relationalOp, String
  value)
  ```

  Returns the SoDs based on the DAL attribute filter.

### getGroupSodViolations

```
SodBeans getGroupSodViolations(String groupDn)
```

Returns the SoD violations for a specified group.

### getRoleSodViolations

```
SodBeans getRoleSodViolations(String roleDn)
```

Returns the SoD violations for a specified role.

### getContainerSodViolations

```
SodBeans getContainerSodViolations(String containerDn
```

Returns the SoD violations for a specified container.

# 10 Configuring Provisioning Teams

The **Requests & Approvals** tab in the Identity Manager User Application includes a group of actions called **My Team's Work**. The **My Team's Work** actions allow you to work with team member tasks and requests in a workflow. This section describes how to create a team and define its characteristics (such as members, manager, and request rights).

## About Teams

A *team* identifies a group of users and determines who can manage provisioning requests and approval tasks associated with this team. The team definition consists of a list of team managers, team members, and team options, as described below:

- The *team managers* are those users who can administer requests and tasks for the team. Team managers can also be given permission to set proxies and delegates for team members. Team managers can be users or groups.
- The *team members* are those users who are allowed to participate on the team. Team members can be users, groups, or containers within the directory. Alternatively, they can be derived through directory relationships. For example, the list of members could be derived by the manager-employee relationship within the organization. In this case, the team members would be all users that report to the team manager.

  **NOTE:** The Provisioning Application Administrator can configure the directory abstraction layer to support cascading relationships so that multiple levels within an organization can be included within a team. The number of levels to include is configurable by the administrator.

- The *team options* determine the provisioning request scope, which specifies whether the team managers can act on an individual provisioning request, one or more categories of requests, or all requests. The team options also determine whether team managers can set proxies for team members or set the availability of team members for the purpose of delegation.

  **NOTE:** The User Application supports only a single level for proxy assignments. Proxy assignments are not propagated to multiple levels.

The Provisioning Application Administrator can perform all team management functions.

The teams you define are stored locally in the Designer project's `Provisioning\AppConfig\TeamDefs` directory. The filenames are derived from the object key with the `.team` or `.rbpmTeam` (for RBPM 3.7 or higher) extension.

Although a team can sometimes refer to a group in the Identity Vault, a team is not the same thing as a group. When you define a group in the Identity Vault, you identify a set of users that have something in common. However, the group does not automatically have the capabilities of a team within the User Application. To take advantage of the team capabilities within the User Application, you must define a team that points to the group.

## About Team Requests

A *team request object* specifies the requests that a team can work on. The request rights specify the actions that team managers can perform on the provisioning requests and tasks.

The team definition has a *one-to-many relationship* with team request objects. This means that each team must have at least one team request object defined for it, but it can have more. Each team request object is associated with only one team definition.When you configure the team request object, you configure the task scope and permissions for the team manager.

The task scope options define the manager's ability to act on tasks:

- Where a team member is an addressee
- Where a team member is a recipient

**WARNING:** For security reasons, the recipient task scope option is disabled by default. Giving a team manager the ability to act on tasks where the recipient of the request is a team member can raise several security issues. First, the manager is then able to view data included on any of the forms that are displayed during the course of workflow execution, regardless of his or her trustee rights. Second, depending on the permission options (see below), a team manager could circumvent the approval process by claiming or approving the task, or by reassigning it to someone else.

The permissions options define the team manager's ability to:

- Initiate a provisioning request on behalf of a team member.
- Retract a provisioning request on behalf of a team member.
- Make a team member a delegate for other team members' provisioning requests.
- Claim a task for a team member who is a recipient or addressee (based on the task scope).
- Reassign a task for a team member who is a recipient or addressee (based on the task scope).

If both of the task scope options are disabled, the team manager cannot view or act on any active requests. Therefore, you must enable at least one of the Permissions options for the team manager.

**NOTE:** The User Application supports only a single level for delegate assignments. Delegate assignments are not propagated to multiple levels.

The trustee rights defined for a provisioning request apply to team managers who want to initiate a request on behalf of their team members. If the team manager does not have the trustee rights to a provisioning request definition, the team manager cannot make the request because the User Application does not display the provisioning request.

## Using a Team to Manage Direct Reports

You can define a team that allows managers throughout an organization to control the provisioning environment for their direct reports. If defined properly, a single team definition can be used to allow *all* managers to control the activities of their direct reports. This means that you do not need to define a separate team for each reporting relationship.

A team that supports direct reports within an organization has the following basic requirements:

- The members of the team are defined by the Manager-Employee relationship.
- The managers of the team are defined by a dynamic group that searches subcontainers, using a a search filter that retrieves only the managers.

After the team has been defined, the User Application allows all managers to use the team management actions within the navigation menu. This gives the managers the ability to control the provisioning activities that their direct reports can perform.

For details on how to define a team to manage direct reports, see "Creating a Team to Manage Direct Reports" on page 324.

# Managing Teams

## Creating a Team

To create a new provisioning team:

1 Launch the New Team Wizard in any of these ways:

From Designer's menus:

- Select **File > New > Provisioning Team**, then click **Next**.

From the Provisioning view:

Right-click **Provisioning Teams**, then select **New**.

The New Provisioning Teams dialog box displays. When the dialog box is launched from the File menu, it contains fields that are not displayed when it is launched from the Provisioning view.

2 Fill in the fields as follows:

| Field | Description |
|---|---|
| **Identity Manager Project and Provisioning Application** | Select the correct Identity Manager project and Provisioning Application.<br>**NOTE:** This field displays when you create queries from the **File** menu. |
| **Identifier** | Type a common name (CN) for the team. |
| **Display Label** | Type the name of the provisioning team. This is the name displayed in Designer and also in the User Application runtime. The label is localizable in the Team editor. |
| **Description** | Provide a description of the provisioning team. |
| **Domain** | Provide the domain for the team. It could be Roles, Resources, or Provisioning. |

3. Click **Finish**. The Team panel of the Provisioning Team editor displays.
4. Type a description.
5. To define the team’s members, do one of the following:
   - Click **DAL Relationship**, then select the relationship that represents the team’s membership.
   - Click **All Users** to select all users as members of this team.
   - Click **Identity Vault Objects**. Click +, then select the members from the Identity Vault. Members can be users, groups, containers, organizational units (OU), or organizations (O). Specifying an O or OU can impact the User Application’s runtime performance. The manager needs to search for the member using a select-pick list to reduce the performance impact.
6. Click **Permissions**. The Team Permissions Configuration page displays.

| Selection | Description |
|---|---|
| **Object Type** | Specifies the type of authorized object. |
| **Authorized Objects** | Specifies the name of the authorized object. |
| **Permission** | Specifies the permissions that the team has on that object. |

7. Click **Save**.

   The Team Permissions Configuration page is read-only. The object information is populated from the User Application.

You must save the Provisioning Team for it to be available to the User Application. See “Deploying Provisioning Objects” on page 31. A provisioning team creates one object (`srvprvRbpmTeam`) in the User Application driver Appconfig Teams node. The `srvprvRbpmTeam` contains the provisioning teams object.

**IMPORTANT:** The team and the team request objects represented a team in the User Application versions prior to 3.7. The team request object contained the request that could be accessed by the team. User Application 3.7, 4.0 and later teams store the permissions on individual requests or request containers eliminating the need for two objects. For more information on team authorization, refer to “Team Configuration” in the *NetIQ Identity Manager - Administrator’s Guide to the Identity Applications*.

## Deleting a Provisioning Team

You delete the Provisioning Team object from the Provisioning view by right-clicking the team, then selecting **Delete**. The Delete confirmation dialog box lets you specify whether to delete the object locally only, or from the Identity Vault during the next deploy of the parent object.

## Creating a Team to Manage Direct Reports

For information on creating the team, refer to “Team Configuration” in the *NetIQ Identity Manager - Administrator’s Guide to the Identity Applications*.

For more information on saving and deploying the team, see “Deploying Provisioning Objects” on page 31.

# 11 Configuring Roles

This section describes how to use the Roles Based tools to configure the contents of the **Roles** tab of the User Application.

## About the Roles Based Provisioning Module

The Identity Manager User Application's Roles Based provides an easy way to assign people to privileges in target systems through their role membership. The module allows you to easily ensure that employees have access to the resources they need.

A role defines a set of privileges related to one or more target systems or applications. When you assign a user to a role, the user is granted all the entitlements associated with the role (with any parameter values as specified in the Role editor). When a user is removed from a role, all entitlements granted when the user was assigned to the role are revoked. Only the entitlements granted through the role are revoked; entitlements the user has been granted through other means are not revoked.

## About the Role Catalog

The Role Catalog uses the Identity Vault to store role definitions that the User Application uses to determine:

- The set of roles that it can display or modify.
- The separation of duties (SoD) constraints between roles.
- The provisioning request definition to execute for role membership requests.
- The provisioning request definition to execute for SoD constraint exceptions.

The User Application ships with:

- Two roles based provisioning request definitions.
- A Roles Category list.
- Default role levels.
- Default mid-level system roles.

You use the Roles Based Provisioning tools to create new Role Catalog objects and customize existing ones for your own business needs. The Role Catalog node of the Provisioning view provides access to the Identity Manager Roles Based design and configuration tools.

You can use the Role Catalog node to import, export, deploy, validate, compare, and localize the roles definitions, separation of duties constraints, and the Roles Configuration object as a group or individually. It also provides access to each of the Roles Based tools.

When you use any of the editors available through the Role Catalog, you modify a set of local XML files. The local files are created when you add a Role Service driver to the Identity Manager project. The files are created in the workspace in the project's `Provisioning\AppConfig\RoleConfig` folder.

***Table 11-1** Local Roles Directories*

| Directory Name | Description |
|---|---|
| `RoleDefs` | Contains a folder for each role level. These folders can contain additional hierarchy levels, depending on how you set up your roles. If you add categories or additional levels, they are reflected in the folder structure. The folders contain the definitions for the roles within that level, and the file extensions correspond to the level. For example, the files in the level10 folder have `.level10` as the extension. |
| `SoDDefs` | Contains the files that define the separation of duties constraints. Files have the `.sod` extension. |

The Roles Configuration object definition file resides at the root of the `RoleConfig` folder. There can be only one such file, and its name is `configuration.roleconfig`.

The Role Catalog is deployed in the User Application driver's `AppConfig.RoleConfig` file.

# About the Role Editor

The Role editor allows you to create and configure the roles you want to assign and manipulate in the **Roles** tab of the User Application. You use the editor to define the role details.

## Understanding Role Hierarchy

The Roles Based uses a role hierarchy to simplify the model for assigning users to roles (and thus permissions to users). The role hierarchy allows you to assign roles in a more efficient way. For example, rather than assigning a user to twenty roles, you can do it by assigning role levels.

### About Role Levels

Role levels define role hierarchy. Roles defined at the highest level (called Business Roles) define operations that have business meaning within the organization. Mid-level roles (called IT Roles) supports technology functions. Roles defined at the lowest level of the hierarchy (called Permission Roles) define lower-level privileges.

A higher-level role automatically includes privileges from the lower-level roles that it contains. For example, a Business Role automatically includes privileges from the IT Roles that it contains. Similarly, an IT Role automatically includes privileges from the Permission Roles that it contains.

Role relationships are not permitted between peer roles within the hierarchy. In addition, lower-level roles cannot contain higher-level roles.

You can modify the label used for each role level in the User Application by defining localized strings for the level's Name and Description in the role configuration editor.

## About Role Containers

A role container is an organizational unit within the User Application driver. The User Application allows you to assign a role to a container. When you to assign a role to a container, the users in the container are assigned to the role. This type of role assignment is called an indirect assignment. Roles explicitly assigned to a user from within the User Application are called direct assignments.

Role containers reside under role levels. The User Application shows only the role containers that reside under the role level that you choose. You can create a role either directly in a role level, or in a container within the role level. Specifying the role container is optional.

**NOTE:** Designer does not allow you to create roles with the same name under different containers, although the role container that you specify while creating roles might be different for different roles.

To create a role container, right-click the role level in which you want to create the container, select **New Role Sub-Container**, specify an identifier for the new container, and click **OK**.

# Using the Role Editor

## Creating New Roles

1 Open the Create a Role Wizard in one of these ways:

- From the **Provisioning view**, open **Role Catalog**, right-click **Roles**, then select **New**.
- Right-click a role container, then select **New**.
- Select **File > New > Provisioning > Role**.

2 Fill in the fields as follows (* indicates a required field):

| Field | Description |
|---|---|
| I**dentity Manager Project and Provisioning Application*** | The name of the Identity Manager project and the provisioning application where you want to create the role.<br>**NOTE:** These two fields display when you launch the wizard from the **File** menu. |
| **Identifier (CN)*** | The unique identifier for the role. |
| **Display Name*** | The text displayed as the **Role Name** field in the User Application. You can translate this text into any of the languages supported by the User Application. For more information, see "Localizing Provisioning Objects" on page 39. |
| **Description** | The text displayed as the **Role Description** field in the User Application. You can translate this text into any of the languages supported by the User Applications. For more information, see "Localizing Provisioning Objects" on page 39. |
| **Role Container** | The root location of the roles objects within the User Application driver. It defaults to Business Role.<br>To specify a Role Container:<br>1. Click **Search** to open the container selection dialog box.<br>2. Select a container or subcontainer from the list.<br>3. Click **OK**. |

| Field | Description |
|---|---|
| **Category*** | Allows you to categorize roles. Categories are used for filtering role lists in the User Application. The category names are defined in the directory abstraction layer Role Category list. |
| **Trustee Rights** | Specifies the users, groups, or containers that can read, compare, and browse the roles. (Read, compare, and browse are the default privileges.) |

3 Click **Finish**. Designer creates the role locally and opens the Role editor.

4 Fill in the remaining fields in the **Overview** tab as described in Table 11-4, "Role Overview Properties," on page 333.

5 Click **Approval**.

For more information on completing the Contained Roles section, see "Specifying a Role Hierarchy" on page 329. For more information on how to use the Entitlements section, see "Specifying Entitlements" on page 330.

6 In the **Approval** section, choose **Approval Details**.

You are prompted for different values, depending your selection. See Table 11-3, "Role Approval Properties," on page 332 for information about each type.

7 Save the role.

For information on deploying a role, see "Deploying Provisioning Objects" on page 31.

## Specifying a Role Hierarchy

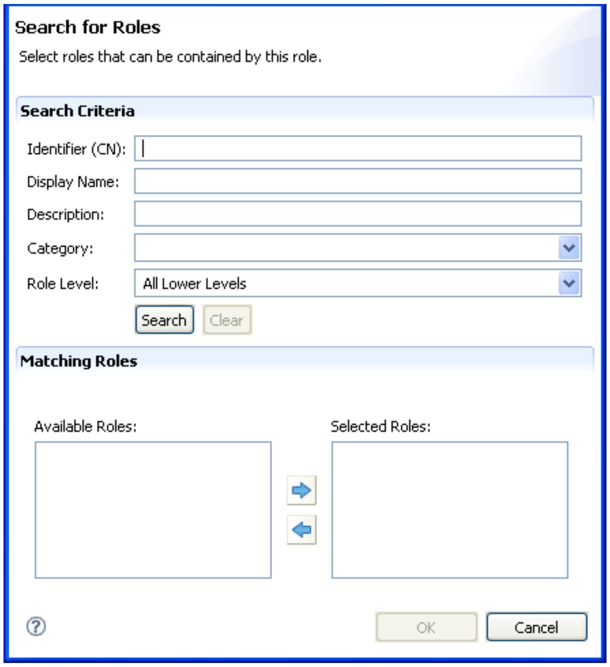
You specify a role hierarchy by defining the roles that contain other roles (called Role Relationships in the User Application).

To define a role hierarchy for a new or existing role:

1 Navigate to the **Associations** tab of the Role editor.

2 In the **Contained Roles** section, click + to add a lower-level role to the current role.

The current role must be a mid-level (IT Role) or top-level role (Business Role), because the lowest level role (Permission Role) cannot contain other roles. The Role Search dialog box displays:



Search for Roles

Select roles that can be contained by this role.

Search Criteria

Identifier (CN):

Display Name:

Description:

Category:

Role Level: All Lower Levels

Matching Roles

Available Roles:

Selected Roles:

3 To use the Role Search dialog box:

**3a** Specify the **CN**, **Display Name**, **Description**, **Category**, and **Role Level** on which you want to search.

For CN, Display Name, and Description, you can enter a wildcard (such as S*, *S), or regular expressions (such as [A-Z][a-z]*).

You can enter a value for all of the fields or none of the fields. If you do not supply a value in a particular field, the search returns all of the possible values for that field. If you enter values in one or more of the fields, the values are ANDed together to create the search filter. The search occurs on the roles defined locally, not the roles deployed to the driver.

Role Level values are All Lower Levels, Level 10, and Level 20 depending on the level of the currently selected role.

**3b** Click **Search**. Roles matching the search criteria are displayed in the **Matching Roles** section within the **Available** roles list.

**3c** Double-click the role or select the role and click ➡.

**3d** Click **OK** when you are done adding roles.

Designer closes the search dialog box and displays the roles you selected in the **Contained Roles** section.

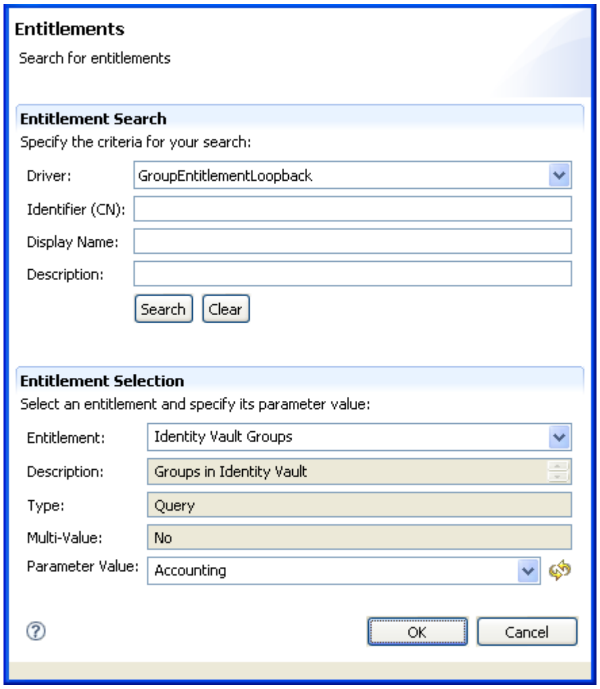
---

**NOTE:** The ability to add entitlements to a role will be deprecated in future. You should add entitlements to a resource by using the User Application Web client.

---

## Specifying Entitlements

1 In the **Entitlements** section, click + to add an entitlement for this role. The Entitlement Search dialog box displays:



**Entitlements**

Search for entitlements

**Entitlement Search**

Specify the criteria for your search:

Driver: GroupEntitlementLoopback

Identifier (CN):

Display Name:

Description:

**Entitlement Selection**

Select an entitlement and specify its parameter value:

Entitlement: Identity Vault Groups

Description: Groups in Identity Vault

Type: Query

Multi-Value: No

Parameter Value: Accounting

2 To complete the Entitlement search:

**2a** Choose the driver that contains the entitlement you want.

**2b** Specify the **CN**, **Display Name**, and **Description** on which you want to search.

You can enter a wildcard (such as S*, *S) or a regular expression (such as [A-Z][a-z]*), then click **Search**.

You can enter a value for all of the fields or none of the fields. If you do not supply a value in a particular field, the search returns all of the possible values for that field. If you enter values in one or more of the fields, the values are ANDed together to create the search filter. The search occurs locally. Entitlements contained by the selected driver that match the search criteria are displayed in the Entitlements Selection section.

The search is complete when the **Entitlement** field displays **<Select an Entitlement>**.

3 To complete the Entitlement selection:

**3a** Choose the entitlement from the Entitlement drop-down list.

The **Description**, **Type**, and **Multi-Value** fields are read-only. These values are obtained from the Entitlement definition.

**3b** Choose the parameter value.

| Type | Parameter Value Options |
|---|---|
| None | No parameter value needed. |
| User-defined | Specify your own value. |
| Admin-defined | Select from the available parameter values provided in the drop-down list. These values are retrieved from the local Entitlement definition. |
| Query | Select from the available parameter values provided in the drop-down list. Designer connects to your Identity Vault to retrieve a cached list of available parameter values. These values were obtained by a prior run of the query defined in the Entitlement section.<br>If Designer is unsuccessful in retrieving these values, it displays a dialog box reporting the problem. You can either resolve these issues before attempting to create this entitlement reference again or simply enter your own value in the **Parameter Value** field.There are two buttons to help you retrieve your query parameter values:<br>⟳: **Refresh cached query result**. Click this button if you want Designer to attempt to connect to the Identity Vault and retrieve the query parameter values. This is most useful if Designer was not able to connect to the Identity Vault on the first attempt.<br>▶: **Run query in Identity Vault**. Click this button if you want Designer to run the query in the Identity Vault and return the refreshed results. |

**3c** Click **OK** to save the definition. Designer displays the definition in the Entitlements table. Query parameter values are translated to the query's full CN when displayed in the table.

The entitlements defined for the role are triggered when the role is granted. However, if the entitlement is invalid, the role assignment still succeeds, but a message about the entitlement failure is written to the role service Audit log.

## Specifying Resource Associations

The Resource Associations table is read-only. It is populated through the User Application Web client.

***Table 11-2*** *Specifying Resource Associations*

| Field | Description |
|---|---|
| **Resource Name** | Name of the resource associated with the role. |
| **Association Description** | Description of the reason for associating the resource with the role. |
| **Association Values** | Values applied to the resource when the role is assigned. |

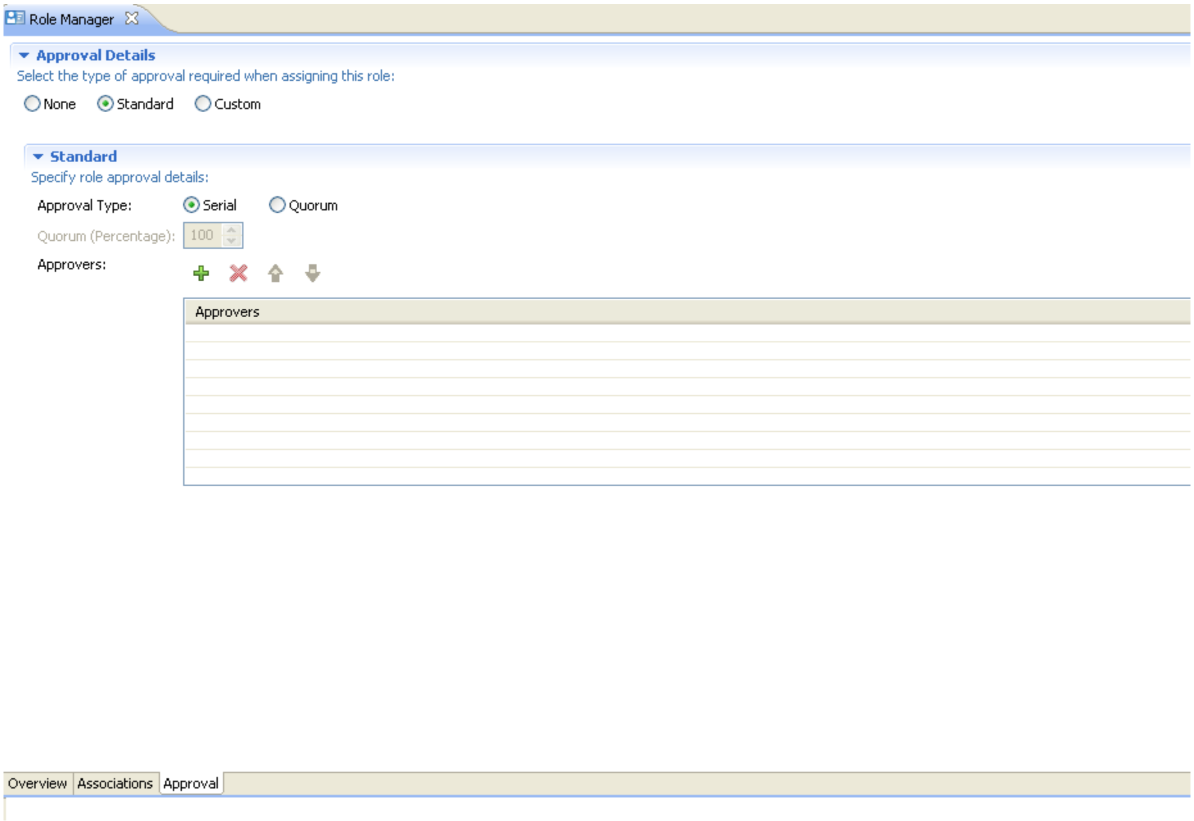
The information present in the Role editor is updated for all the roles when the Role Catalog is imported from eDirectory. You can only detect new resource associations but not the resource associations that have been removed in the User Application.

Ensure that the deleted resource associations are removed from the Resources list.

1. Before performing a Live Import from the Role Catalog, go to the Navigator View and navigate to the `\MyProject\Model\Provisioning\AppConfig\RoleConfig\ResourceAssociations` folder.
2. Remove all the files in the folder except the `ResourceAssociations.digest` file.
3. From the Provisioning View, select the **Role Catalog** object and run the **Live Import** to import all the resource associations again and to provide you the updated correct information.

## Specifying Role Approvals

Navigate to the **Approval** tab from the Role editor.



Select the type of approval required when assigning this role:

○ None ● Standard ○ Custom

Specify role approval details:

Approval Type: ● Serial ○ Quorum

Quorum (Percentage): 100

Approvers:

Select the type of approval required when assigning a role.

*Table 11-3 Role Approval Properties*

| Field | Description |
|---|---|
| **No Approval** | Select this option if the role does not require approval when requested. |

| Field | Description |
|---|---|
| **Standard Approval** | Select this option if the role requires approval when requested, and you want the approval to execute the standard provisioning request definition that ships with the Roles Based . You must select the type of approval (serial or quorum) and the valid approvers.<br>**Serial:** Select this option if you want the role to be approved by the approvers the Approvers list. The approvers are processed sequentially in the order they appear in the list.<br>**Quorum:** Select this option if you want the role to be approved in parallel and to be complete when the percentage of approvers specified is reached.<br>For example, if you wanted to require that 25 percent of approvers in the list approve the condition, you would specify Quorum and specify a number; the value is assumed to be a percentage. |
| **Approvers** | An approver can be a user, group, or role. To add approvers:<br>1. Click +.<br>If you are connected to the Identity Vault, the Browse Identity Vault dialog box automatically displays.<br>2. Navigate the Identity Vault to locate your approvers.<br>To locate roles, navigate to the User Application driver's `AppConfig.RoleConfig.RoleDefs container`.<br>3. Select the approver, then click **OK**.<br>If Designer is not able to connect to the Identity Vault, you can add the approver manually by clicking in the row and typing the approver's distinguished name, for example, `admin.novell`. Only deployed roles can be specified. |
| **Custom Approval** | **Approval process definition:** Select a provisioning request definition to execute when the role is requested. The approvals displayed in this list have **Process type of Role Approval**. |

## Role Properties Reference

*Table 11-4 Role Overview Properties*

| Section | Field | Description |
|---|---|---|
| **Role** | **Identifier** | The unique identifier for the role. |
| | **Display Name** | The text displayed in the **Roles** tab of the User Application as the Role Name. You can translate this text into any of the languages supported by the User Application. For more information, see "Localizing Provisioning Objects" on page 39. |
| | **Description** | The text displayed in the **Roles** tab of the User Application as the Role Description. You can translate this text into any of the languages supported by the User Application. For more information, see "Localizing Provisioning Objects" on page 39. |
| | **Role Level** | Defines the role's level in the role hierarchy. Level 30 roles are top-level roles. Level 20 roles are mid-level roles. Level 10 roles are the lowest-level roles. Higher-level roles include privileges from lower-level roles. |

| Section | Field | Description |
|---|---|---|
| **Categories** | **Available Categories** | Lists the categories that are available for the new role to be associated with. The items in this list are populated from the Role Category list in the directory abstraction layer. |
| | **Selected Categories** | Lists the categories that the new role is associated with. Use the **Add Category** and **Remove Category** buttons to associate the current role with one or more categories. |
| **Role Trustees** | **Trustees** | Specifies the users, groups, or containers that can read, compare, and browse the roles. (Read, compare, and browse are the default privileges.) |
| **Role Owners** | **Owners** | A user who is designated as the owner of the role definition. When you generate reports against the Role Catalog, you can filter these reports based on the role owner. The role owner does not automatically have the authorization to administer changes to a role definition. In some cases, the owner must ask a role administrator to perform any administration actions on the role. |

***Table 11-5** Role Approval Properties*

| Section | Field | Description |
|---|---|---|
| **Contained Roles** | **Contained Roles** | One or more roles of a lower level than the one being defined. |
| **Entitlements** | **Entitlements** | One or more Identity Vault objects that represent a resource in a connected system. |
| **Approval Options** | **No Approval** | Select this option if the role does not require approval when requested. |
| | **Standard Approval** | Select this option if the role requires approval when requested, and you want the approval to execute the standard provisioning request definition that ships with the Roles Based . You must select the type of approval (serial or quorum) and the valid approvers. |
| | **Approvers** | An approver can be a user, group, or role. |
| | **Custom Approval** | **Approval process definition**: Select a provisioning request definition to execute when the role is requested. The approvals displayed in this list have **Process type** of **Role Approval**. |

***Table 11-6** Role Association Properties*

| Section | Field | Description |
|---|---|---|
| **Resources** | **Resource Name** | Name of the resource associated with the role. |
| | **Association Description** | Description of the reason for associating the resource with the role. |
| | **Association Values** | Values applied to the resource when the role is assigned. |

# About the Separation of Duties Editor

The Separation of Duties (SoD) editor allows you to:

- Define a separation of duties constraint (or rule).
- Define how to process requests for exceptions to the constraint.

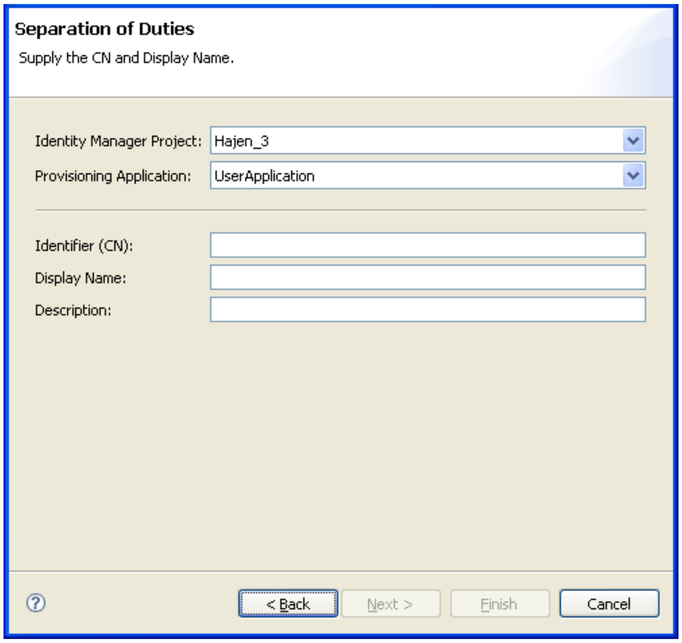
Each SoD constraint represents a rule that makes two roles mutually exclusive. If a user is in one role, he or she cannot be in the second role, unless there is an exception allowed for that constraint. You can define whether exceptions to the constraint are always allowed or are only allowed through an approval flow.

## Using the Separation of Duties Editor

To create a new separation of duties constraint:

1. Open the Separation of Duties Wizard in one of these ways:
   - From the **Provisioning view**, open **Role Catalog**, right-click **Separation of Duties**, then select **New**.
   - Select **File > New > Provisioning > Separation of Duties**.

   The SoD Wizard displays:



Separation of Duties

Supply the CN and Display Name.

Identity Manager Project: Hajen_3

Provisioning Application: UserApplication

Identifier (CN):

Display Name:

Description:

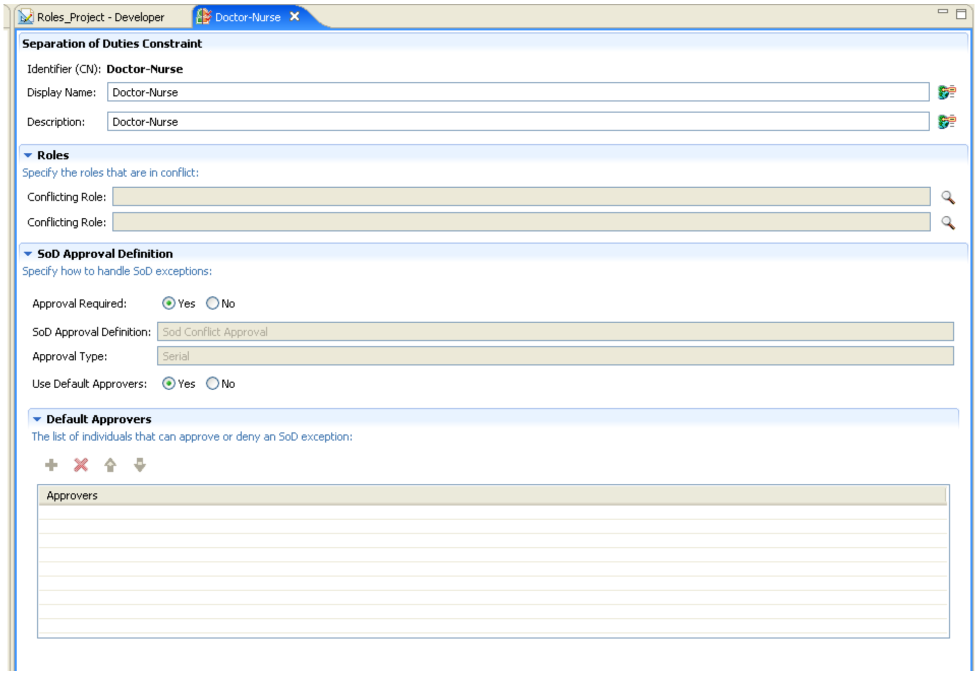
**2** Fill in the fields as follows:

* Indicates the field is required.

| Field | Description |
|---|---|
| **Identity Manager Project and Provisioning Application*** | The name of the Identity Manager project and the provisioning application where you want to create the SoD.<br>**NOTE:** These two fields display only when you launch the wizard from the **File** menu. |
| **Identifier (CN)*** | The unique identifier for the SoD. |
| **Display Name*** | The text used when the SoD name displays in the User Application. You can translate this text into any of the languages supported by the User Application. For more information, see "Localizing Provisioning Objects" on page 39. |
| **Description** | The text displayed as the SoD Description in the User Application. You can translate this text into any of the languages supported by the User Application. For more information, see "Localizing Provisioning Objects" on page 39 |

**3** Click **Finish**.

Designer creates the SoD constraint and launches the SoD editor.



**Separation of Duties Constraint**

Identifier (CN): **Doctor-Nurse**

Display Name: Doctor-Nurse

Description: Doctor-Nurse

**Roles**

Specify the roles that are in conflict:

Conflicting Role:

Conflicting Role:

**SoD Approval Definition**

Specify how to handle SoD exceptions:

Approval Required: ● Yes ○ No

SoD Approval Definition: Sod Conflict Approval

Approval Type: Serial

Use Default Approvers: ● Yes ○ No

**Default Approvers**

The list of individuals that can approve or deny an SoD exception:

| Approvers |
|---|

**4** Fill in the fields as described in Table 11-8, "Roles Configuration Properties," on page 340.

**5** Save and deploy the constraint definition.

# Separation of Duties Constraints Properties

Table 11-7 describes the fields on the SoD property page.

## Using the Separation of Duties Properties

*Table 11-7 Separation of Duties Properties*

| Section | Field | Description |
|---|---|---|
| **Separation of Duties Constraints** | **Identifier (CN)** | Read-only. Unique ID for the SoD. |
| | **Display Label** | The text displayed as the **SoD Constraint Name** in the User Application. You can translate this text into any of the languages supported by the User Application. For more information, see "Localizing Provisioning Objects" on page 39. |
| | **Description** | The text displayed as the **SoD Constraint Description** field in the User Application. You can translate this text into any of the languages supported by the User Application. For more information, see "Localizing Provisioning Objects" on page 39. |
| **Roles** | **Conflicting Role** | The name of the role for which you want to define a constraint.<br>Click **Browse** to locate a specific role from the available roles. See "Using the Role Search Dialog Box" on page 338.<br>A role defines a set of privileges related to one or more target systems or applications. |
| | **Conflicting Role** | The name of the role in conflict. Click **Browse** to locate an existing role from the available roles. This search excludes the role already selected. |

| Section | Field | Description |
|---|---|---|
| **Sod Approval Definition** | **Approval Required** | Select Yes if you want to launch a workflow when a user requests an exception to the SoD constraint.<br>Select No if the user can request an exception to the SoD constraint and no approval is required. In this case, the exception is never denied. |
| | **SoD Approval Definition** | Displays the read-only name of the provisioning request definition that executes when a user requests an SoD constraint exception. The value is derived from the Roles Configuration object. It is only executed when the **Approval Type** is **SoD allowed with workflow**. |
| | **Approval Type** | A read-only field that displays the processing type for the provisioning request definition displayed above. This value is derived from the Roles Configuration object. |
| | **Use Default Approvers** | Select **Yes** to use the default approvers defined in the Roles Configuration object. Does not enable the **Approvers** selection list in this property page.<br>**IMPORTANT:** When you choose this option, you must define the approvers in the role configuration editor. If you do not specify approvers, you are able to deploy the SoD, but users encounter a runtime error because there are no approvers defined.<br>Select **No** to enable the **Approvers** selection list in this property page.<br>If you change the selection from **Yes** to **No** and then perform a Compare, the objects are considered equal. After you specify Approvers, the comparison is no longer equal. |
| **Approvers** or<br>**Default Approvers** | **Approvers** | An approver can be a user, group, or role. To add approvers:<br>1. Click +.<br>If you are connected to the Identity Vault, the Browse Identity Vault dialog box automatically displays.<br>2. Navigate the Identity Vault to locate your approvers.<br>To locate roles, navigate to the User Application driver's `AppConfig.RoleConfig.RoleDefs container`.<br>3. Select the approver, then click **OK**.<br>If Designer is not able to connect to the Identity Vault, you can add the approver manually by clicking in the row and typing the approver's distinguished name, for example, admin.novell. Only deployed roles can be specified. |

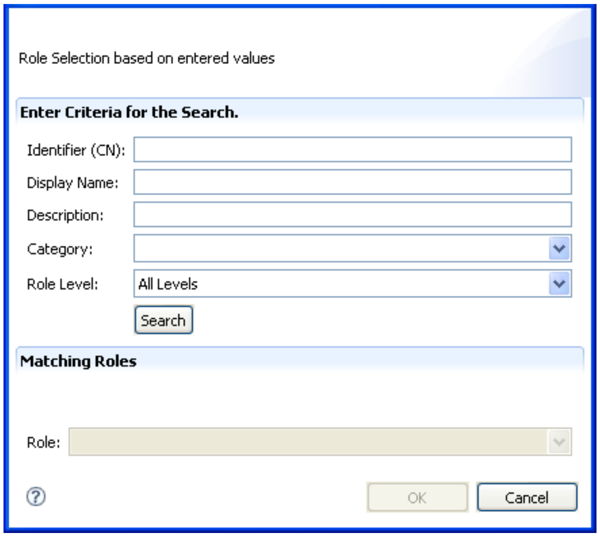
## Using the Role Search Dialog Box

The Role Search dialog box displays when you click **Browse** in the **Roles** section of the SoD editor. The dialog box helps you locate the existing roles for which you can create SoD constraints.

1 In the dialog box, specify the **CN**, **Display Name**, **Description**, **Role Category**, and **Role Level** on which you want to search.

For **CN**, **Display Name**, and **Description**, you can enter a wildcard (such as S*, *S) or regular expressions (such as [A-Z][a-z]*).

You can enter a value for all of the fields or none of the fields. If you do not supply a value in a particular field, the search returns all of the possible values for that field. If you enter a value in one or more of the fields, the values are ANDed together to create the search filter. The search occurs on the roles defined locally. Roles matching the search criteria are displayed in the **Matching Roles** selection list.



Role Level: All Levels

2 Select a role from the **Roles** selection list, then click **OK** to return to the SoD property page.

3 Click **OK**.

Clicking **OK** closes the Search for Role dialog box and populates the role in the SoD properties page. When no roles are available for the specified search criteria, the **OK** button is disabled.

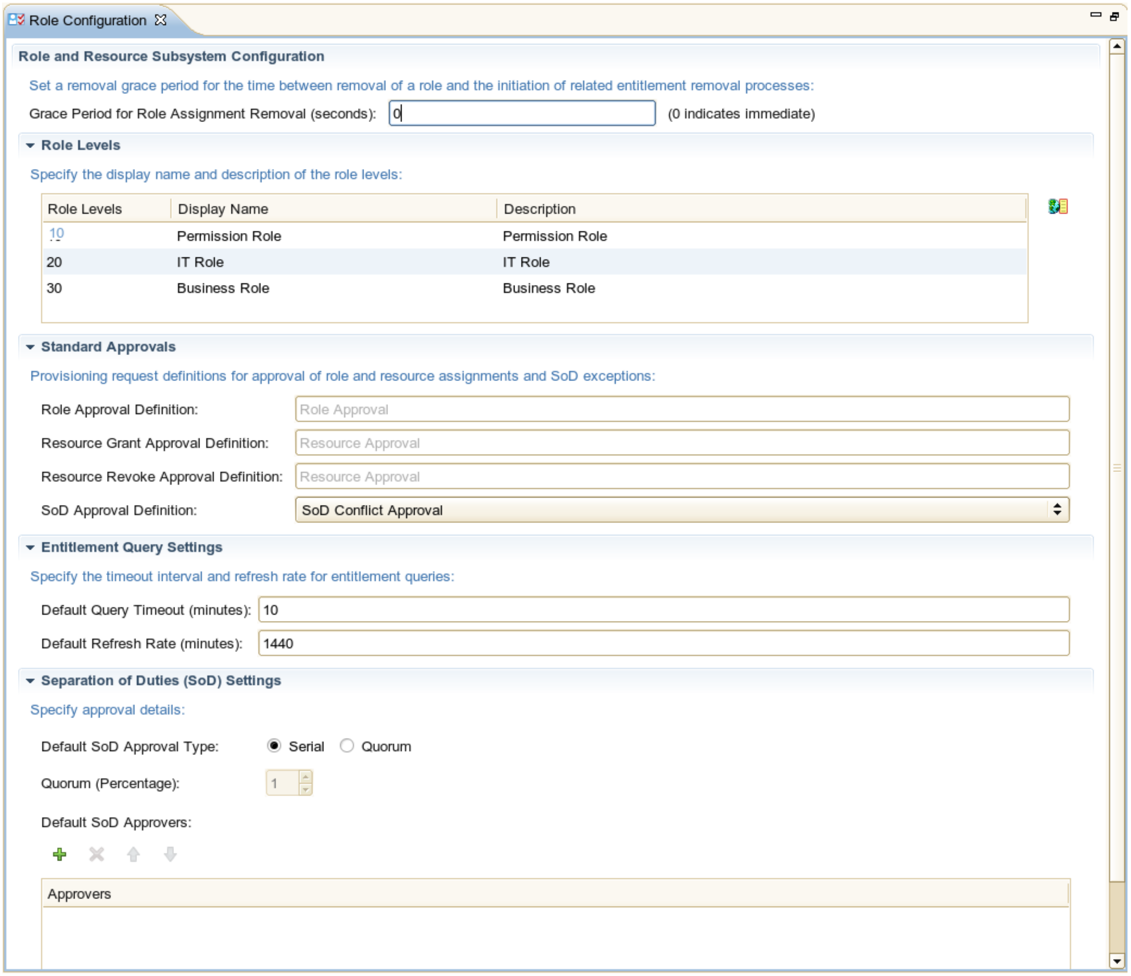
# About the Role Configuration Editor

The role configuration editor is a graphical tool for defining administrative settings for the Roles Configuration object. The Roles Configuration object resides in the Role Catalog (`nrfConfigurationobject`), and it contains basic settings for an instance of the Role subsystem. There is only one configuration object per Role Catalog, and it resides at the root of the `RoleConfig` folder. The Roles Configuration object is a protected object, so the menu items **Cut** and **Delete** are disabled. You can copy and paste this object from another project; a paste operation overwrites the existing object. To start the role configuration editor:

1 Expand the Provisioning view, then navigate to and open the Role Catalog.

2 Double-click the Role Configuration node.

Designer displays the role configuration editor.



Role Configuration

**Role and Resource Subsystem Configuration**

Set a removal grace period for the time between removal of a role and the initiation of related entitlement removal processes:

Grace Period for Role Assignment Removal (seconds): 0 (0 indicates immediate)

**Role Levels**

Specify the display name and description of the role levels:

| Role Levels | Display Name | Description |
|---|---|---|
| 10 | Permission Role | Permission Role |
| 20 | IT Role | IT Role |
| 30 | Business Role | Business Role |

**Standard Approvals**

Provisioning request definitions for approval of role and resource assignments and SoD exceptions:

Role Approval Definition: Role Approval

Resource Grant Approval Definition: Resource Approval

Resource Revoke Approval Definition: Resource Approval

SoD Approval Definition: SoD Conflict Approval

**Entitlement Query Settings**

Specify the timeout interval and refresh rate for entitlement queries:

Default Query Timeout (minutes): 10

Default Refresh Rate (minutes): 1440

**Separation of Duties (SoD) Settings**

Specify approval details:

Default SoD Approval Type: ● Serial ○ Quorum

Quorum (Percentage): 1

Default SoD Approvers:

| Approvers |
|---|

3 Fill in the fields as described in Table 11-8.

# Role Configuration Editor Properties

The properties you set in the role configuration editor are described in Table 11-8.

***Table 11-8** Roles Configuration Properties*

| Category | Field | Description |
|---|---|---|
| **General** | **Grace Period for Role Assignment Removal (seconds)** | Specifies the amount of time, in seconds, before a role assignment is removed from the Role Catalog.<br>The value is 0 by default. A grace period of zero means that when someone is removed from a role assignment, the removal happens immediately and the subsequent revocation of entitlements is initiated immediately.<br>You might use the grace period to delay the removal from a role of an account that would subsequently be re-added (for example if a person was being moved between containers). An entitlement can disable an account (this is the default) rather than removing it. |

| Category | Field | Description |
|---|---|---|
| **Role Levels** | **Role Levels** | Read-only level that defines the role hierarchy. The hierarchy rules are:<br>◆ Level 30 roles are higher-level roles in the hierarchy.<br>◆ Level 20 and Level 10 roles are lower-level roles.<br>◆ Level 30 roles include permissions from lower-level roles.<br>◆ Lower-level roles have permissions that are included in higher-level roles. |
| | **Display Name** | Specifies the text to display in the User Application **Roles** tab for each role level. By default, they are Permission Role (Level 10), IT Role (Level 20), and Business Role (Level 30). You can translate this text into any of the languages supported by the User Application. For more information, see "Localizing Provisioning Objects" on page 39.<br>The User Application caches this value in the RoleSystem cache holder. For your changes to Role Level Display Name to be visible in the User Application, you must flush the RoleSystem cache after you deploy the Role Configuration object. |
| | **Description** | Specifies the text to display in the User Application **Roles** tab for each Role Level Description. You can translate this text into any of the languages supported by the User Application. For more information, see "Localizing Provisioning Objects" on page 39.<br>The User Application caches this value in the RoleSystem cache holder. For your changes to Role Level Description to be visible in the User Application, you must flush the RoleSystem cache after you deploy the Role Configuration object. |

| Category | Field | Description |
|---|---|---|
| **Separation of Duties (SoD) Settings** | **Approval Type** | Select **Serial** if you want the SoD to be approved sequentially by the approvers in the order they appear in the approvers list.<br>Select **Quorum** if you want the SoD to be approved in parallel and to be complete when the percentage of users specified is reached.<br>For example, if you wanted to require that 25 percent of approvers in the list approve the condition, you would specify Quorum and specify a number; the value is assumed to be a percentage. |
| | **Approvers** | The actual list of individuals, users, groups, or roles that can approve or deny an SoD exception/override. This list can be overridden in the definition of an SoD constraint in the SoD editor. You can use the following buttons to manage the **Approvers** list:<br>◆ ✚ Click to add an approver. Adds the name to the bottom of the list.<br>◆ ✖ Click to delete the selected approver.<br>◆ 🔍 Click to access the Identity Vault to search for an approver to add.<br>◆ ⬇ Click to move an approver lower on the list.<br>◆ ⬆ Click to move an approver higher on the list. |
| **Standard Approvals** | **Role Approval Definition** | Read-only name of the provisioning request definition that runs for a role approval request for this driver. |
| | **SoD Approval Definition** | Read-only name of the provisioning request definition that runs for a SoD exception approval for this driver. |
| | **Resource Grant Approval Definition** | Read-only name of the provisioning request definition that runs for a resource grant approval request for this driver. |
| | **Resource Revoke Approval Definition** | Read-only name of the provisioning request definition that runs for a resource revoke approval request for this driver. |
| **Entitlement Query Settings** | **Default Query Timeout (minutes)** | The Roles Based periodically queries the external entitlement system to refresh the details of the entitlements that are displayed in the **Resource Catalog**.<br>You can limit the time that the system waits for the query result by using the **Default Query Timeout** option. |
| | **Default Refresh Rate (minutes)** | For the entitlement queries, you can set the time that the system waits for the query result by using the **Default Refresh Rate** option. |

# Importing Roles Defined in CSV Files

The Role Catalog provides a wizard for importing roles defined in a comma-separated values (CSV) file. For example, if you define the set of roles you want to implement by using a spreadsheet, you can export the definitions of those roles to a CSV file format, then use the Import Roles Wizard to add the roles to the Role Catalog.

## Setting Up the File to Import

When you create a file to use as input to the Import Roles wizard, you must follow the column layout defined in Table 11-9. In addition, you must also follow the CSV file format described in "Required CSV File Format" on page 346.

***Table 11-9*** *Import Record Format*

| Column Number | Field Name | Description |
|---|---|---|
| 1 | role level | Required field. Valid role levels are: 10, 20, and 30.<br>◆ 10: Permission Role<br>◆ 20: IT Role<br>◆ 30: Business Role<br>If an invalid role level is specified, the wizard writes the role record to the error file. It does not create the role.<br>For example:<br>`30` |
| 2 | sub-container | Optional field. The name of the subcontainer relative to the role level. The wizard creates any subcontainers that do not exist. There is no limit on the number of subcontainers, but five levels is the recommended maximum depth.<br>For example:<br>`"System\OTB"` |
| 3 | id | Required field. The role's identifier (CN). This name must be unique within the role level. If the CSV file contains multiple rows with the same ID, the wizard imports and creates a record for the first one it encounters. It then writes any subsequent records with the same ID to the error file.<br>For example:<br>`"Doctor"` |

| Column Number | Field Name | Description |
|---|---|---|
| 4 | localized display names | Optional field. The the translated string used to display the role name. Accepts zero or more values. The value must be in this format:<br>`"java-locale-code~string"`<br>The ~ delimits the locale and its localized string. The \| symbol delimits each set of locale data.<br>For example:<br>`"en~Doctor\|it~Dottore\|fr~Docteur"`<br>If you do not want to localize display names, you can supply a single string. The wizard uses this string as the value for the default Designer locale upon import. If no value is present when you attempt to deploy the associated role, Designer generates a validation error. |
| 5 | localized descriptions | Optional field. The translated string used to display the role description. Accepts a list of zero or more values. The value must be in this format:<br>`"java-locale-code~string"`<br>The ~ delimits the locale and its localized string. The \| symbol delimits each set of locale data.<br>For example:<br>`"en~Doctor\|it~Dottore\|fr~Docteur"`<br>If you do not want to localize descriptions, you can supply a single string. The wizard uses this string as the value for the default Designer locale upon import. If no value is present when you attempt to deploy the associated role, Designer generates a validation error. |
| 6 | categories | Required field. This value should map to a valid category key based on the Role Category list defined in the directory abstraction layer. Accepts a list of zero or more values.<br>If you do not specify a value, the wizard inserts the role category key `default`.<br>If the value is invalid (it does not exist in the directory abstraction layer), the wizard still includes it in the newly created role; however, Designer's validation requires that this be fixed before the role can be deployed.<br>For example:<br>`"system\|doctor\|nurse"` |
| 7 | owners | Optional field. Represents the distinguished name of the owner of the role. Accepts a list of zero or more values.<br>For example:<br>`"admin.novell\|ablake.users.medical-idmsample.novell"` |

| Column Number | Field Name | Description |
|---|---|---|
| 8 | trustees | Optional field. Represents the distinguished name of the trustees of the role. Accepts a list of zero or more values.<br>For example:<br>`"admin.novell\|ablake.users.medical-idmsample.novell"` |
| 9 | contained roles | Optional field. The role level and common name (CN) of the child or contained roles. Accepts zero or more values.<br>For example:<br>`"10~Administer Drugs\|10~Fill Prescriptions"` |
| 10 | entitlements | Optional field. DN and parameter values of the role’s entitlements. Accepts zero or more values.<br>For example:<br>`"Groups.GroupEntitlementLoopback.TestDrivers.novell~Medical Operations\|Groups.GroupEntitlementLoopback.TestDrivers.novell~Pharmacy"` |
| 11 | approval workflow | Optional field. Specifies the name of the provisioning request common name and its quorum value. Valid values include:<br>◆ None: Provide the empty string ““.<br>◆ Standard: Supply key word `Standard` followed by the quorum value. For example:<br>`"Standard~50"`<br>◆ Custom: Enter the provisioning request definition CN. For example:<br>`"MyCustomPrdCN"`<br>Specify Quorum values as follows:<br>◆ Serial: Specify a quorum value of 0.<br>◆ Quorum percentage: Specify a value between 1-100. |
| 12 | approvers | Optional field. Represents the distinguished name (DN) of the approvers when the approval workflow value is **Standard**. The order of the approvers in this field is important if the quorum value is **serial**. Accepts zero or more values.<br>For example:<br>`"admin.novell\|ablake.users.medical-idmsample.novell"`<br>If the approval workflow is not Standard and you specify a list of approvers, the wizard writes the record to the error file because approvers are not valid. |

## General Field Formatting Rules

- Multi-value properties: Use the | symbol as the delimiter between values.
- DN properties: Specify in dot notation. Designer validates these properties on deploy to ensure that the values correspond to existing Identity Vault objects.
- Character set encoding must be UTF-8

# Required CSV File Format

When you create your spreadsheet to use as input to the Import Roles Wizard, keep in mind that the wizard expects a specific format. It expects a twelve-column document with the columns defined in the order described in Table 11-9 on page 343. The wizard also expects the input file to follow the CSV format rules defined in RFC4180. This format is briefly summarized below:

- Each role record is on a separate line.
- Each field in a role record is separated by a comma and is quoted.
- Each line is delimited by a line break (CRLF)
- The first line of the file can be a header line, but this is optional. The wizard allows you to identify whether the file contains a header line.
- If your file contains a header line, it must contain the role record’s field names. The header line field count must correspond to the field count of each line in the file.
- Quotes on numbers are not required.
- A role record example:

  ```
  20,"
  ","Doctor","en~Doctor|it~Dottore|fr~Docteur","en~Doctor|it~Dottore|fr~Docteur"
  ,"doctor",,"admin.novell|ablake.users.medical-
  idmsample.novell",,"Groups.GroupEntitlementLoopback.TestDrivers.novell~Medical
  Operations|Groups.GroupEntitlementLoopback.TestDrivers.novell~Pharmacy",,
  ```

- Quotes and nested quotes: You can use single quotes within a text field (such as Display name). Use double quotes to enclose a column.

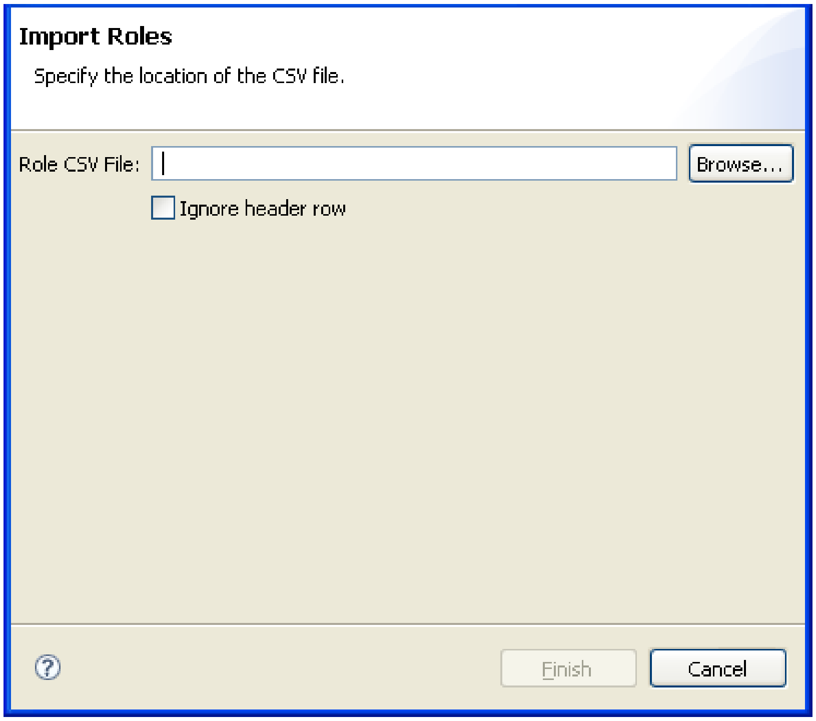
**NOTE:** For optional fields, the line must include an empty string "" as a placeholder.

# Using the Wizard to Import Roles

1. Open the Provisioning view of the Designer project where you want to import the roles.
2. Right-click a **Role Catalog**, the **Roles** node, or a role level (such as **Business Role**), then select **Import from CSV**.

   Designer launches the wizard.

   If you select a role level, the wizard imports only the roles for that level and ignores the other roles in the file.



3 Fill in the fields as follows:

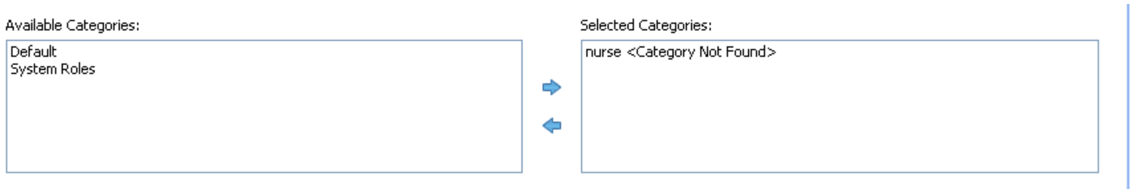
| Field Name | Description |
|---|---|
| **Role CSV File** | Specify the name and location of the CSV file you want to import. |
| **Ignore header row** | If the file you specify contains a header row, then select **Ignore header row in CSV file**. |

4 Click **Finish**.

The wizard reads the CSV file and adds all of the roles that meet the criteria for import. If the wizard encounters an error (see Error Handling for a list of possible errors), the wizard writes the role record to an error file.The wizard creates the error file in the same location as the Role CSV file to import, and it names the file the same name as the Role CSV file with the `_errors` appended to the name.

Only the errors identified in Error Handling are severe enough to prevent the wizard from creating the role. If the wizard encounters other types of errors, it adds the role, but you must make corrections before the role can be deployed. For example, if the category specified in the role is not yet added to the directory abstraction layer role category list, the role can be added, but Designer displays the role with an informational message as shown in Figure 11-1.

***Figure 11-1*** *Role Imported with Invalid Category Specified*



Roles that are created with errors like this cannot be deployed until the errors are corrected. The Project Checker notifies you of the errors if you attempt to deploy the roles or if you validate the roles objects.

**TIP:** If the role has no category, the wizard adds the Default category. If the category supplied does not exist, then it causes the error shown in Figure 11-1 on page 347.

## Error Handling

Table 11-10 on page 348 describes the cases where a role cannot be imported. When the wizard encounters these errors, it generates an error file and writes the complete role record to the file. It maintains the role's original column order except that it inserts a new column as the first column in the record. This column includes the error code. You can modify the associated role to fix the error directly within the error file, delete the error code column, then specify this error file as input to the wizard.

***Table 11-10*** *CSV Import Wizard Error Codes*

| Error Code | Description |
|---|---|
| `INVALID_LEVEL` | The row contains a container level that is not valid. This can occur if the level is missing or is not one of these values: 10, 20, or 30.<br>To fix this problem, change the level to 10, 20, or 30. |
| `ROLE_ID_NOT_UNIQUE` | A role with the specified ID in that level already exists. To fix this problem change the ID or the role level. |
| `INVALID_SUBCONTAINER_NAME` | The subcontainer name can contain only: alphanumeric characters, digits, underscore, and spaces. To fix this problem, update the subcontainer name to follow these rules. |
| `INVALID_ID_NAME` | The role ID contains invalid characters. To fix this problem, edit the name to follow the rules for valid characters: alphabetic characters, digits, underscores, and spaces. |
| `INVALID_ROLE_CN` | |

# 12 Configuring Resources

This section describes how to use the Resource editor to configure resources and entitlements.

## About Resources

A resource is any digital entity such as a user account, computer, or database that a business user needs to be able to access. The User Application provides a convenient way for end users to request the resources they need. In addition, it provides tools that administrators can use to define resources. Each resource is mapped to an entitlement. A resource definition can have no more than one entitlement bound to it. A resource definition can be bound to the same entitlement more than once, with different entitlement parameters for each resource.

## About the Resource Editor

- "About Resource Containers" on page 349
- "Using the Resource Editor" on page 349
- "Resource Property Reference" on page 352

### About Resource Containers

A resource container is an organizational unit within the User Application driver. The User Application allows you to assign a resource to a container. When you to assign a resource to a container, the users in the container are assigned that resource. This type of resource assignment is called an indirect assignment. Resources explicitly assigned to a user from within the User Application are called direct assignments.

Resource containers reside under the main Resources container in the Role Catalog. You can create a role either directly in Resources, or in a container within Resources. Specifying the resource container is optional.

To create a resource container, right-click **Resources**, select **New Resource Sub-Container**, specify an identifier for the new container, and click **OK**.

### Using the Resource Editor

#### Creating Resources

1 Open the Create Resource Wizard in one of these ways:

- From the Provisioning view, open **Role Catalog**, right-click **Resources**, then select **New**.
- Select **File > New > Provisioning > Resource**.

The Create Resource Wizard displays:

**2** Fill in the fields as follows (*indicates a required field).

| Field | Description |
|---|---|
| **Identity Manager Project and Provisioning Application** * | The name of the Designer project and the provisioning application where you want to create the resources.<br>**NOTE:** These two fields display when you launch the wizard from the **File** menu. |
| **Identifier (CN)*** | The unique identifier for the resource. |
| **Display Name*** | The text displayed as the **Resource Name** field in the User Application. You can translate this text into any of the languages supported by the User Application. For more information, see "Localizing Provisioning Objects" on page 39 |
| **Description** | The text displayed as the **Resource Description** in the User Application. You can translate this text into any of the languages supported by the User Application. For more information, see "Localizing Provisioning Objects" on page 39. |
| **Category*** | Allows you to categorize resources. Used for filtering resource lists in the User Application. The category names are defined in the directory abstraction layer Resource Category list. |
| **Trustee Rights** | Specifies the users, groups, or containers that can read, compare, and browse the resources. (Read, compare, and browse are the default privileges.) |

**3** Click **Finish**. Designer creates the resource locally and opens the Resource editor.

Use the **General** tab to modify the values you entered in the wizard, and to specify a Resource Owner. For more information on the **General** properties, see Table 12-2 on page 352.

## Specifying Entitlements

Navigate to the **Entitlement** tab. The Entitlement page is in read-only mode. It shows entitlements associated with a resource.

| Field | Description |
|---|---|
| **Entitlement Name** | The description of the entitlement if the entitlement has been imported and is known to the Designer Identity Vault. Otherwise, it is simply the entitlement DN. |
| **Entitlement Description** | Information about the entitlement description. It could also be the entitlement DN. |
| **Entitlement Value** | The entitlement value can be static or dynamic.<br>◆ If it is static, the value displayed is the one chosen by the resource administrator when the resource was created.<br>◆ If it is dynamic, the **Entitlement Value** is set at the request time under the specified **Request Form** field. |

## Specifying Request Form

Navigate to the **Request Form** tab. The **Request Form** page is in read-only mode. The information is displayed in the **Request Form** fields when a Resource is requested.

***Table 12-1*** *Form Field Properties*

| Field | Description |
|---|---|
| **ID** | The system-generated ID for the field. |
| **Label** | The display label to be used on the field. |
| **Binding** | ◆ Static, if the value is assigned at design time.<br>◆ Dynamic, if the value is assigned at request time. |
| **Data Type** | Can be a String, Integer, Boolean, List, or EntitlementRef type. |
| **Data Value** | The binding value can be static or dynamic.<br>◆ If it is static, it uses the value specified by the Resource Administrator.<br>◆ If it is dynamic, the value is specified at request time. |
| **List ID** | If the **Data Type** is List, then a **List ID** is specified. |
| **Entitlement DN** | If the **Data Type** is EntitlementRef, then an **Entitlement DN** is specified. |
| **Is Multi-Value** | Boolean. True, if users can specify more than one value for this field, else False. |
| **Is Hide** | Boolean. True, if the value is hidden during the request time. |

## Specifying Approvals

1. Navigate to the **Approvals** tab.
2. Select **Allow role approval to override resource approval** when you want the requesting system (such as role provisioning) to override approvals of the resource provisioning.
3. Click the **Grant** or **Revoke** tab, then select the type of grant or revoke for the resource.
   - **None**: Select this option when no approval is required for a resource grant or revoke request. Continue with Step 5.
   - **Standard**: Select this option if the resource requires approval for a grant or revoke request, and you want the approval to execute the standard provisioning request definition that ships with the Roles Based . Continue with Step 4.
   - **Custom**: Select this option when you want to specify a custom provisioning request definition for granting or revoking resources. You are prompted to select a provisioning request definition from the dropdown list. The list is populated with approvals whose Process Type is Resource. Continue with Step 5.

**4** For Standard Approval types, fill in the fields as follows:

| Field | Description |
|---|---|
| **Approval Type** | **Serial:** Select this option if you want the resource grant or revoke request to be approved by the approvers listed in the Approvers list. The approvers are processed sequentially in the order they appear in the list.<br>**Quorum:** Select this option if you want the resource grant or revoke request to be approved in parallel and to be complete when the percentage of approvers specified is reached. For example, if you wanted to require that 25 percent of approvers in the list approve the condition, you would specify Quorum and specify a number; the value is assumed to be a percentage. |
| **Approvers** | An approver can be a user, group, or role. To add approvers:<br>1. Click +.<br>If you are connected to the Identity Vault, the **Browse Identity Vault** dialog box automatically displays.<br>2. Navigate the Identity Vault to choose your approvers.<br>To locate roles, navigate to the User Application driver’s `AppConfig.RoleConfig.ResourceDefs container`.<br>3. Select the approver, then click **OK**.<br>If Designer is not able to connect to the Identity Vault, you can add the approver manually by clicking in the row and typing the approver’s distinguished name, for example, admin.novell. Only deployed roles can be specified. |

**5** Save the Resource definition.

# Resource Property Reference

***Table 12-2*** *Resource Overview Properties*

| Property | Description |
|---|---|
| Identifier (CN) | The unique identifier for the resource. |
| Display Name | The text displayed as the **Resource Name** field in the User Application. You can translate this text into any of the languages supported by the User Application. For more information, see “Localizing Provisioning Objects” on page 39 |
| Description | The text displayed as the **Resource Description** in the User Application. You can translate this text into any of the languages supported by the User Application. For more information, see “Localizing Provisioning Objects” on page 39. |
| Categories | Allows you to categorize resources. Used for filtering resource lists in the User Application. The category names are defined in the directory abstraction layer Resource Category list. |
| Trustees | Specifies the users, groups, or containers that can read, compare, and browse the resources. (Read, compare, and browse are the default privileges.) |

| Property | Description |
|---|---|
| Owners | A user who is designated as the owner of the resource definition. The resource owner does not automatically have the authorization to administer changes to a resource definition. |

# Importing Resources Defined in CSV Files

The Resource Catalog provides a wizard for importing resources defined in a comma-separated values (CSV) file. For example, if you define the set of resources you want to implement by using a spreadsheet, you can export the definitions of those resources to a CSV file format, then use the Import Resources wizard to add the resources to the Resource Catalog.

## Setting Up the File to Import

When you create a file to use as input to the Import Resources Wizard, you must follow the column layout defined in Table 12-3. In addition, you must also follow the CSV file format described in "Required CSV File Format" on page 356.

***Table 12-3*** *Import Record Format*

<table>
<tr><th>Column Number</th><th>Field Name</th><th>Description</th></tr>
<tr><td>1</td><td>Container</td><td>Optional field. If specified, the wizard creates the resource under this container. Otherwise, the driver creates the resource under the default resource container called <code>Resources</code>.<br>For example:<br><code>"Clinic",Doctor8,en~Doctor8,en~Doctor|it~Dottore|fr~Docteur,,admin.novell|ablake.users.medical-idmsample.novell,admin.novell|ablake.users.medical-idmsample.novell,admin.novell|ablake.users.medical-idmsample.novell,Standard~50,admin.novell|ablake.users.medical-idmsample.novell,MyCustomPrdCN,true</code><br>where <code>clinic</code> is the name of the container. The wizard will create the resource under <code>clinic</code>.<br>If you do not want to specify the container, you can supply an empty string. The wizard uses this string as the value for the container name. If no value is present, Designer does not create a resource and generates a validation error.</td></tr>
<tr><td>2</td><td>id</td><td>Required field. The resource's identifier (CN). This name must be unique. If the CSV file contains multiple rows with the same ID, the wizard imports and creates a record for the first one it encounters. It then writes any subsequent records with the same ID to the error file. For example:<br><code>"Doctor"</code></td></tr>
</table>

| Column Number | Field Name | Description |
|---|---|---|
| 3 | localized display names | Optional field. The translated string used to display the resource name. Accepts zero or more values. The value must be in this format:<br>`"java-locale-code~string"`<br>The ~ delimits the locale and its localized string. The \| symbol delimits each set of locale data.<br>For example:<br>`"en~Doctor\|it~Dottore\|fr~Docteur"`<br>If you do not want to localize display names, you can supply a single string. The wizard uses this string as the value for the default Designer locale upon import. If no value is present when you attempt to deploy the associated resource, Designer generates a validation error. |
| 4 | localized descriptions | Optional field. The translated string used to display the resource description. Accepts a list of zero or more values. The value must be in this format:<br>`"java-locale-code~string"`<br>The ~ delimits the locale and its localized string. The \| symbol delimits each set of locale data.<br>For example:<br>`"en~Doctor\|it~Dottore\|fr~Docteur"`<br>If you do not want to localize descriptions, you can supply a single string. The wizard uses this string as the value for the default Designer locale upon import. If no value is present when you attempt to deploy the associated resource, Designer generates a validation error. |
| 5 | categories | Required field. This value should map to a valid category key based on the resource Category list defined in the directory abstraction layer. Accepts a list of zero or more values.<br>If you do not specify a value, the wizard inserts the resource category key default.<br>If the value is invalid (it does not exist in the directory abstraction layer), the wizard still includes it in the newly created resource; however, Designer's validation requires that this be fixed before the resource can be deployed. |
| 6 | owners | Optional field. Represents the distinguished name of the owner of the resource. Accepts a list of zero or more values.<br>For example:<br>`"admin.novell\|ablake.users.medical-idmsample.novell"` |
| 7 | trustees | Optional field. Represents the distinguished name of the trustees of the resource. Accepts a list of zero or more values.<br>For example:<br>`"admin.novell\|ablake.users.medical-idmsample.novell"` |

| Column Number | Field Name | Description |
|---|---|---|
| 8 | Grant Approvers | Optional field. Represents the distinguished name (DN) of the approvers when the approval workflow value is Standard. The order of the approvers in this field is important if the quorum value is serial. Accepts zero or more values.<br>For example:<br>`"admin.novell\|ablake.users.medical-idmsample.novell"`<br>If the approval workflow is not Standard and you specify a list of approvers, the wizard writes the record to the error file because approvers are not valid. |
| 9 | Grant Approvers Workflow | Optional field. Specifies the name of the provisioning request common name and its quorum value. Valid values include:<br>◆ None: Provide the empty string `""`.<br>◆ Standard: Supply key word Standard followed by the quorum value. For example:<br>`"Standard~50"`<br>◆ Custom: Enter the provisioning request definition CN. For example:<br>`"MyCustomPrdCN"`<br>Specify Quorum values as follows:<br>◆ Serial: Specify a quorum value of 0.<br>◆ Quorum percentage: Specify a value between 1-100. |
| 10 | Revoke Approvers | Optional field. Represents the distinguished name (DN) of the approvers when the approval workflow value is Standard. The order of the approvers in this field is important if the quorum value is serial. Accepts zero or more values.<br>For example:<br>`"admin.novell\|ablake.users.medical-idmsample.novell"`<br>If the approval workflow is not Standard and you specify a list of approvers, the wizard writes the record to the error file because approvers are not valid. |
| 11 | Revoke Approvers Workflow | Optional field. Specifies the name of the provisioning request common name and its quorum value. Valid values include:<br>◆ None: Provide the empty string " ".<br>◆ Standard: Supply the keyword Standard followed by the quorum value. For example:<br>`"Standard~50"`<br>◆ Custom: Enter the provisioning request definition CN. For example:<br>`"MyCustomPrdCN"`<br>Specify Quorum values as follows:<br>◆ Serial: Specify a quorum value of 0.<br>◆ Quorum percentage: Specify a value between 1-100. |

| Column Number | Field Name | Description |
|---|---|---|
| 12 | Role Approval overrides Resource Approval | Boolean field for Role Approval to override Resource Approval. It takes True or False. |

### General Field Formatting Rules

- Multi-value properties: Use the | symbol as the delimiter between values.
- DN properties: Specify in dot notation. Designer validates these properties on deploy to ensure that the values correspond to existing Identity Vault objects.
- Character set encoding must be UTF-8.

## Required CSV File Format

When you create your spreadsheet to use as input to the Import Resources Wizard, keep in mind that the wizard expects a specific format. It expects a twelve-column document with the columns defined in the order described in Table 12-3. The wizard also expects the input file to follow the CSV format rules defined in RFC4180. This format is briefly summarized below:

- Each Resource record is on a separate line.
- Each field in a Resource record is separated by a comma and is quoted.
- Each line is delimited by a line break (CRLF).
- The first line of the file can specify the name of the container under which the wizard will create a resource. This field is optional. If your file does not contain this field, ensure that you supply an empty string for the wizard to successfully create a resource from the CSV file.
- The second line is a header line and a required field. The wizard allows you to identify whether the file contains a header line. If your file contains a header line, then it must contain the Resource record's field names. The header line field count must correspond to the field count of each line in the file.
- Quotes on numbers are not required.
- A resource record example:

  ```
  "Clinic",Doctor8,en~Doctor8,en~Doctor|it~Dottore|fr~Docteur,,admin.novell|abla
  ke.users.medical-idmsample.novell,admin.novell|ablake.users.medical-
  idmsample.novell,admin.novell|ablake.users.medical-
  idmsample.novell,Standard~50,admin.novell|ablake.users.medical-
  idmsample.novell,MyCustomPrdCN,true
  ```

- Quotes and nested quotes: You can use single quotes within a text field (such as Display name). Use double quotes to enclose a column.

  **NOTE:** For optional fields, the line must include an empty string " " as a placeholder.

## Using the Wizard to Import Roles from a CSV File

1. Open the Provisioning view of the Designer project where you want to import the roles.

   Select the **Resources** node, right-click then select **Import from CSV**.Designer launches the wizard.

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

| Field Name | Description |
|---|---|
| **Role CSV File** | Specify the name and location of the CSV file you want to import. |
| **Ignore header row** | If the file you specify contains a header row, select **Ignore header row** in CSV file. |

**3** Click **Finish**.

The wizard reads the CSV file and adds all the resources that meet the criteria for import. If the wizard encounters an error (see Error Handling for a list of possible errors), the wizard writes the role record to an error file.The wizard creates the error file in the same location as the Role CSV file to import, and it names the file the same name as the Resource CSV file with the _errors appended to the name.

Only the errors identified in Error Handling are severe enough to prevent the wizard from creating the resource. If the wizard encounters other types of errors, it adds the resource, but you must make corrections before the resource can be deployed. For example, if the category specified in the role is not yet added to the directory abstraction layer role category list, the resource can be added, but Designer displays the resource with an informational message.

Resource that are created with errors like this cannot be deployed until the errors are corrected. The Project Checker notifies you of the errors if you attempt to deploy the resource or if you validate the resource objects.

## Error Handling

Table 12-4 describes the cases where a resource cannot be imported. When the wizard encounters these errors, it generates an error file and writes the complete resource record to the file. It maintains the resource original column order except that it inserts a new column as the first column in the record. This column includes the error code. You can modify the associated resource to fix the error directly within the error file, delete the error code column, then specify this error file as input to the wizard.

***Table 12-4*** *CSV Import Wizard Error Codes*

| Error Code | Description |
|---|---|
| RESOURCE_ID_NOT_UNIQUE | A resource with the specified ID already exists. |
| INVALID_ID_NAME | The resource ID contains invalid characters. To fix this problem, edit the name to follow the rules for valid characters: alphabetic characters, digits, underscores, and spaces. |
| INVALID_RESOURCE_CN | The role ID contains invalid characters. |

# A ECMAScript Core Reference

This section provides details on using the ECMA Expression Builder.

## ECMAScript Operators

The following tables provide descriptions of the operators supported by the ECMA Expression Builder.

- Table A-1, “Math,” on page 359
- Table A-2, “Assignment,” on page 359
- Table A-3, “Other,” on page 360
- Table A-4, “Relational,” on page 361
- Table A-5, “Logical,” on page 361
- Table A-6, “String,” on page 361

***Table A-1*** *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 A-2*** *Assignment*

| Operator | Description |
|---|---|
| = Assignment | Assigns the value of the 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. |

| Operator | Description |
|---|---|
| %= 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. |
| >>>= 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 A-3*** *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. |

| Operator | Description |
|---|---|
| >>> 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 A-4*** *Relational*

| Operator | Description |
|---|---|
| == Equal | Assigns the value of the right operand to the left operand. |
| != Not Equal | Returns a Boolean True if 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. |
| <= 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 A-5*** *Logical*

| Operator | Description |
|---|---|
| && AND | Returns a Boolean true if both 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 A-6*** *String*

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

# Functions/Methods

For a description of the functions and methods available in the ECMA Expression Builder, see "User Application API" on page 280.

# DOM Methods

This section lists all DOM-related methods and properties supported by the ECMA Expression Builder, including not only DOM-1 and DOM-2 extensions (defined by the relevant W3C standards), but also Designer's own ECMAScript extensions. Extension methods are specifically noted as such in the text. DOM methods are displayed in the ECMA Expression Builder when you are working with expressions in the Integration activity.

## Node

Lets you work with nodes.

### attributes

W3C DOM Level 1 Node property. This property returns a `NamedNodeMap` object of the attributes for the Node.

### childNodes

W3C DOM Level 1 Node property. This property returns a `NodeList` object consisting of the immediate children of the Node.

### firstChild

W3C DOM Level 1 Node property. This property returns the first child node of a Node object.

### lastChild

W3C DOM Level 1 Node property. This property returns the last child node of a Node object.

### nextSibling

W3C DOM Level 1 Node property. This property returns the next sibling node for a Node object.

### nodeName

W3C DOM Level 1 Node property. This property returns the node name as a String object.

### nodeType

W3C DOM Level 1 Node property. This property returns the node type as a short in with one of the following values:

1 = Element
2 = Attribute
3 = Text
4 = CDATASection
5 = EntityReference
6 = Entity
7 = ProcessingInstruction

8 = Comment
9 = Document
10 = DocumentType
11 = DocumentFragment
12 = Notation

## nodeValue

W3C DOM Level 1 Node property. This property returns the node text data as a String.

## ownerDocument

W3C DOM Level 1 Node property. This property returns a Document object.

## parentNode

W3C DOM Level 1 Node property. This property returns the parent node object for a Node object.

## previousSibling

W3C DOM Level 1 Node property. This property returns the previous sibling node for a Node object.

## XML

Designer extension property. This property returns a string representing the DOM. Useful in Log actions for debugging components (for example, Input.XML).

## appendChild(newChild)

```
Node appendChild(newChild)
```

W3C DOM Level 1 Node method. Appends a node as the last child for a Node. The `newChild` parameter is of type Node.

## cloneNode(deep)

```
Node cloneNode(deep)
```

W3C DOM Level 1 Node method. Creates an unattached Node object. The deep parameter is of type Boolean.

## createXPath(XPathType asPattern)

```
Object createXPath(XPathType asPattern)
```

ECMAScript extension method. Creates the XPath pattern. The `XPath Type asPattern` parameter supports only abbreviated XPath notation and explicit ordinals. XPath functions are not supported.

## hasChildNodes()

```
boolean hasChildNodes()
```

W3C DOM Level 1 Node method. Returns a Boolean indicating whether the node has children.

## insertBefore(newChild, refChild)

```
Node insertBefore(newChild, refChild)
```

W3C DOM Level 1 Node method. Inserts a node object into the parent node before the `refChild` node. The `newChild` parameter is of type Node. The `refChild` parameter is of type Node.

## removeChild(oldChild)

```
Node removeChild(oldChild)
```

W3C DOM Level 1 Node method. Removes a node from a parent and returns an unattached node. The `oldChild` parameter is of type Node.

## replaceChild(newChild, oldChild)

```
Node replaceChild(newChild, oldChild)
```

W3C DOM Level 1 Node method. Replaces one node with another node. The `newChild` parameter is of type Node. The `oldChild` parameter is of type Node.

## getXML()

```
String getXML()
```

ECMAScript extension method. This property returns a string representing the DOM. Useful in Log actions for debugging components. Example:

```
Input.XPath("root/child").getXML()
```

## ownerDocument

W3C DOM Level 2 modified Node property. Returns the Document object associated with this node. This is also the Document object used to create new nodes. Example:

```
someNodeObject.ownerDocument
```

## namespaceURI

W3C DOM Level 2 Node property. Returns the namespace URI of this node, or null if the namespace URI is not specified. Example:

```
someNodeObject.namespaceURI
```

## prefix

W3C DOM Level 2 Node property. Returns the namespace prefix of this node, or null if the namespace prefix is not specified. Example:

```
someNodeObject.prefix
```

## localName

W3C DOM Level 2 Node property. Returns the local part of the qualified name of this node. Example:

```
someNodeObject.localName
```

## normalize()

```
void normalize()
```

W3C DOM Level 2 modified Node method. Puts all Text nodes in the full depth of the subtree underneath this Node, including attribute nodes, into a “normal” form in which only structure separates Text nodes, (for example, elements, comments, processing instructions, CDATA sections, and entity references). In other words, there are neither adjacent Text nodes nor empty Text nodes.

## hasAttributes()

```
boolean hasAttributes()
```

W3C DOM Level 2 Node method. Returns True if the node has any attributes; otherwise, returns False. Example:

```
Temp.XPath("A/B/C").item(0).hasAttributes()
```

## isSupported(feature, version)

```
boolean isSupported(feature, version)
```

W3C DOM Level 2 Node method. Returns True if the specified feature is supported on this node; otherwise, returns False.

***Table A-7*** *Parameters of the IsSupported Method*

| Parameter | Features |
|---|---|
| feature | Core<br>XML<br>HTML<br>Views<br>Stylesheets<br>CSS<br>CSS2<br>Events<br>UIEvents<br>MouseEvents<br>MutationEvents<br>HTMLEvents<br>Range<br>Transversal |
| version | Specifies the version number of the feature to test. In Level 2, version 1, this is the string “2.0”. If the version is not specified, supporting any version of the feature causes the method to return True. |

Example:

```
aNodeObject.isSupported("Core","2.0")
```

# Document

Lets you work with documents.

## doctype

W3C DOM Level 1 Document property. This property returns a `DocumentType` object reflecting the DTD for the document. A Document also has all the properties and methods of Node.

## documentElement

W3C DOM Level 1 Document property. This property returns an Element object (the root element). A Document also has all the properties and methods of Node.

## implementation

W3C DOM Level 1 Document property. This property returns a `DOMImplementation` object. A Document also has all the properties and methods of Node.

## text

Designer extension property. This property returns a concatenated string of all the text nodes (content) under it.

## createAttribute(name)

```
Attr createAttribute(name)
```

W3C DOM Level 1 Document method. Returns an unattached `Attr` object. The `name` parameter is of type String. A Document also has all the properties and methods of Node.

## createCDATASection(data)

```
CDATASection createCDATASection(data)
```

W3C DOM Level 1 Document method. Returns an unattached `CDATASection` object. The `data` parameter is of type String. A Document also has all the properties and methods of Node.

## createComment(data)

```
Comment createComment(data)
```

W3C DOM Level 1 Document method. Returns an unattached Comment object. The data parameter is of type String. A Document also has all the properties and methods of Node.

## createDocumentFragment()

```
DocumentFragment createDocumentFragment()
```

W3C DOM Level 1 Document method. Returns an unattached `DocumentFragment`. A Document also has all the properties and methods of Node.

## createElement(tagName)

```
Element createElement(tagName)
```

W3C DOM Level 1 Document method. Creates an unattached `Element`. The `tagName` parameter is of type String. A Document also has all the properties and methods of Node.

## createEntityReference(name)

```
EntityReference createEntityReference(name)
```

W3C DOM Level 1 Document method. Creates an unattached `EntityReference`. The `name` parameter is of type String. A Document also has all the properties and methods of Node.

## createProcessingInstruction(target,data)

```
ProcessingInstruction createProcessingInstruction(target,data)
```

W3C DOM Level 1 Document method. Returns an unattached `ProcessingInstruction` object. The target and data parameters are of type String. A Document also has all the properties and methods of Node.

## createTextNode(data)

```
Text createTextNode(data)
```

W3C DOM Level 1 Document method. Creates an unattached `Text` object. The data parameter is of type String. A Document also has all the properties and methods of Node.

## getElementsByTagName(tagName)

```
NodeList getElementsByTagName(tagName)
```

W3C DOM Level 1 Document method. Returns a `NodeList` object consisting of the `tagname` element nodes. The `tagName` parameter is of type String. A Document also has all the properties and methods of Node.

## reset()

```
void reset()
```

W3C DOM Level 1 Document method. Clears the document.

## setDTD(Node RootElementName, Object PublicName, Object URL)

```
setDTD(Node RootElementName, Object PublicName, Object URL)
```

ECMAScript extension method. Sets the DTD file for the document.

## setValue(Object aValue)

```
setValue(Object aValue)
```

ECMAScript extension method. Sets the Value of a document from the passed objects. If the passed object is another document, then it copies child nodes (elements and attributes). If the passed object is text, the text is parsed to create a DOM.

## toString()

```
String toString()
```

ECMAScript extension method. Converts a DOM document to an XML formatted string.

Example:

```
Input.XPath("root/child").item(0).toString()
```

## XPath(String asPattern)

```
NodeList XPath(XPathType asPattern)
```

ECMAScript extension method. `XPathTypes` can be of type `NodeList`, `String`, `Number`, or `Boolean`. Usually used to return a `NodeList` matching the XPath pattern. Use brackets to select a particular node from the list. For example, `Input.XPath("INVOICE/LINEITEM[1]")` or `Input.XPath("INVOICE/LINEITEM[last()]")`. Use the `@` symbol to select a node by attribute. For example, `Input.XPath("INVOICE/LINEITEM[@myattr]")` To select by attribute value: `Input.XPath("INVOICE/LINEITEM[@myattr='abc']")`.

## importNode(sourceNode, deep)

```
Node importNode(sourceNode, deep)
```

W3C DOM Level 2 Document method. Imports a node from a document to the current document. Creates a new copy of the `sourceNode`. The `sourceNode` is not altered. A Document also has all the properties and methods of Node.

***Table A-8*** *Parameters for the ImportNode Method*

| Parameter | Description |
|---|---|
| sourceNode | The node to import. |
| deep | A Boolean. If True, recursively import the subtree under the specified node. If False, import only the node itself. |

Example:

```
Temp.importNode(Input.XPath("A/B[2]"), false)
```

## createElementNS(namespaceURI, qualifiedName)

```
Element createElementNS(namespaceURI, qualifiedName)
```

W3C DOM Level 2 Document method. Creates an `Element` of the given `qualifiedName` and `namespaceURI`. A Document also has all the properties and methods of Node.

***Table A-9*** *Parameters for the createElementNS Method*

| Parameter | Description |
|---|---|
| namespaceURI | A string representing the namespace URI that you want to create for the element. |
| qualifiedName | A string representing the name to create for the element. qualifiedName = namespaceprefix + : + localName |

Example:

```
Temp.createElementNS("someURI","nsprefix:PRICE")
```

## createAttributeNS(namespaceURI, qualifiedName)

```
Attr createAttributeNS(namespaceURI, qualifiedName)
```

W3C DOM Level 2 Document method. Creates an `Attribute` of the given `qualifiedName` and `namespaceURI`. A Document also has all the properties and methods of Node.

*Table A-10 Parameters for the createAttributeNS Method*

| Parameter | Description |
|---|---|
| namespaceURI | A string representing the namespace URI that you want to create for the attribute. |
| qualifiedName | A string representing the name to create for the attribute. qualifiedName = namespaceprefix + : + localName |

Example:

```
Temp.createAttributeNS("someURI","nsprefix:PRICE")
```

## getElementsByTagNameNS(namespaceURI, localName)

```
NodeList getElementsByTagNameNS(namespaceURI, localName)
```

W3C DOM Level 2 Document method. Returns a `NodeList` of all the `Elements` with a given `localName` and `namespaceURI`, in the order in which they are encountered in a preorder traversal of the Document tree. A Document also has all the properties and methods of Node.

*Table A-11 Parameters for the getElementsByTagnameNS Method*

| Parameter | Description |
|---|---|
| namespaceURI | A string of the elements on which to match. The special value "*" matches all namespaces. |
| qualifiedName | A string of the elements on which to match. The special value "*" matches all local names. |

Example:

```
Temp.getElementsByTagNameNS("someURI", "someName")
```

## getElementById(elementId)

```
Element getElementById(elementId)
```

W3C DOM Level 2 Document method. Returns the `Element` for which the ID is given by `elementId`. If no such element exists, returns `null`. Behavior is not defined if more than one element has this ID. A Document also has all the properties and methods of Node.

Example;

```
Temp.getElementById("someId")
```

## setSkipNameSpaces(abFlag)

```
void setSkipNameSpaces(boolean flag)
```

Can be used to turn off usage of namespaces and match nodes without any prefixes, behaving like a wildcard match.

### setEncoding(encoding)

```
void setEncoding(String encoding)
```

Sets the character set encoding for the document.

## Element

Lets you work with elements.

### tagName

W3C DOM Level 1 Element property. This property returns a String object containing the element `name`. An Element also has all the properties and methods of Node.

### text

Designer extension property. This property returns the concatenated text of all the text nodes under it.

### booleanValue()

```
boolean booleanValue()
```

ECMAScript extension method. Returns the Boolean value (`True` or `False`) of this object, if possible.

### countOfElement(String propertyName)

```
Number countOfElement(String propertyName)
```

ECMAScript extension method. Returns a count of the named child.

### doubleValue()

```
double doubleValue()
```

ECMAScript extension method. Returns a double value for this object if possible.

### exists(String propertyName)

```
Boolean exists(String propertyName)
```

ECMAScript extension method. Checks for the existence of the named child.

### getAttribute(name)

```
String getAttribute(name)
```

W3C DOM Level 1 Element method. Returns a String consisting of the attribute value. The name parameter is of type String. An Element also has all the properties and methods of Node.

## getAttributeNode(name)

```
Attr getAttributeNode(name)
```

W3C DOM Level 1 Element method. Returns an Attr. The name parameter is of type String. An Element also has all the properties and methods of Node.

## getElementsByTagName(name)

```
NodeList getElementsByTagName(name)
```

W3C DOM Level 1 Element method. Returns a NodeList of all elements with a specified name. The name parameter is of type String. An Element also has all the properties and methods of Node.

## getIndex()

```
int getIndex()
```

ECMAScript extension method. Returns the current index.

## getParent()

```
Node getParent()
```

ECMAScript extension method. Returns the parent element.

## normalize()

```
void normalize()
```

W3C DOM Level 1 Element method. Returns a void. An Element also has all the properties and methods of Node.

## removeAttribute(name)

```
void removeAttribute(name)
```

W3C DOM Level 1 Element method. Removes an attribute from an element. The `name` parameter is of type String. An Element also has all the properties and methods of Node.

## removeAttributeNode(oldAttr)

```
Attr removeAttributeNode(oldAttr)
```

W3C DOM Level 1 Element method. Removes an attribute from an element and returns an unattached Attr. The oldAttr parameter is of type Attr. An Element also has all the properties and methods of Node.

## setAttribute(name,value)

```
void setAttribute(name, value)
```

W3C DOM Level 1 Element method. Sets the value of an attribute node for an element. The name parameter is of type String. The value parameter is of type String. An Element also has all the properties and methods of Node.

## setAttributeNode(newAttr)

```
Attr setAttributeNode(newAttr)
```

W3C DOM Level 1 Element method. Attaches an attribute node to an element. The newAttr parameter is of type Attr. An Element also has all the properties and methods of Node.

## setIndex(int aiIndex)

```
setIndex(int aiIndex)
```

ECMAScript extension method. Sets the iterator index value for this element.

## setText(String asText)

```
setText(String asText)
```

ECMAScript extension method. Sets the text node associated with this element.

## setValue(Object aValue)

```
setValue(Object aValue)
```

ECMAScript extension method. Sets the value of an element from the passed object. If the passed object is another element, then it also copies child nodes (elements and attributes).

## toNumber()

```
Number toNumber()
```

ECMAScript extension method. Gets the text node and converts it to a number.

## toString()

```
String toString()
```

ECMAScript extension method. Gets the text node associated with this element.

## XPath(XPathType asPattern)

```
NodeList XPath(XPathType asPattern)
```

ECMAScript extension method. The XPathType parameter can be of type NodeList, String, Number, or Boolean. Usually used to return a Nodelist matching the XPath pattern. Use brackets to select a particular node from the list. For example, `Input.XPath("INVOICE/LINEITEM[1]")` or `Input.XPath("INVOICE/LINEITEM[last()]")`. Use the `@` symbol to select a node by attribute. For example, `Input.XPath("INVOICE/LINEITEM[@myattr]")`. To select by attribute value: `Input.XPath("INVOICE/LINEITEM[@myattr='abc']")`.

## getAttributeNS(namespaceURI, localName)

```
string getAttributeNS(namespaceURI, localName)
```

W3C DOM Level 2 Element method. Returns the Attr value as a string. An Element also has all the properties and methods of Node.

***Table A-12*** *Parameters for the getAttributeNS Method*

| Parameter | Description |
|---|---|
| namespaceURI | Specifies a string representing the namespace URI of the target Attr. |
| localName | Specifies a string of the localName of the target Attr. |

Example:

```
Temp.XPath("A/B[0]").getAttributeNS("someURI", "someAttr")
```

## setAttributeNS(namespaceURI, qualifiedName, value)

```
void setAttributeNS(namespaceURI, qualifiedName, value)
```

W3C DOM Level 2 Element method. Adds a new attribute. If an attribute with the same namespaceURI and localName is already present in the element, its prefix is changed to be the prefix part of the qualifiedName parameter, and its value is changed to be the value parameter. An Element also has all the properties and methods of Node.

***Table A-13*** *Parameters for the setAttributeNS Method*

| Parameter | Description |
|---|---|
| namespaceURI | The namespace URI of the attribute to create or alter. |
| qualifiedName | Specifies the qualified name of the attribute to create or alter.<br>**TIP:** qualifiedName = namespaceprefix + : + localName |
| value | Specifies the value to set in string form. |

Example:

```
Temp.XPath("A/B[0]").setAttributeNS("someURI", "someAttrName", "someAttrvalue")
```

## removeAttributeNS(namespaceURI, localName)

```
void removeAttributeNS(namespaceURI,localName)
```

W3C DOM Level 2 Element method. Removes an attribute by local name and namespace URI. If the removed attribute has a default value, it is immediately replaced. The replacing attribute has the same namespace URI and local name, as well as the original prefix. An Element also has all the properties and methods of Node.

***Table A-14*** *Parameters for the removeAttributeNS Method*

| Parameter | Description |
|---|---|
| namespaceURI | Specifies the namespaceURI of the attribute to remove. |
| localName | Specifies the name of the attribute to remove. |

Example:

```
Temp.XPath("A/B[0]").removeAttributeNS("someURI", "someAttrName")
```

## getAttributeNodeNS(namespaceURI, localName)

```
Attr getAttributeNodeNS(namespaceURI, localName)
```

W3C DOM Level 2 Element method. Retrieves an attribute node by local name and namespace URI. An Element also has all the properties and methods of Node.

***Table A-15*** *Parameters for the getAttributeNodeNS Method*

| Parameter | Description |
|---|---|
| namespaceURI | Specifies the namespaceURI of the attribute to retrieve. |
| localName | Specifies the name of the attribute to retrieve. |

Example:

```
Temp.XPath("A/B[0]").getAttributeNodeNS("someURI", "someAttr"
```

## setAttributeNodeNS(newAttr)

```
Attr setAttributeNodeNS(newAttr)
```

W3C DOM Level 2 Element method. Adds a new attribute. If an attribute with the same local name and namespace URI is already present in the element, it is replaced by the new attribute. If the newAttr attribute replaces an existing attribute with the same local name and namespace URI, the replaced Attr node is returned, otherwise null is returned. The newAttr parameter is a new attribute object. An Element also has all the properties and methods of Node.

Example:

```
Temp.XPath("A/B[0]").setAttributeNodeNS(newAttr)
```

## getElementsByTagNameNS(namespaceURI, localName)

```
NodeList getElementsByTagNameNS(namespaceURI, localName)
```

W3C DOM Level 2 Element method. Returns a NodeList of all the descendant Elements with a given local name and namespace URI in the order in which they are encountered in a preorder traversal of this Element tree. An Element also has all the properties and methods of Node.

***Table A-16*** *Parameters for the getElementsByTagNameNS Method*

| Parameter | Description |
|---|---|
| namespaceURI | Specifies the namespaceURI of the elements on which to match. The special value “*” matches all namespaces. |
| localName | Specifies the localName of the elements on which to match. The special value “*” matches all local names. |

Example:

```
Temp.XPath("A/B[0]").getElementsByTagNameNS("someURI", "someName")
```

## hasAttribute(name)

```
boolean hasAttribute()
```

W3C DOM Level 2 Element method. Returns True when an attribute with a given name is specified for this element or has a default value. Otherwise, returns False. The parameter name is a string that specifies the attribute name for which to look. An Element also has all the properties and methods of Node.

Example:

```
Temp.XPath("A/B[0]").hasAttribute("someName")
```

## hasAttributeNS(namespaceURI, localName)

```
boolean hasAttributeNS(namespaceURI, localName)
```

W3C DOM Level 2 Element method. Returns True when an attribute with a given local name and namespace URI is specified on this element or has a default value. Otherwise, returns False. An Element also has all the properties and methods of Node.

***Table A-17*** *Parameters for the hasAttributeNS Method*

| Parameter | Description |
|---|---|
| namespaceURI | Specifies the namespaceURI of the attribute for which to look. |
| localName | Specifies the localName of the attribute for which to look. |

Example:

```
Temp.XPath("A/B[0]").hasAttributeNS("someURI", "someName")
```

# Attribute

Lets you work with attributes.

## name

W3C DOM Level 1 attribute property. This property returns a String object indicating the tag name of the attribute. An attribute also has all the properties and methods of Node.

### specified

W3C DOM Level 1 Attr property. This property returns a Boolean. An attribute also has all the properties and methods of Node.

### text

Designer extension property. This property returns the text value of the attribute.

### value

W3C DOM Level 1 Attr property. This property returns a String object representing the text value of the attribute. An attribute also has all the properties and methods of Node.

### setValue(Object aValue)

```
setValue(Object aValue)
```

Designer extension method. Sets the value of an attribute from the passed object.

### toString()

```
String toString()
```

ECMAScript extension method. Gets the text node associated with the attribute.

### ownerElement

W3C DOM Level 2 Attr property. Returns the Element node to which this attribute is attached. Returns null if this attribute is not in use. An Attr also has all the properties and methods of Node.

Example:

```
attributeObject.ownerElement
```

## CharacterData

Lets you work with character data.

### data

W3C DOM Level 1 CharacterData property. This property is of type String and represents the contents of the CharacterData object. CharacterData also has all the properties and methods of Node.

### length

W3C DOM Level 1 CharacterData property. This property represents the length of the CharacterData object. CharacterData also has all the properties and methods of Node.

### appendData(arg)

```
void appendData(arg)
```

W3C DOM Level 1 CharacterData method. Appends text to the CharacterData object. The arg parameter is of type String. CharacterData also has all the properties and methods of Node.

### insertData(offset, arg)

```
void insertData(offset, arg)
```

W3C DOM Level 1 CharacterData method. Inserts text in the CharacterData object. The offset parameter is of type unsigned long. The `arg` parameter is of type String. CharacterData also has all the properties and methods of Node.

### deleteData(offset, count)

```
void deleteData(offset, count)
```

W3C DOM Level 1 CharacterData method. Deletes text in the CharacterData object. The offset and `count` parameters are of type unsigned long. CharacterData also has all the properties and methods of Node.

### replaceData(offset, count, arg)

```
void replaceData(offset, count, arg)
```

W3C DOM Level 1 CharacterData method. Replaces text in the CharacterData object. The offset and count parameters are of type unsigned long. The `arg` parameter is of type String. CharacterData also has all the properties and methods of Node.

### substringData(offset, count)

```
String substringData(offset, count)
```

W3C DOM Level 1 CharacterData method. Returns a substring of the CharacterData object. The offset and count parameters are of type unsigned long. CharacterData also has all the properties and methods of Node.

## NodeList

Lets you work with node lists.

### length

W3C DOM Level 1 NodeList property. This property returns the number of nodes in a NodeList object.

### avg('[NodeList]')

```
Number avg('[NodeList]')
```

ECMAScript aggregate extension method. Returns a number equal to the average value in the NodeList. The NodeList parameter is of type XPath. If no parameter is supplied, then the current NodeList/GroupName is used. The function argument should be in single quotes, and must be escaped for nested calls.

Example:

```
Input.XPath("rootElem/childElem").avg()
```

## count('[NodeList]')

```
Number count('[NodeList]')
```

ECMAScript aggregate extension method. Returns a number equal to a count of the nodes in the NodeList that have data. Nodes without data, or nodes with only child elements are not counted. To count all nodes, use the .length property on a nodeList object. The optional NodeList parameter is of type XPath. If no parameter is supplied (the usual case), then the current NodeList/GroupName is used. The function argument should be in single quotes, and must be escaped for nested calls.

Example:

```
Input.XPath("rootElem/childElem").count()
```

## item(index)

```
Node item(index)
```

W3C DOM Level 1 NodeList method. Returns the indicated Node from the NodeList. The index parameter is of type unsigned long. The Index is 0-based.

## min('[NodeList]')

```
Number min('[NodeList]')
```

ECMAScript aggregate extension method. Returns a number equal to the lowest value in the NodeList. The NodeList parameter is of type XPath. If no parameter is supplied, then the current NodeList/GroupName is used. The function argument should be in single quotes, and must be escaped for nested calls.

Example:

```
Input.XPath("rootElem/childElem").min()
```

## max(['NodeList]')

```
Number max('[NodeList]')
```

ECMAScript aggregate extension method. Returns a number equal to the highest value in the NodeList. The NodeList parameter of type XPath. If no parameter is supplied, then the current NodeList/GroupName is used. The function argument should be in single quotes, and must be escaped for nested calls.

Example:

```
Input.XPath("rootElem/childElem").max()
```

## sum('[NodeList]')

```
Number sum('[NodeList]')
```

ECMAScript aggregate extension method. Returns a number equal to the sum of the values in NodeList. The NodeList parameter is of type XPath. If no parameter is supplied, then the current NodeList/GroupName is used. The function argument should be in single quotes, and must be escaped for nested calls.

Example:

```
Input.XPath("rootElem/childElem").sum()
```

## where(XPathType asPattern)

```
NodeList where(String asPattern)
```

ECMAScript extension method. Gets a NodeList of nodes matching the XPath pattern.

## toNumber()

```
toNumber()
```

Converts the data of the first instance in the NodeList to an ECMAScript Number object. Any alphabetic characters or embedded spaces in data return NaN. Leading and trailing spaces are permitted.

Example:

```
var myNum = Input.XPath("Invoice/Amount").toNumber()
```

# NamedNodeMap

Lets you work with named node maps.

## length

length W3C DOM Level 1 NamedNodeMap property. This property returns the number of nodes in a NamedNodeMap.

## getNamedItem(name)

```
Node getNamedItem(name)
```

W3C DOM Level 1 NamedNodeMap method. Returns all selected Nodes of the indicated name. The name parameter is of type String.

## getNamedItemNS(namespaceURI, localName)

```
Node getNamedItemNS(namespaceURI, localName)
```

W3C DOM Level 2 NamedNodeMap method. Returns a node specified by local name and namespace URI.

*Table A-18 Parameters for the NamedNodeMap Method*

| Parameter | Description |
|---|---|
| namespaceURI | Specifies the namespaceURI of the node to retrieve. |
| localName | Specifies the localName of the node to retrieve. |

Example:

```
Temp.XPath("A/B").item(0).getAttributes() .getNamedItemNS("someURI", "anAttrName")
```

## item(index)

```
Node item(index)
```

W3C DOM Level 1 NamedNodeMap method. Returns the indicated Node from the NamedNodeMap. The index parameter is of type unsigned long. The index is 0-based.

## removeNamedItem(name)

```
Node removeNamedItem(name)
```

W3C DOM Level 1 NamedNodeMap method. Removes the indicated node from the NamedNodeMap and returns an unattached node. The name parameter is of type String.

## removeNamedItemNS(namespaceURI, localName)

```
Node removeNamedItemNS(namespaceURI, localName)
```

W3C DOM Level 2 NamedNodeMap method. Removes and returns the node specified by namespace URI and local name.

*Table A-19 Parameters for the removeNamedItemNS Method*

| Parameter | Description |
|---|---|
| namespaceURI | Specifies the namespaceURI of the node to remove. |
| localName | Specifies the localName of the node to remove. |

Example:

```
Temp.XPath("A/B").item(0).getAttributes() .removeNamedItemNS("someURI",
"anAttrName")
```

## setNamedItem(arg)

```
Node setNamedItem(arg)
```

W3C DOM Level 1 NamedNodeMap method. Returns a Node. The arg parameter is of type Node.

## setNamedItemNS(Node arg)

```
Node setNamedItemNS(arg)
```

W3C DOM Level 2 NamedNodeMap method. If the new Node replaces an existing node, the replaced Node is returned, otherwise null is returned.

Example:

```
var item = Temp.XPath("A/B").item(0);
```

```
item.getAttributes().setNamedItemNS(aNodeObject)
```

# Text

Lets you work with text.

## splitText(offset)

```
Text splitText(offset)
```

W3C DOM Level 1 Element method. Removes the text up to the offset and creates an unattached text node with the removed text. The offset parameter is of type unsigned long. A Text also has all the properties and methods of CharacterData.

# DocumentType

Lets you work with document types.

## name

W3C DOM Level 1 DocumentType property. This property returns a String representing the document type name.

## entities

W3C DOM Level 1 DocumentType property. This property returns a NamedNodeMap of the entities defined in the document.

## internalSubset

W3C DOM Level 2 DocumentType property. This property returns a String representing the internal subset as a string.

## notations

W3C DOM Level 1 DocumentType property. This property returns a NamedNodeMap of the notations defined in the document.

## publicId

W3C DOM Level 2 DocumentType property. This property returns a String representing the public identifier of the external subset.

### systemId

W3C DOM Level 2 DocumentType property. This property returns a String representing the system identifier of the external subset.

## DOMImplementation

Lets you work with DOM implementations.

### createDocument(namespaceURI, qualifiedName, doctype)

```
Document createDocument(namespaceURI, qualifiedName, doctype)
```

W3C DOM Level 2 DOMImplementation method. Creates an XML Document object of the specified type with its document element.

***Table A-20*** *Parameters for the DOMImplementation Method*

| Parameter | Description |
|---|---|
| namespaceURI | Specifies the namespaceURI of the document element to create. |
| qualifiedName | Specifies the qualified name of the document element to create.<br>qualifiedName = namespaceprefix + : + localName |
| doctypei | Specifies the type of document to create, or null. |

### createDocumentType(qualifiedName, publicID, systemID)

```
DocumentType createDocumentType(qualifiedName, publicID, systemID)
```

W3C DOM Level 2 DOMImplementation method. Creates an empty DocumentType node. Parameters: qualifiedName is a string of the name of the document type to create. publicID is the external subset public identifier. systemID is the external subset system identifier. Note: qualifiedName = namespaceprefix + : + localName

***Table A-21*** *Parameters for the createDocumentType Method*

| Parameter | Description |
|---|---|
| qualifiedName | Specifies the qualified name of the document element to create.<br>qualifiedName = namespaceprefix + : + localName |
| publicID | Specifies the external subset public identifier. |
| systemID | Specifies the external subset system identifier. |

### hasFeature(feature, version)

```
boolean hasFeature(feature, version)
```

W3C DOM Level 1 DOMImplementation method. Returns a Boolean. The feature parameter is of type String. The version parameter is of type String.

# Notation

Lets you work with notation.

## publicId

W3C DOM Level 2 This property returns a String representing the public identifier of the external subset.

## systemId

W3C DOM Level 2 property. This property returns a String representing the system identifier of the external subset.

# Entity

Lets you work with entities.

## publicId

W3C DOM Level 2 property. This property returns a String representing the public identifier of the external subset.

## systemId

W3C DOM Level 2 property. This property returns a String representing the system identifier of the external subset.

## notationName

W3C DOM Level 1 Entity property. This property is of type String. An Entity also has all the properties and methods of Node.

# ProcessingInstruction

Lets you work with processing instructions.

## target

W3C DOM Level 1 ProcessingInstruction property. This property is a String representation of the target part of a Processing Instruction.

## data

W3C DOM Level 1 ProcessingInstruction property. This property is a String representation of the data part of a Processing Instruction.

# ECMAScript Core

This section lists all ECMAScript core methods and properties supported by the ECMA Expression Builder.

## 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.

# 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 either 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.

# 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 seven parameters, in the following format: new Date(year,month,date,hrs,mins,secs,ms). This date must be a java.util.Date object and not an ECMAScript Date object if you intend to use it with the Identity Manager User Application workflow system.

## 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 it must be used to obtain a year from a 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 number which is a 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 existing 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 existing 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()
```

Can accept a 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.

## 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 parameters.

## 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.

# 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 (Not a Number), +*, 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 393) 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.

# 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 a positive sign. The input value x can be any number value.

Example:

```
Math.abs(-123.23940) = 123.23940
```

## acos(x)

```
Number acos(x)
```

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)
```

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)
```

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)
```

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)
```

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)
```

Returns an implementation-dependent approximation to the cosine of the argument. The argument must be expressed in radians.

## exp(x)

```
Number exp(x)
```

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)
```

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)
```

Returns an implementation-dependent approximation to the 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)
```

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)
```

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)
```

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()
```

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)
```

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)
```

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)
```

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)
```

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.

# 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 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.

## 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.

## 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.