Project and Portfolio Management Center

Software Version: 9.40

Deployment Management Configuration Guide

Document Release Date: September 2016
Software Release Date: September 2016

Hewlett Packard Enterprise

Legal Notices

Documentation Updates

To check for recent updates or to verify that you are using the most recent edition of a document, go to: https://softwaresupport.hpe.com/.

This site requires that you register for an HP Passport and to sign in. To register for an HP Passport ID, click **Register** on the HPE Software Support site or click **Create an Account** on the HP Passport login page.

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HPE sales representative for details.

The following table indicates changes made to this document since the last released edition.

Support

Visit the HPE Software Support site at: https://softwaresupport.hpe.com/.

Most of the support areas require that you register as an HP Passport user and to sign in. Many also require a support contract. To register for an HP Passport ID, click **Register** on the HPE Support site or click **Create an Account** on the HP Passport login page.

To find more information about access levels, go to: https://softwaresupport.hpe.com/web/softwaresupport/access-levels.

HPE Software Solutions Now accesses the HPSW Solution and Integration Portal website. This site enables you to explore HPE Product Solutions to meet your business needs, includes a full list of Integrations between HPE Products, as well as a listing of ITIL Processes. The URL for this website is https://softwaresupport.hpe.com/km/KM01702731.

Contents

Chapter 1: Getting Started with Deployment Management Configuration 11

Introduction to Deployment Management 11

Deployment Management Concepts 11

Overview of a Simplified Deployment Process 13

Overview of Configuring Deployment Management 14

Related Information 15

Chapter 2: Gathering Process Requirements 17

Overview of Gathering Process Requirements 17

Defining Workflows 18

Gathering Information for Workflow Steps 19

Gathering Information for Decision Steps 19

Gathering Information for Execution Steps 20

Gathering Information for Condition Steps 22

Gathering Information for Subworkflow Steps 23

General Workflow Design Guidelines 24

Defining Object Types 26

Determining Commands Needed for Objects 28

Object Type Checklist 28

Defining Environments 30

Environments Checklist 30

Defining Notification Templates 32

Notification Templates Checklist 32

Defining User Data Fields 32

User Data Checklist 33

Defining Security and Access 33

Security and User Access Checklist 34

Chapter 3: Configuring Workflows 36

Overview of Workflows 36

Mapping Workflows 39

Opening Workflow Workbench 44

Creating Workflows 44
- Configuring General Information for a Workflow 44
- Choosing Workflow Steps 45
  - Overview of Decision Workflow Steps 45
  - Overview of Condition Workflow Steps 45
  - Overview of Execution Workflow Steps 47
  - Overview of Subworkflow Workflow Steps 48
- Adding Steps to a Workflow 48
- Adding Close Step 49
  - Configuring Reopen Workflow Steps 50
- Adjusting Workflow Step Sequences 50
- Verifying and Enabling Workflows 51

Configuring Workflow Steps 51
- Configuring Properties of a Workflow Step 52
- Configuring Security for Workflow Steps 53
- Configuring Notifications for Workflow Steps 54
  - Configuring Setup Tabs 56
    - Sending Notifications When Workflow Steps Become Eligible 56
    - Sending Notifications when Workflow Steps Have Specific Results 57
    - Sending Notifications When Workflow Steps Result in Specific Errors 58
    - Specifying the Time Notifications Are Sent 60
    - Sending Follow-Up Notifications (Reminders) 61
    - Configuring Notification Recipients 62
  - Configuring Message Content 64
    - Using Tokens in the Message Body 65
    - Including URLs (Smart URLs) 65
- Configuring Timeouts for Workflow Steps 66
- Configuring Transitions for Workflow Steps 67
  - Adding Transitions Based on Specific Results 68
  - Adding Transitions Not Based on Specific Results 68
    - Adding Transitions Based on Data in Tables 69
    - Adding Transitions Based on All But One Specific Value 69
    - Adding Transitions Based on All Results 70
    - Adding Transitions Based on Errors 70

Adding Transitions Back to Same Step 71
Adding Transitions Based on Previous Workflow Step Results 74
Adding Transitions to and Removing Them from Subworkflows 76
Configuring Validations for Workflow Steps 77
Validations and Execution Type Relationships 78
Adding Color to Workflow Steps 79
Configuring Segregation of Duties for Workflow Steps 80
Integrating Object Type Commands and Workflows 81
Integrating Environments and Workflows 81
Choosing Source Environments Based on Application Code 82
Integrating Request and Package Workflows 83
Step 1. Setting Up WF - Jump/Receive Step Label Validations 85
Step 2. Generating Jump Step Sources 85
Step 3. Generating Receive Step Sources 86
Step 4. Including Jump and Receive Workflow Steps in Workflows 88
Chapter 4: Configuring Workflow Components 90
Overview of Workflow Step Sources 90
Configuring and Using Workflow Step Source Restrictions 91
Opening Workflow Workbench 91
Overview of Creating Workflow Step Sources 92
Configuring Ownership of Workflow Step Sources 93
Creating Decision Workflow Step Sources 93
Creating Execution Workflow Step Sources 96
Setting Up Execution Steps 100
Defining Execution Types 100
Executing Object Type Commands 101
Closing Packages as Successful 101
Closing Packages as Failed 102
Marking Packages Ready for Release 103
Executing PL/SQL Functions and Creating Transitions Based on Results 104
Executing SQL Statements and Creating Transitions Based on Results 105
Evaluating Tokens and Creating Transitions Based on Results 106
Executing Multiple System Level Commands 107

Creating Subworkflow Workflow Step Sources 108
Subworkflows Returning to Deployment Management Workflows 108
Using Workflow Parameters 109
Creating Workflow Parameters 109
Example: Building a Loop Counter Using Workflow Parameters 110
Modifying Workflows Already In Use 111
Performance Considerations 112
Copying and Testing Trial Versions of Workflows 113
Modifying Production Workflows 113
Disabling Workflow Steps 114
Redirecting Workflows 114
Moving Requests or Packages Out of Steps 115
Chapter 5: Configuring Object Types 116
Overview of Object Types 116
Opening Object Type Workbench 119
Configuring General Information for Object Types 119
Creating Object Type Fields 120
Overview of Object Type Field Validations 121
Selecting Validations 122
Creating Object Type Fields 123
Configuring Field Dependencies 125
Copying Object Type Fields 127
Editing Object Type Fields 128
Removing Fields 129
Configuring Layouts for Object Types 129
Changing Field Widths 129
Moving Fields 130
Setting Object Names 131
Setting Object Revisions 131
Configuring Commands for Object Types 132
Adding Commands to Object Types 132
Editing Commands of Object Types 133
Copying Commands in Object Types 134
Deleting Commands in Object Types 134
Command Conditions 135

Configuring Ownership for Object Types 135
Adding Ownerships to Object Types 136
Deleting Ownerships from Object Types 136
Using Commands to Change Field Values 137
Chapter 6: Configuring Releases and Distributions 138
Overview of Releases and Distributions 138
Workflow Scope 138
Release Management and Package Workflows 139
Release Distribution Workflows 140
Package Level Subworkflows 141
Dependencies and Run Groups 143
Opening Releases 145
Submitting Releases 145
Overview of Using Release Management - Process 145
Release Management Preconfiguration 145
Creating Releases 146
Processing Releases 146
Distributions 147
Overview of Configuring Releases 148
Opening Release Workbench 148
Creating Releases 149
Adding Packages to Releases 150
Adding Packages Through Release Window 150
Adding Packages Through Package Window 151
Adding Packages by Ready for Release Workflow Step 151
Adding Packages from Requests 152
Adding Requests to Releases 152
Adding Requests Through Release Window 152
Adding Requests Through Requests Window 153
Verifying Releases 153
Creating Distributions 154
Enabling or Disabling Package Lines in a Distribution 155
Running Distributions through a Workflow 156
Processing Distribution Steps 156
Processing Package Lines 156

Completing Distributions 157
Chapter 7: Configuring Environments 158
Overview of Environments 158
Environment Connection Protocols 158
Configuring to Use the SSH2 Client 159
Enabling Logging for SSH2 Library 160
Adding Support for Ciphers of Unlimited Strength 160
SSH2 Client FAQ 161
Environment Transfer Protocols 162
Transfer Protocol Configuration Notes 162
Selecting FTP Protocol 162
Overview of Configuring Environments 165
Opening Environments Workbench 166
Configuring General Information for Environments 166
Creating Environments 167
Using Application Code Environments 171
Copying Application Codes from Other Environments 173
Setting Ownership and Participants for Environments 174
Adding Ownerships to Environments 174
Deleting Ownerships from Environments 175
Adding Participants to Environments 175
Deleting Participants from Environments 176
Environment Maintenance and Utilities 176
Testing Environment Setups 176
Mass Updates of Base Paths 177
Environment Password Management Utility 178
Overview of Environment Groups 179
Overview of Configuring Environment Groups 180
Opening Environment Group Workbench 181
Configuring General Information for Environment Groups 181
Creating Environment Groups 182
Setting Order of Executions 182
Setting Ownership and Participants for Environment Groups 183
Adding Ownerships to Environment Groups 183
Deleting Ownerships from Environment Groups 184

Adding Participants to Environment Groups 184
Deleting Participants from Environment Groups 185
Overview of Environment Refresh 185
Overview of Configuring Environment Refresh 187
Opening Environment Refresh Workbench 188
Configuring General Information for Environment Refreshes 188
Configuring a Workflow with an Environment Refresh 190
Configuring a Package with an Environment Refresh 191
Chapter 8: Configuring Notification Templates 193
Overview of Notification Templates 193
Opening Notification Template Workbench 193
Deleting Notification Templates 194
Creating Notification Templates 194
Configuring Ownership of Notification Templates 196
Deleting Ownerships from Notification Templates 197
Configuring Notification Intervals 197
Checking Usage of Notification Templates 198
Chapter 9: Configuring User Data 200
Overview of User Data 200
Referencing User Data 201
Migrating User Data 201
User Data Configuration Tasks 202
Opening User Data Workbench 202
Viewing General Information for User Data Types 202
Creating a User Data Context 203
Creating User Data Fields 204
Copying a Field Definition 207
Editing User Data Fields 208
Configuring User Data Field Dependencies 209
Removing Fields 211
Configuring User Data Layouts 211
Changing Column Widths 212
Moving Fields 213
Swapping Positions of Two Fields 213

Previewing Layout 214

Appendix A: Worksheets 215

Configuration Workflow Worksheets 215

Execution Workflow Step Worksheets 216

Decision Workflow Step Worksheets 218

Subworkflow Workflow Step Worksheets 219

Object Type Configuration Sheets 221

Example of Completed Object Type Configuration Sheets 223

Send documentation feedback 227

Chapter 1: Getting Started with Deployment Management Configuration

- "Introduction to Deployment Management" below
- "Deployment Management Concepts " below
- "Overview of Configuring Deployment Management" on page 14
- "Related Information" on page 15

Introduction to Deployment Management

Deployment Management is a Project and Portfolio Management Center (PPM) application that automates and manages the migration and deployment of changes for packaged applications, custom applications, legacy systems, web content, and more. Leveraging and enforcing best practice deployment processes, Deployment Management performs all tasks required to install software changes correctly across your system landscape. With Deployment Management, packages are deployed automatically, eliminating errors inherent with manual processesDeployment Management's detailed audit trail helps you pinpoint problems quickly; roll back changes if necessary; and support regulatory compliance requirements, such as segregation of duties (SOD), at both the role level and the process step level.

Deployment Management allows developers to attach changes as package line items within a package. Each package line uses an object type to tell the system the type of a change. Packages then follow a digitized process to ensure they are appropriately handled with reviews and approvals. Based on the process, each package line (change) can then be deployed to the environments defined, such as development, test, and production.

This document provides the details on how to configure an Deployment Management system using the PPM Workbench, and how to make sure your packages follow your digitized process. "Getting Started with Deployment Management Configuration" above provides an overview of how to configure Deployment Management to support your business processes.

Deployment Management Concepts

The following are the major concepts of Deployment Management.

- **Deployment**: Moving a file, script, code, or full application (object) between two or more instances. For example, a file can be deployed from a development instance to a testing instance and finally into one or more production instances. Deployment typically involves connecting systems together, moving files between the systems, and running support scripts.
- **Environments**: Can consist of a server, a single database instance, or an associated remote client machine. Not all of these components need be present in an environment. For example, it is possible to have an environment which does not contain a database.
- **Environment Groups**: Define a set of environments which can be referenced as the source and destinations for object migrations and deployments.
- **Environment Refresh**: Replaces one environment with another environment. After the physical refresh, data in Deployment Management is updated to be consistent with the refreshed environment.
- **Object Types**: Define the technical steps required to migrate or deploy application files or changes for packaged applications, custom applications, legacy systems, web content. For example, a File Migration object type might contain the information and commands required to transfer a file from one machine to another, while an SQL script object type might address the migration and execution of database scripts.
- **Notification Templates**: Preconfigured notification forms that can be selected and used with the various Deployment Management entities, such as workflows and packages.
- **Package**: A set of objects (package lines) being migrated or deployed together. The package follows an assigned workflow through the various review, approval, and migration steps. Each object in a package is defined by a separate package line. While each package line is acted upon separately, the group of package lines (the package) represent a logical unit that should be moved and tracked together.
- **Package Lines**: A package line is one object being migrated or deployed within the package. Each package line follows the assigned workflow through the various review, approval, and migration steps.
- **Physical Refresh**: A physical refresh replaces one environment with a physical copy of another environment.
- **Refresh Group**: Reflects the environment being refreshed and the environment providing the new environment.
- **Release and Release Distribution**: Release Management introduces repeatable, reliable processes surrounding software and application releases. Deployment Management provides an interface for grouping and processing the packages and requests associated with a specific

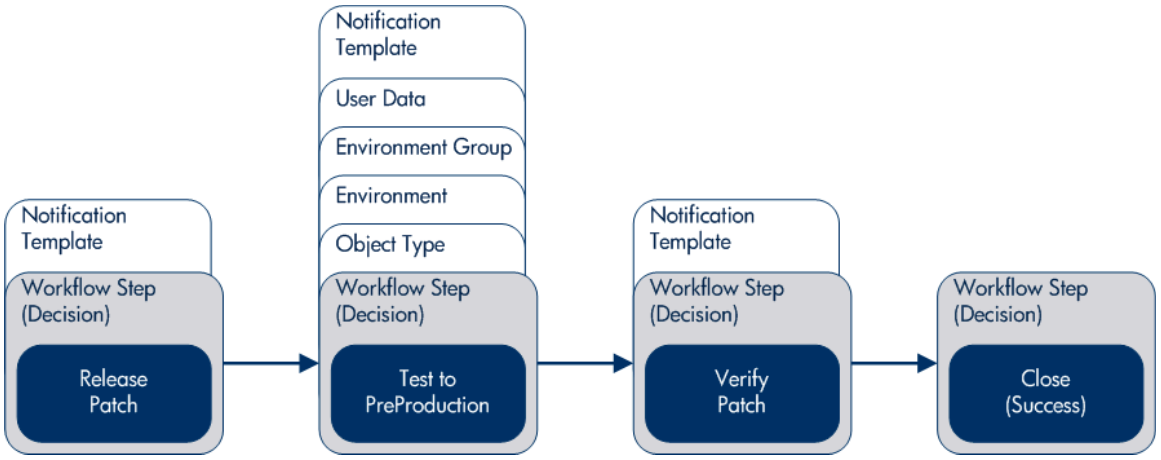
release. Groups of related packages can then be activated from a single window.

- **User Data**: Deployment Management entities, such as packages and workflows, include a set of standard fields that provide information concerning those entities. While these standard fields are normally sufficient for day-to-day processing, you can add "user data" fields to capture additional information specific to your business process.
- **Workflow**: A digitized process where a logical series of steps define business process. Workflow steps can range in usage from review and approvals to performing migrations and deployments.

Overview of a Simplified Deployment Process

"Figure 1-1. PPM Center components" below shows a simple four-step deployment process for a patch release.

Figure 1-1. PPM Center components



The first step, **Release Patch,** is a decision step where a user is notified that a decision needs to be made, in this case, should the patch be released. After manually approved, the process moves to the second step.

The second step, **Test to PreProduction,** is an execution step where the patch (object) is automatically deployed from one environment to another environment. As part of this second step, the patch is deployed and a user is notified that the deployment has been completed. After the execution step is complete, the process moves to the third step.

The third step, **Verify Patch,** is a decision step where a user is notified to check and verify that the patch successfully deployed from one environment to another environment. After verified and manually approved, the process moves to the fourth and final step.

The fourth step, **Close (Success),** is an execution step that automatically closes the deployment process and automatically notifies users of the successful closure of the patch release.

Overview of Configuring Deployment Management

Deployment Management can be configured using the following steps.

- Step 1: Gather process requirements.

  You should collect specific information concerning your process, the types of objects to be deployed, source and destination environments, and so on.

  For detailed information, see "Gathering Process Requirements" on page 17. For worksheets to help gather the information required to build a deployment system, see "Worksheets" on page 215.

- Step 2: Configure workflows.

  Configuring workflows involves setting up the required workflow steps (decision and execution), adding transitions between the steps, and configuring each workflow step for notifications, security groups, segregation of duties, and so on. For information on how to configure workflows, see "Configuring Workflows" on page 36 and "Configuring Workflow Components" on page 90. For configuration requirements for Deployment Management's release process, see "Configuring Releases and Distributions" on page 138. For worksheets to help gather the information required to configure a workflow, see "Worksheets" on page 215.

- Step 3: Configure object types.

  Object types define the technical step or steps required to "copy" a change from one environment to another environment. For information on how to configure object types, see "Configuring Object Types" on page 116. For worksheets to help gather the information required to configure an object type, see "Worksheets" on page 215.

- Step 4: Configure environments.

  Configuring environments requires defining the server and database instances identified by your business or deployment process. For details about how to configure environments, environment groups, and environment refreshes, see "Configuring Environments" on page 158.

- Step 5: Configure notification templates.

  Notification templates are preconfigured notification "formats" used by other Deployment Management entities, such as workflows, object types, and environments. "Configuring

Notification Templates" on page 193"Configuring Notification Templates" on page 193 provides information on how to configure notification templates.

- Step 6: Configure user data fields.

  User data fields add customized data fields used by other Deployment Management entities, such as workflows, object types, and environments. "Configuring User Data" on page 200 discusses how to configure these user data fields.

- Step 7: Configure your security and access requirements.

  Part of any migration or deployment process are the permissions required to perform various decisions or executions. Two of the ways in which PPM controls access to perform these decisions or executions are by licenses and access grants.

  - **Licenses**: Provide users with access to a PPMproduct such as Deployment Management, but they do not dictate what actions the users can perform.
  - **Access Grants**: When used with licenses, Access Grants define the actions a user can perform within a PPM product.

  For more information concerning security and access grants, see the *Security Model Guide and Reference*.

- Step 8: Educate your deployment system users.

  After your Deployment Management system is configured and tested, you should train your users on the new business process. The following offer some guidance on how to educate your Deployment Management system users:

  - **Basic Deployment Management training.** Ensure that each user understands how to create, process, and report on packages.
  - **Process-specific training.** Ensure that each user understands the new process. Consider holding a formal meeting or publishing documents on the workflow steps and packages.
  - **User responsibilities.** Ensure that users understand their individual roles in the process. For example, the QA team may be restricted to approving the testing phase of a package. You can use email notifications that are part of Deployment Management to communicate information about user roles. Notifications can be very detailed, informing the recipients of their responsibilities.

Related Information

The following documents also include information related to configuring Deployment Management:

- *Deployment Management User’s Guide*
- *Commands, Tokens, and Validations Guide and Reference*
- *Open Interface Guide and Reference*
- *Reports Guide and Reference*
- *Security Model Guide and Reference*
- *Creating Portlets and Modules*
- *HPE-Supplied Entities Guide*(includes descriptions of all Deployment Managements portlets, object types, and workflows)

Chapter 2: Gathering Process Requirements

- "Overview of Gathering Process Requirements" below
- "Defining Workflows" on the next page
- "Defining Object Types" on page 26
- "Defining Environments" on page 30
- "Defining Notification Templates" on page 32
- "Defining User Data Fields" on page 32
- "Defining Security and Access" on page 33

Overview of Gathering Process Requirements

Before configuring a Deployment Managementprocess, you should gather information concerning your process, such as the steps in the workflow, the objects to be deployed, and the source and destination environments. After this information is collected, you can then begin configuring your Deployment Management process.

This section offers you guidance on how to gather information for your deployment process. The subjects covered in this section are:

- **Defining workflows.** What are the steps in your deployment process (workflow)? Which steps require manual decisions (reviews and approvals)? Which steps require automatic executions? See "Defining Workflows" on the next page for detailed information.
- **Defining object types.** What are you deploying? Are you deploying files? Are you deploying data? What scripts might be required to support the deployment? See "Defining Object Types" on page 26 for detailed information.
- **Defining environments.** What is the source environment for your deployment? What are the destination environments for your deployment? See "Defining Environments" on page 30 for detailed information.
- **Defining notification templates.** Is the correct notification template in place? Does your deployment require a new notification template? See "Defining Notification Templates" on page 32 for detailed information.

- **Defining user data fields.** Does your deployment require additional user information to process correctly? See "Defining User Data Fields" on page 32 for detailed information.
- **Defining security and access.** Who will be allowed to build the deployment packages? Who should receive notifications? Who will be allowed to approve the deployment at each step? See "Defining Security and Access" on page 33 for detailed information.

Defining Workflows

A workflow is a digitized process where a logical series of steps define the path followed by a package. Workflow steps can range from reviews and approvals to performing the actual migration.

When defining a deployment workflow, you must first determine the intent of the business process the deployment workflow will follow. For example:

- Are you designing a simple migration of a file from one environment to another environment with little oversight or supervision?
- Are you designing a business-wide deployment process with a great deal of oversight and supervision?

After you understand the intent of the business process, you can then begin to define the deployment workflow itself. The following lists the basic components of a workflow:

- **Workflow steps.** Workflow steps are the events (steps) of the process (workflow) that start your business process. Workflow steps are where decisions are made and actions are taken. The following lists the different types of workflow steps:
  - **Decision steps**: Require an external process to decide their outcome.
  - **Execution steps**: Perform actual work or actions.
  - **Condition steps**: Logic steps used for complex workflow processing.
  - **Subworkflows steps**: Represent multiple workflow steps that follow a consistent pattern.
- **Transitions between workflow steps**: Represent the outcome of one workflow step that leads to the next workflow step. Workflow steps can have more than one transition.
- **Security**: Determines who can access a workflow step. Each workflow step includes a list of who can access a workflow step, who can approve a workflow step, whether only one user can approve the workflow step, if several users can approve the workflow step, and whether multiple users must approve the workflow step.

- **Notification**: Determines who hears about the workflow step. Each workflow step includes a list of who will be notified about the workflow step and when the notification will occur.

Gathering Information for Workflow Steps

Workflow steps are the events of the process. Deployment Management employs the following types of workflow steps to gather information:

- **Decision steps**: Require an external process to decide their outcome, such as reviews, approvals, or coding.
- **Execution steps**: Perform actual work or actions, such as file migration, automatic time-stamping, or automatic package status changes.
- **Condition steps**: Logic steps used for complex workflow processing, such as AND and OR.
- **Subworkflow steps**: Represent multiple workflow steps that follow a consistent pattern, such as code rework or unit testing.

Gathering Information for Decision Steps

Decision steps are workflow steps that require an external process to decide their outcome, such as reviews, approvals, or coding. "Table 2-1. Decision Workflow Checklist" below provides a configuration consideration checklist to help define decision steps. See "Worksheets" on page 215 for a complete list of decision step considerations.

Table 2-1. Decision Workflow Checklist

Complete	Decision Step Check Item	Example
	What is the name of this workflow step?	• Review Request • On Hold • In Rework
	What is the status of the package at this workflow step?	• On Hold • New • In Review
	What are the transitions from this workflow step?	• Assign

Table 2-1. Decision Workflow Checklist, continued

Complete	Decision Step Check Item	Example
		• Review • On Hold
	Who or what groups can act on this step (approve, cancel, reassign)?	• Security Groups • Users • Tokens
	How many decisions are required to exit this workflow step?	• Only one • At Least One • All
	What event triggers the notification?	• When the process reaches the workflow step • When a specific result is reached
	Who or how many receive the notification?	• Email Address (group alias) • Security Group
	What is the notification message?	• Test complete • Approval required
	Use this workflow step as a timeout? If so, for how long?	• 1 day • 2 days
	Are you using segregation of duties?	• Based on owner of the workflow • Based on workflow step

Gathering Information for Execution Steps

Execution steps are workflow steps that perform actual work or actions, such as file migration, automatic time-stamping, or automatic package status changes. "Table 2-2. Execution Workflow Checklist" on the next page provides a configuration consideration checklist to help define execution steps. See "Worksheets" on page 215 for a complete list of execution step considerations.

Table 2-2. Execution Workflow Checklist

Complete	Execution Step Check Item	Example
	What is the name of this workflow step?	• Create Package • Close • Set Temp Date
	Will this workflow step run this command?	• Cancel project • Update project
	Will the object type (package line) run this command?	• Open database • Copy file to environment
	What is the source environment?	(If required) • KINTANA_SERVER
	What is the destination environment?	(If required) • KINTANA_SERVER
	What are the transitions from this workflow step?	• Succeeded • Failed
	Who owns this execution step?	• Security Group • User
	What event triggers the notification?	• When the process reaches the workflow step • When a specific result is reached
	Who or how many receive the notification?	• Email Address (group alias) • Security Group
	What is the notification message?	• Test complete • Approval required
	Use this workflow step as a timeout? If so, for how long?	• 1 day • 2 days
	Are you using segregation of duties?	• Based on owner of the workflow • Based on workflow step

Gathering Information for Condition Steps

Condition steps are logic steps used for complex workflow processing, such as AND and OR. "Table 2-3. Condition Workflow Checklist" below provides a configuration consideration checklist to help define your condition steps.

Table 2-3. Condition Workflow Checklist

Complete	Condition Step Check Item	Example
	What is the name of this workflow step?	• AND • OR
	What is the status of the request at this workflow step?	• On Hold • New • In Review
	What are the transitions from this workflow step?	• Succeeded • Failed
	Who owns this workflow step?	• Security Group • User • Standard Token
	What event triggers the notification?	• When the process reaches the workflow step • When a specific result is reached
	Who or how many receive the notification?	• Email Address (group alias) • Security Group
	What is the notification message?	• Test complete • Approval required
	Use this workflow step as a timeout? If so, for how long?	• 1 day • 2 days
	Are you using segregation of duties?	• Based on owner of the workflow • Based on workflow step

Gathering Information for Subworkflow Steps

Subworkflow steps represent multiple workflow steps that follow a consistent pattern, such as code rework or unit testing. "Table 2-4. Subworkflow Workflow Checklist" below provides a configuration consideration checklist to help define your subworkflow steps. See "Worksheets" on page 215 for a complete list of subworkflow step considerations.

Table 2-4. Subworkflow Workflow Checklist

Complete	Subworkflow Step Check Item	Example
	Is an existing workflow available as a subworkflow?	• Yes • No
	What is the name of this subworkflow?	• QA Test Cycle • QA Review Cycle
	What are the transitions from this workflow step?	• Succeeded • Failed
	Who owns this workflow step?	• Security Group • User
	What event triggers the notification?	• When the process reaches the workflow step • When a specific result is reached
	Who or how many receive the notification?	• Email Address (group alias) • Security Group
	What is the notification message?	• QA Test Cycle Succeeded • QA Test Cycle Failed
	Use this workflow step as a timeout? If so, for how long?	• 1 day • 2 days
	Are you using segregation of duties?	• Based on owner of the workflow • Based on workflow step

General Workflow Design Guidelines

"Table 2-5. Workflow logical guidelines" below provides a workflow logical guideline checklist that you can use to configure your deployment workflow.

Table 2-5. Workflow logical guidelines

Complete	Guideline	Reason
	Workflows	
	Make one or more workflows available to process the packages.	Each workflow is assigned one workflow scope. The possible workflow scopes are: • Request (Demand Management) • Packages (Deployment Management) • Release Distributions (Deployment Management)
	Beginning and Closing Steps	
	Workflow must have a beginning step.	No processing can be done if the workflow has no beginning step.
	Workflow must have at least one step.	No processing can be done if the workflow has no steps.
	Workflow must have at least one Close step.	The package line cannot be closed without a Close step in the workflow.
	First workflow step cannot be a condition.	Workflow processing may not be correct if the first step is a condition.
	Close steps must not have a transition on `Success' or `Failure.' Return steps must have no outgoing transitions.	The package or request will not close if a transition exists on `Success.'
	Close step in subworkflow closes entire package line or request.	Do not include a Close step in a subworkflow unless you want to close the workflow in the subworkflow.
	All Steps	
	All steps must be enabled.	Disabled steps cannot be used by the workflow so the process stops.
	Each enabled workflow step must have at least one incoming transition (except the	It is not possible to flow to a workflow step without an incoming transition.

Table 2-5. Workflow logical guidelines, continued

Complete	Guideline	Reason
	beginning step).	
	Transition value is not a valid validation value (error).	The validation value has changed since the transition has been made.
	\`Other Values' and \`All Values' transitions must not exist in the same step.	\`Other Values' transition is always ignored if an \`All Values' transition exists.
	Each workflow step must have at least one outbound transition.	The branch of the workflow stops indefinitely without closing the package line or request.
	Each value from a list-validated validation must have an outbound transition.	Some validation values do not have transitions defined.
	Step with text or numeric validation must have an \`Other Values' or \`All Values' transition.	Because text and numeric validations are not limited, an \`Other Values' or \`All Values' transition should be defined.
	Notifications with reminders must not be set on results that have transitions.	Transition into the Return step does not match the validation.
	Decision Steps	
	Each decision step must have at least one security group, user, or token defined on the **Security** tab.	No one is authorized to act on the step without a security group.
	Execution Steps	
	Each manual execution step must have at least one security group, user, or token defined on the **Security** tab.	No one is authorized to act on the step without a security group.
	An immediate execution step must not have a transition to itself on \`Success' or \`Failure.'	The workflow could loop indefinitely.
	Condition Steps	
	A condition step must not have a transition to itself.	A condition with a transition to itself could cause the workflow to run indefinitely.
	AND or OR condition steps must have at least two incoming transitions.	An AND or OR condition with only one incoming transition will always immediately be true and have no effect.
	Subworkflows	
	Subworkflows must have at least one	Must include a Return step.

Table 2-5. Workflow logical guidelines, continued

Complete	Guideline	Reason
	Return step.	
	Top-level workflow must not have a Return step.	Only subworkflows have a Return step.

Defining Object Types

Object types define the technical steps required to deploy application changes for packaged applications, custom applications, legacy systems, and web content. Many types of objects are deployed through a workflow, such as files, SQL scripts, and data. Each object requires different information for proper processing. The information required to properly process an object is defined by the object type. For example, the object type for a file migration requires the following:

- The name of the file
- The type of file
- The location of the file

Additional information, such as file compilation after file migration, can also be specified by the object type. A single workflow can process many different object types, since the workflow specifies the process and the environments. The package specifies each object and the associated object type as a package line in the package. Information contained on a package line (defined in the object type) works with the workflow process to ensure that the object is correctly deployed.

For each object deployed through the process, you should collect the following information:

- Name of the object
- Object category (optional, used for reporting purposes)
- Parameters describing the object: Such as, what it is, where it is, its name, and what needs to be done to it. This information will translate into object type fields. For each object type field, define the following:
    - Field name
    - Validation and component type (dictated by the validation)
    - Field behavior: Such as whether it is displayed, required, and any default behavior.
- Commands required for the object being deployed. Object type commands often reference

information stored in the parameters. These commands are run at specific points (execution steps) in the workflow. For example, you might shut down a database before inserting new data into the database. For more information, see "Determining Commands Needed for Objects" on the next page.

For an example of conceptual workflow steps using different object types, see "Table 2-6. Example workflow" below. See "Configuring Object Types" on page 116 for information on how to configure object types.

Table 2-6. Example workflow

Step Name	Workflow Step	Transition Values	Object Type Description
Migrate DEV to TEST	Decision	Approved Not Approved	Not applicable. Decision workflow step.
Stop the Server	Execution	Succeeded Failed	Connect to the server and stop the processes running on it.
Evaluate Object Type	Execution	Database File SQL Script	Evaluate the object type for each package line. Resolve the object type token.
Migrate to Database Environment	Execution	Succeeded Failed	Migrate the database changes to the TEST database environment. To do this, run commands located in the object type.
Migrate to Server Environment	Execution	Succeeded Failed	Migrate the changes to the TEST server environment. To do this, run commands located in the object type.
Sync	Condition	Success	Have all package lines provide this step before continuing to the next process step.
Compile Code	Execution	Succeeded Failed	Connect to the server and compile the code located on it.
Start the Server	Execution	Succeeded Failed	Connect to the server and start the processes on it.
Close FAILED	Execution	Succeeded	When a package line (object) provides this step, close the package.

Determining Commands Needed for Objects

When defining a deployment process, consider what commands need to be run to achieve the desired results. Commands control which steps must be run for each workflow step for the deployment process to run properly. This can involve such things as migrating a file, executing a script, or compiling code.

At early stages of process development, it often helps to list the functional steps and desired results of the commands. It also helps to specify when in the deployment process these commands should be run. It is then possible to use this information to build your commands adhering to Deployment Management's command standards, or to deliver these as design specifications for engineers in your group.

Collect the following information for each object type:

- The goal or purpose of the commands
- Functional steps within the commands
- When the commands should be run

For additional information on building commands, see the *Commands, Tokens, and Validations Guide and Reference*.

Object Type Checklist

"Table 2-7. Object type configuration checklist" below provides a configuration consideration checklist to help define your deployment system. See "Worksheets" on page 215 for a complete list of object type considerations.

Table 2-7. Object type configuration checklist

Complete	Object Type Check Item	Configuration Consideration
	Define an object type for each type of object to be deployed.	Include creating fields that describe the object and commands required to process it during deployment.
	Fields defined for	• Fields are required to define the object. Make sure that the

Table 2-7. Object type configuration checklist, continued

Complete	Object Type Check Item	Configuration Consideration
	the object type.	correct parameters describe the object to be deployed. • See "Configuring Object Types" on page 116. • See the *Commands, Tokens, and Validations Guide and Reference*.
	Commands defined for the object type.	• All commands needed to process and deploy the object have been constructed. • See "Configuring Object Types" on page 116. • See the *Commands, Tokens, and Validations Guide and Reference*.
	Conditions set in commands for the object type.	• Conditions to steps within the command that dictate when the specific command steps have run, and that they have also been added. • See the *Commands, Tokens, and Validations Guide and Reference*.
	Is the object type enabled?	Disabled object types cannot be used.
	Object type and workflow.	The following items should be coordinated between the workflow and object type: • Decide which workflow steps will run the object type commands. • Decide which object type commands to run at specific workflow steps (using command conditions). • Workflow step source validations and object type field validations are in agreement. This is required when transitioning based on a field value (using token, SQL or PL/SQL execution types). • Allow the object type use for the workflow (set in the workflow window - **Deployment Management Settings** tab).
	Object types and environments.	Specify any environment overrides that must correspond to the object type commands.

Defining Environments

An environment can be a server, a single database instance, or an associated remote client machine. Deployment workflows require the source and destination environments as part of the execution workflow steps. Each environment must be carefully configured to ensure that passwords, communication protocols, and transfer protocols are configured properly. These configured environments can then be used in the deployment process.

> **Note:** If there are multiple applications on a single environment, you can use the application codes feature in the environment definition.

"Table 2-8. Example environments settings" below shows the environments for several steps in a workflow. For information concerning the configuration of environments, see "Configuring Environments" on page 158.

Table 2-8. Example environments settings

Workflow Execution Step	Source Environment	Destination Environment
Stop the Server	DEV Server	TEST Server
Migrate to Server Environment	DEV Server	TEST Server
Compile Code	DEV Server	TEST Server
Start the Server	DEV Server	TEST Server
Migrate to Database Environment	DEV Database	TEST Database

Environments Checklist

"Table 2-9. Environments checklist" below provides a configuration consideration checklist to help define your environments.

Table 2-9. Environments checklist

Complete	Environments Check Item	Configuration Consideration
	Is the source environment defined?	• Is the source environment a server or a client? • Does the source environment

Table 2-9. Environments checklist, continued

Complete	Environments Check Item	Configuration Consideration
		include a database?
	Is the destination environment defined?	• Is the destination environment a server or a client? • Does the destination environment include a database?
	What is the connection protocol?	Is the connection protocol telnet, SSH, or SSH2?
	What is the transfer protocol?	Is the transfer protocol FTP, FTP (active), FTP (passive), Secure Copy, or Secure Copy 2?
	Is the environment enabled?	Disabled environments cannot be used.
	Is there an environment group?	• What is the environment group name? • What is the source environment? • Is the execution order serial or parallel?
	Is the environment group enabled?	Disabled environment groups cannot be used.
	Is there an environment refresh?	• What is the environment refresh name? • What is the source environment? • What is the destination environment?
	Is the environment refresh enabled?	Disabled environment refreshes cannot be used.
	Environments and workflow considerations.	Specify the source and destination environments (or environment groups) on the workflow execution steps.
	Environments and object type considerations.	Specify any environment overrides in the object type commands.

Defining Notification Templates

Notification templates are preconfigured email forms that can be used to quickly construct the body of a message. Notification templates are used by entities of Deployment Management, such as workflows and packages. When configuring a workflow, select the notification template you want to use for each workflow step. Deployment Management comes with a set of standard notification templates. You can use these existing templates, modify the existing templates, or create new notification templates for your business process. "Configuring Notification Templates" on page 193 provides detailed information concerning the configuration of notification templates.

Notification Templates Checklist

"Table 2-10. Notification template checklist" below provides a configuration consideration checklist to help define your notification templates.

Table 2-10. Notification template checklist

Complete	Notification Template Check Item	Configuration Consideration
	Is the notification template enabled?	Disabled notification template cannot be used.
	Notification template and security group considerations.	Set ownership groups for these entities. Members of the ownership group (determined by associating security groups) are the only users who can edit the entities.

Defining User Data Fields

Deployment Management entities, such as packages and workflows, include a set of standard fields that provide information about those entities. While these fields are normally sufficient for day-to-day processing, you can add "user data" fields to capture additional information specific to your business process. "Configuring User Data" on page 200 provides detailed information concerning the configuration of user data fields.

User Data Checklist

"Table 2-11. User data checklist" below provides a configuration consideration checklist to help define your user data fields.

Table 2-11. User data checklist

Complete	User Data Check Item	Configuration Consideration
	Are the user data fields enabled?	Disabled user data fields cannot be used.
	User data and security group considerations.	Set ownership groups for these entities. Members of the ownership group (determined by associating security groups) are the only users who can edit the user data fields.

Defining Security and Access

Included as part of a deployment process are the permissions required to perform various decisions or executions. PPM controls access to perform decisions and executions by:

- **Licenses**: Provide users with access to a PPM products such as Deployment Management, but licenses do not dictate what actions a user is authorized to perform.
- **Access Grants**: When used with licenses, access grants define the actions a user is authorized to perform within a product PPM.

For example, you can restrict a user's actions, as follows:

- License, Deployment Management
  - Access Grant, View Packages - Those who can view packages
  - Access Grant, Edit Packages - Those who can edit packages

For more information concerning licenses and access grants, see the *Security Model Guide and Reference*.

When designing deployment processes, use security groups or dynamic access (tokens). Avoid specifying a list of users to control an action. If the list of users changes, you must update your workflow in a variety of places to keep the deployment process running correctly. By using a security group instead of a list of users, you can update the security group once, and the changes are propagated throughout the workflow.

"Table 2-12. Example of workflow security groups" below provides an example of which security groups can access a deployment workflow, and at which workflow step.

Table 2-12. Example of workflow security groups

Workflow Step	Security Groups
Create a package	Financial Apps - Engineer Financial Apps - Database Financial Apps - Manage Deployment
Use the deployment workflow	Financial Apps - Engineer Financial Apps - Manage Deployment
Use a file object type	Financial Apps - Engineer Financial Apps - Database Financial Apps - Manage Deployment
Use a database object type	Financial Apps - Database Financial Apps - Manage Deployment

Security and User Access Checklist

"Table 2-13. Security and user access checklist" below provides a configuration consideration checklist to help define your security and user access requirements.

Table 2-13. Security and user access checklist

Complete	Security and User Access Check Item	Configuration Consideration
	Created security groups for access to screens and functions	Security groups to be used to grant access to certain screens and functions have been created.
	Created security groups for association with workflow steps	Security groups to allow users to act on a specific workflow step have been created.

Table 2-13. Security and user access checklist, continued

Complete	Security and User Access Check Item	Configuration Consideration
	Set security on package creation	All available options for restricting who can create and submit packages have been set.
	Set security on package processing	All available options for restricting who can process packages have been set.
	Set security on deployment system configuration	You have specified who can modify the deployment process. This includes editing the workflow, object type, environment, and security groups.
	Security group and workflow considerations	• Associate security groups with workflow steps. Users in the included groups can act on the step. • Set workflow and workflow step ownership.
	Security group and object type considerations	Set ownership groups for object types. Members of the ownership group (determined by associating security groups) are the only users who can edit the object type.
	Security group and environment considerations	Set ownership groups for environments. Members of the ownership group (determined by associating security groups) are the only users who can edit the environments.
	Security group and notification template considerations	Set ownership groups for notification templates. Members of the ownership group (determined by associating security groups) are the only users who can edit the notification templates.
	Security group and user data considerations	Set ownership groups for user data. Members of the ownership group (determined by associating security groups) are the only users who can edit user data.

Chapter 3: Configuring Workflows

- "Overview of Workflows " below
- "Mapping Workflows " on page 39
- "Opening Workflow Workbench " on page 44
- "Creating Workflows" on page 44
- "Configuring Workflow Steps" on page 51
- "Integrating Object Type Commands and Workflows " on page 81
- "Integrating Environments and Workflows " on page 81
- "Integrating Request and Package Workflows " on page 83

Overview of Workflows

A workflow represents a business process, and is used to map business rules and processes to your organization. This section covers information about Deployment Management workflows.

The basic components of a workflow are as follows:

- **Begin**: For each workflow, you must explicitly define the first eligible workflow step.
- **Workflow steps**: Events that are linked together to form a complete workflow. The basic types of workflow steps are:
    - **Decision steps**: Represent manual activities performed outside of PPM. For example, a user or group of users approves a request.
    - **Execution steps**: Represent actions that are automated through PPM. For example, a Web page is updated with the results of a test.
    - **Condition steps**: Logic steps used in complex workflow processing. For example, you can set up a condition step that allows the workflow to proceed only when each workflow step is completed.
    - **Subworkflow steps**: Represent multiple workflows steps (the subworkflow) in a workflow. For example, a test workflow step in the main workflow represents a series of tests and approvals.
- **Transition**: The results of a workflow step that must be communicated to another workflow step. A

transition occurs after a workflow step is completed.

Examples:

- The result of a decision step is Approved or Not Approved.
- The transition for a step labeled Analysis and Design (for a software application) could be Completed or Needs More Work.

Because a single step can have several possible results, you can define multiple outgoing transitions for each workflow step.

- **Workflow step security**: Determines who has permission to run or choose a result for a workflow step. For example, you can specify that only the development manager can approve or deny a Design Review decision step.
- **Notifications**: Email alerts sent out at specific workflow steps. For example, when a package reaches an Evaluate decision step in the workflow, an email alert is sent to the development manager.
- **Close step**: Ends the workflow. It is an execution step that marks the package as completed.

"Table 3-1. Component types" below lists component types included in workflows, and the components of each type that you can access from the Workflow Workbench.

Table 3-1. Component types

Component Type	Icon	Out-of-the-Box Component
Decisions	?	• Approval w/ Cancel • Approve (All Users) • Approve (One User) • BTO Approve for Testing • BTO Assign • BTO Design • BTO Develop • BTO Evaluate Quality • BTO Post Change Evaluation • DEM - Approval (One) • DEM - Assign • DEM - Contact Vendor • DEM - Develop

Table 3-1. Component types, continued

Component Type	Icon	Out-of-the-Box Component
		• DEM - Get Patch • DEM - Vendor Response • DEM - Y/N • Design Review • Evaluate • MAM Develop • PM - In Flight • PM - On Hold • PMO - Complete • Perform Manual Step • Production Sanity Testing • QA Test • Quality Center Sanity Test Execution • Quality Center Test Execution • Quality Center Test Lab Setup • Quality Process Entry • Rework Code • User Sign Off
Conditions		SYNCH
		AND
		OR
		FIRST LINE
		LAST LINE

Table 3-1. Component types, continued

Component Type	Icon	Out-of-the-Box Component
Executions		• Close (Immediate failure)
		• Close (Immediate success) • Close (Manual success)
		• Check Object Type • DEM - Rejected On • DEM - SLA Satisfied On • DEM - Scope Check • DLV - Execution (Immediate) • DLV - Execution (Manual) • DLV - Execution w/ Reset • DLV - Execution w/ Reset/Abort • DLV - Exec w/ Partial Reset
		• DEM - Priority Router • DEM - Internal/External Check • Immediate Execution w/Comment • OA - Get Current Instance Mode • OA - Set Instance Mode • OA - Reset Instance Mode • Package Priority

Mapping Workflows

Mapping all the individual workflow steps into a single workflow is a two-stage process:

Stage 1. Create a block diagram. Map each workflow step worksheet as one block in the diagram. On the block diagram, include transitions, workflow step security, and notifications. This is illustrated in "Figure 3-1. Stage 1. Create a block diagram" on page 41.

Stage 2. Map the block diagram to the workflow. Open the Workflow Workbench and create a workflow. Map each component from the block diagram to the new workflow (see "Figure 3-2. Stage 2. Create the workflow" on page 43).

Figure 3-1. Stage 1. Create a block diagram

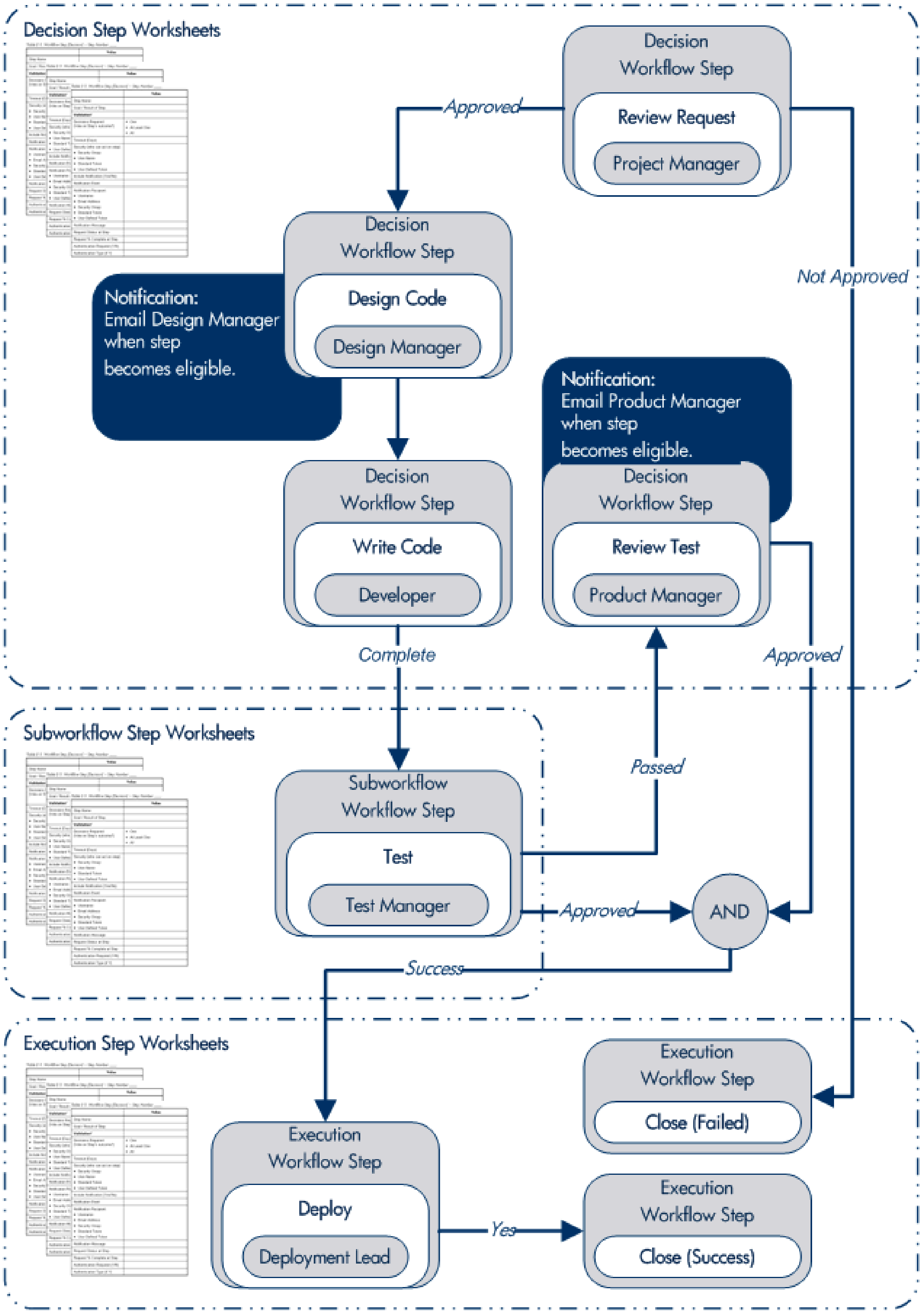
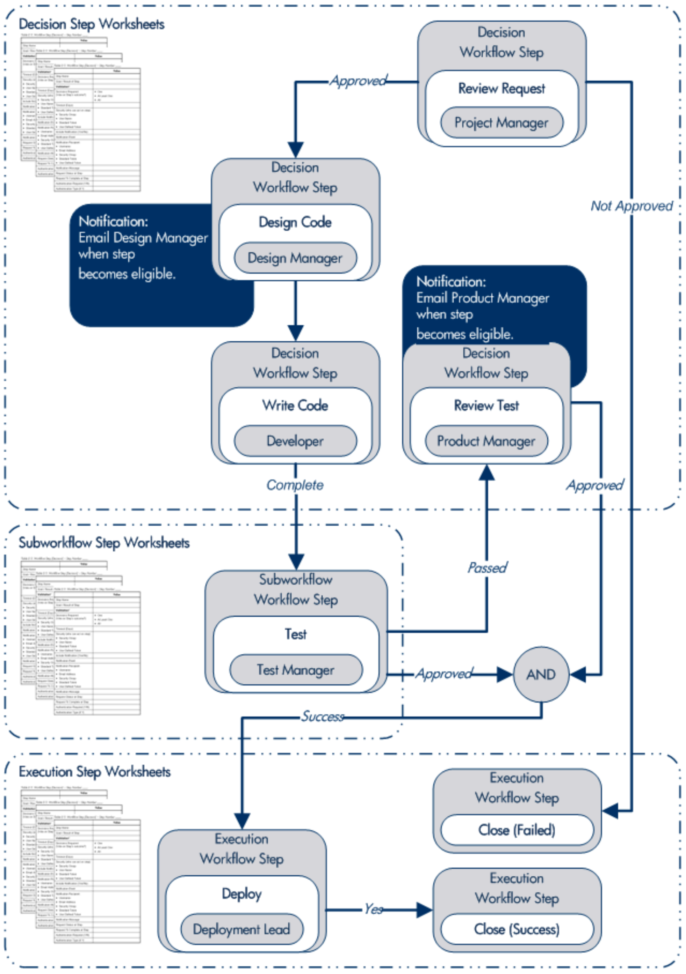
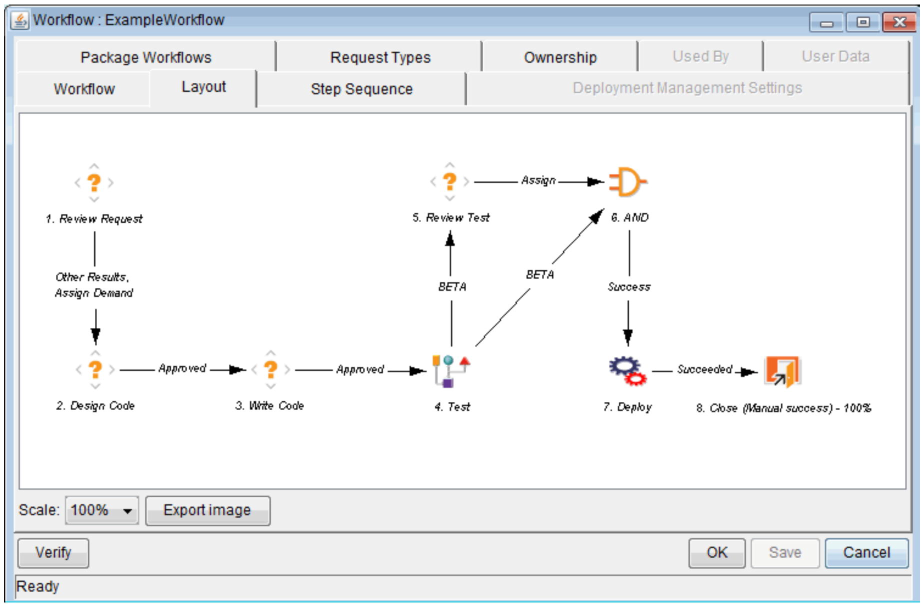

Decision Step Worksheets
Decision Workflow Step
Review Request
Project Manager
Approved
Not Approved
Decision Workflow Step
Design Code
Design Manager
Notification: Email Design Manager when step becomes eligible.
Notification: Email Product Manager when step becomes eligible.
Decision Workflow Step
Write Code
Developer
Decision Workflow Step
Review Test
Product Manager
Complete
Approved
Subworkflow Step Worksheets
Passed
Subworkflow Workflow Step
Test
Test Manager
Approved
AND
Success
Execution Step Worksheets
Execution Workflow Step
Close (Failed)
Execution Workflow Step
Deploy
Deployment Lead
Yes
Execution Workflow Step
Close (Success)

Figure 3-2. Stage 2. Create the workflow





Opening Workflow Workbench

To open the Workflow Workbench:

1. Log on to PPM.
2. From the menu bar, select **Open > Administration > Open Workbench**.

   The Workbench opens.
3. From the shortcut bar, select **Configuration > Workflows**.

   The Workflow Workbench and Workflow Step Sources windows open.

Creating Workflows

This section provides basic instructions on how to use the Workflow Workbench to create a workflow.

Configuring General Information for a Workflow

To provide basic workflow information:

1. From the Workbench shortcut bar, select **Configuration > Workflows**.

   The Workflow Workbench and Workflow Step Sources windows open.
2. In the Workflow Workbench window, click **New Workflow**.

   The Workflow window opens.
3. In the **Name** field, type a name for the workflow.
4. From the **Workflow Scope** list, do one of the following:
   - For Deployment Management packages, select **Packages.**
   - For Deployment Management releases and distributions, select **Release Distributions.**
5. In the **Description** field, you can type a short description of the workflow and its purpose.
6. To make this workflow available for use in PPM, for the **Enabled** option, select **Yes.**
7. Click **Save.**

Next, add steps to the workflow. For instructions, see " Choosing Workflow Steps" below.

Choosing Workflow Steps

PPM comes with predefined workflow steps, which are available through the Workflow Step Sources window. You can use the **Filter by** fields in this window to filter the available workflow steps for display. The following folders, which contain workflow steps classified by type, are available in the Workflow Step Sources window:

- **Decisions**
- **Conditions**
- **Executions**
- **Subworkflows**

To evaluate a workflow step, determine which of the workflow folders it corresponds to. Expand the folder, and then add the workflow step that best suits your needs to the **Layout** tab. For instructions, see " Adding Steps to a Workflow" on page 48.

Validations determine the values a workflow step can have (see "Configuring Validations for Workflow Steps " on page 77). Check these to see if they meet your requirements for transition to the next workflow step.

Overview of Decision Workflow Steps

Decision workflow steps represent manual activities performed outside of PPM. Decision workflow steps include such activities as:

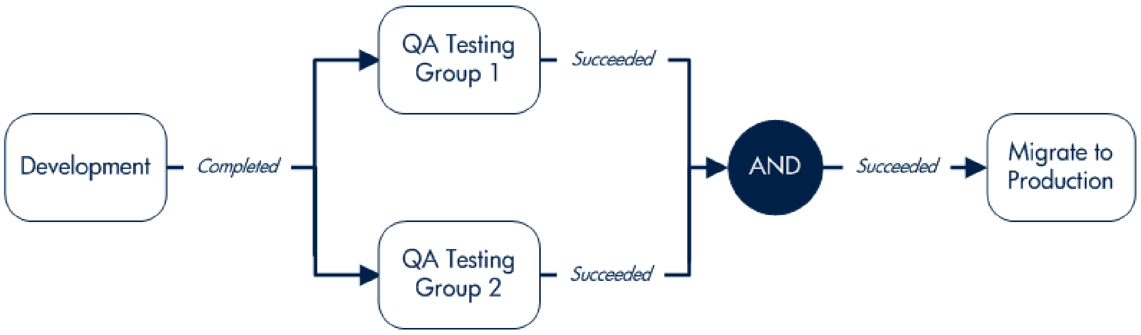
- Decisions made by committees
- Code designs and reviews

Overview of Condition Workflow Steps

Condition workflow steps are logic steps used for complex workflow processing, such as allowing the workflow to proceed only after each workflow step is completed. The condition workflow steps are as follows:

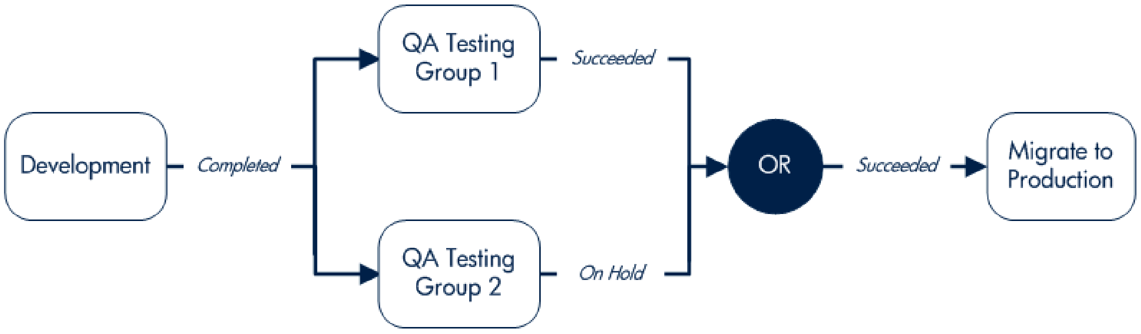
- **AND.** Met only after all workflow steps leading to it reach the specified required status. "Figure 3-5. AND example " below shows an AND condition workflow step.

  Figure 3-5. AND example

- **OR.** Met if at least one of the workflow steps leading to it reaches the required status specified for it. "Figure 3-6. OR example" below shows an OR condition workflow step.

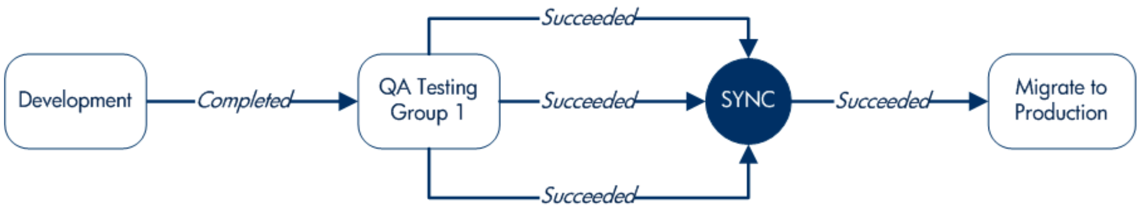
  Figure 3-6. OR example

- **SYNC.** Successful only if all the package lines of that package reach the status required for the workflow step immediately preceding that SYNC step.

  Consider the business process illustrated in "Figure 3-7. SYNC example" below. According to the flow chart, after QA Testing Group 1 succeeds for all package lines, SYNC succeeds, and then the next step, Migrate to Prod, becomes eligible.

  Figure 3-7. SYNC example

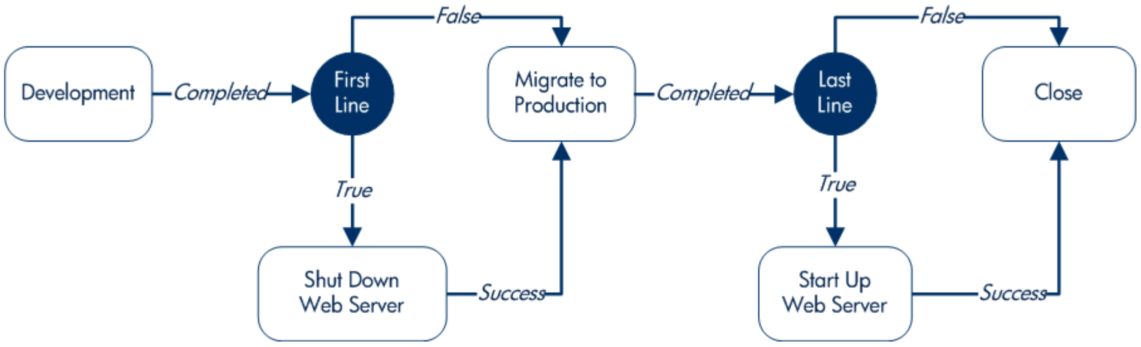
- **FIRST LINE and LAST LINE.** Only the first line to reach the condition workflow step takes the True transition. All subsequent lines take the False transition.

  For LAST LINE, only the last active line to reach the condition workflow step takes the True transition. All previous lines take the False transition.

  The business process illustrated in "Figure 3-8. FIRST LINE and LAST LINE example" below could be part of a Web site maintenance life cycle. As part of this life cycle, three HTML files are processed on three respective package lines in a single package. The Web site updates are large enough to warrant shutting down the Web server during change migration.

  By including a FIRST LINE step, only the first line causes the server to shut down. The server remains down while the rest of the changes are migrated to production. By including a LAST LINE workflow step, the server remains down until the last active line reaches the condition step. The last active line takes the True transition, the Web server starts up, and then the maintenance is complete.

  Figure 3-8. FIRST LINE and LAST LINE example



Overview of Execution Workflow Steps

Execution workflow steps represent actions that are automated through PPM. Execution workflow steps include such activities as:

- Create a package
- Run object type commands
- Package priority
- Create a request
- Run request commands

- Run workflow step commands
- Close the workflow (Close workflow step)

Overview of Subworkflow Workflow Steps

A subworkflow is a process unit that contains a series of steps that perform a functional subcomponent of a workflow. Subworkflows allow you to model complex business processes in logical, manageable, and reusable subprocesses. Within its parent workflow, each subworkflow is represented as a single workflow step.

After the workflow process reaches the subworkflow step, it follows the path defined in that subworkflow. Subworkflows can either end the workflow or return to the parent workflow.

The following restrictions apply to subworkflows:

- You cannot use a subworkflow to process a request or a package as a standalone business process.
- A subworkflow can reference other subworkflows, but not itself.
- A subworkflow can be referenced only by workflows or subworkflows of the same workflow scope.
- Permissions specified on the **Security** tab of the calling subworkflow step determine who can bypass the steps with the subworkflow.

Adding Steps to a Workflow

You assemble workflow steps into workflows on the **Layout** tab of the Workflow window.

To add a step to a new workflow:

1. In the Workflow window for your new workflow, click the **Layout** tab.

   To the right of the Workflow window, the Workflow Step Sources window contains a library of steps, classified by type, that you can use to build your workflows. The window also includes **Filter by** lists, which you can use to selectively display a subset of available steps.

2. From the first **Filter by** list, select one of the following:

- For HPE packages, select **Packages.**
- For HPE releases and distributions, select **Release Distributions.**
- Select **Requests.**

3. In the second **Filter by** list, you can select an additional filter condition to further refine the available workflow steps listed.
4. Click **New**.
5. Click **Save**.
6. To view the available steps, expand the folders in the Workflow Step Sources window.

   > **Note:** For more information on how to select the steps for your workflows, see " Choosing Workflow Steps" on page 45

7. Determine which step to add as the first step, and then drag and drop it onto the **Layout** tab.

   After you add a step to the **Layout** tab, the Workflow Step window opens. Use this window to configure the following:

   - General workflow step properties. See "Configuring Properties of a Workflow Step " on page 52.
   - Workflow step security. See "Configuring Security for Workflow Steps" on page 53.
   - Notifications for the workflow step. See "Configuring Notifications for Workflow Steps " on page 54.
   - Timeouts for the workflow step. See "Configuring Timeouts for Workflow Steps" on page 66.
   - Step fill color for graphic workflow display. See "Adding Color to Workflow Steps" on page 79.
   - Segregation of duties. See "Configuring Segregation of Duties for Workflow Steps" on page 80.
   - Step fill color for graphic workflow display. See "Adding Color to Workflow Steps" on page 79.

8. After you finish configuring all the steps in the workflow, click **OK**.

Adding Close Step

Every workflow must include a close step. A close step is a type of execution workflow step. You can find it in the **Executions** folder in the Workflow Step Sources window.

You can use one of the following three close steps in a workflow:

- **Close (Immediate success).** This close step immediately completes a request or package with a status of Success.
- **Close (Manual success).** This close step requires manual intervention to complete a request or package and set the request or package status to Success.
- **Close (Immediate failure).** This close step immediately completes a request or package with a status of Failure.

You add a close workflow step to a workflow as you would any other type of workflow step.

Configuring Reopen Workflow Steps

Users who have the required access grants can reopen closed requests. A reopened request begins at a step specified as the reopen workflow step for the workflow.

To specify a reopen step for a workflow:

1. In the Workflow Workbench, click the **Workflow** tab.
2. In the **Reopen Step** list, select the reopen workflow step.
3. Click **OK.**

Adjusting Workflow Step Sequences

After you assemble all the workflow steps on the **Layout** tab, you can adjust their sequence.

To adjust the sequence of steps in an open workflow:

1. In the Workflow window, click the **Step Sequence** tab.

   The **Step Sequence** tab lists all of the workflow steps.
2. Select a workflow step, and then click the arrows at the bottom of the tab to move the selected workflow step up or down in the display sequence.
3. Click **Save.**

   On the **Workflow** tab, the **First Step** field displays the first workflow step.

Verifying and Enabling Workflows

To make a workflow available for use you must verify it, and then enable it. Workflow verification ensures correct workflow logic. Enablement makes the workflow available to users.

To verify a workflow:

1. From the Workbench shortcut bar, select **Configuration > Workflows**.

   The Workflow Workbench opens.

2. Open the workflow that you want to verify.

   The Workflow window opens

3. On the **Workflow** tab, click **Verify**.

   If the verification process uncovers no problems in the logic of the workflow, a message is displayed to indicate that no errors were detected. If the verification process uncovers problems with the workflow, its steps, or its transitions, the Verify window opens and lists the errors.

To enable a workflow:

1. Open the Workflow Workbench.
2. Open the workflow that you want to enable.

   The Workflow window opens to the **Workflow** tab.

3. For the **Enabled** option, click **Yes**.
4. Click **Save.**

Configuring Workflow Steps

After you drag a workflow step from the Workflow Step Source window to the **Layout** tab in the Workflow window, the Workflow Step window opens. Provide step information now, or later in the workflow design process.

> **Note:** "Worksheets" on page 215 contains worksheets that you can use to capture detailed information about your workflows, workflow steps, and transitions.

The Workflow Step window contains the following tabs:

- **Properties**: Displays general information about the workflow step.
- **Security**: Displays permission settings for specific individuals or groups authorized to act on a workflow step.
- **Notifications:** Define email notifications to send when a workflow step becomes eligible or after a workflow step is completed. Notifications can inform a user of a task (workflow step) to perform (such as review and approve a new request). Notifications can also inform a group of users of the results of a task.
- **Timeout**: Specify how long a workflow step can remain inactive before an error is generated.
- **User Data**: Product entities, such as packages, workflows, requests, and projects, include a set of standard fields that provide information about those entities. While these fields are normally sufficient for day-to-day processing, user data fields provide the ability to capture additional information specific to each organization. User data is defined under the **User Data** tab. If there are no user data fields, the **User Data** tab is disabled.
- **Results**: Lists the validation included in each workflow step, the component type, and the results.
- **Segregation of Duties**: Configure workflow steps to take into account segregation of duties, excluding the participants for a workflow step from participating in a different workflow step.
- **Display Settings**: Select a fill color for the graphical display of the selected step.

Configuring Properties of a Workflow Step

You can use the **Properties** tab in the Workflow Step window to provide general information about a workflow step.

To add general information to a workflow step:

1. From the Workbench shortcut bar, select **Configuration > Workflows**.

   The Workflow Workbench opens.

2. Open a workflow.

   The Workflow window opens.

3. Click the **Layout** tab.
4. Double-click a workflow step.

   The Workflow Step window opens.

5. Complete the fields on the **Properties** tab.
6. Click **Save**.

Configuring Security for Workflow Steps

To determine the users or groups that are authorized to act on a workflow step, you must set the permissions for the step.

To add security to a workflow step:

1. From the Workflow Workbench, open a workflow.
2. In the Workflow window, click the **Layout** tab.
3. Double-click a workflow step for which you want to configure security.

   The Workflow Step window opens.
4. Click the **Security** tab.
5. Click **New**.

   The Workflow Step Security window opens.
6. In the list at the top of the window, do one of the following:
   - To authorize security groups to act on the workflow step:
     i. Leave **Enter a Security Group Name** selected.
     ii. Use the **Security Group** auto-complete list to select one or more security groups to act on the workflow step. (You can use `Shift` or `Ctrl` to select multiple groups.)
   - To authorize users to act on the workflow step:
     i. Select **Enter a Username**.
     ii. Use the **Username** auto-complete list to select one or more users to act on the workflow step. (You can use `Shift` or `Ctrl` to select multiple usernames.)
   - To authorize users and security groups to act on the workflow step using standard tokens (that resolve to users and security groups):
     i. Select **Enter a Standard Token**.
     ii. Use the **Standard Token** auto-complete list to select a standard token to act on the workflow step.

iii. In the Workflow Step Security window, click **Add**.

The token you select determines the value displayed in the **Security Type** field.

iv. To add another token, repeat Step ii and Step iii.

- To authorize users and security groups to act on the workflow step using user-defined tokens (that resolve to users and security groups):

  i. Select **Enter a User Defined Token**.

  ii. If the token has already been defined, then in the **User Defined Token** field, type the token name. Otherwise, to open the Token Builder and define a new token that returns the resources you want to act on the workflow step, click **Tokens**.

  iii. In the **Security Type** list, select the security type to which the token resolves.

  iv. Click **Add**.

  v. To add another user-defined token, repeat Step ii through Step iii.

  vi. Click **OK**.

    - **Username**: The token resolves to a username.
    - **User ID**: The token resolves to a user ID.
    - **Security Group Name**: The token resolves to a security group name.
    - **Security Group ID**: The token resolves to a security group ID.

7. Click **OK**.
8. To add items of a different security type, repeat Step 6.
9. In the Workflow Step window, click **OK**.
10. On the **Security** tab, click **OK**.
11. In the Workflow window, click **OK**.

Configuring Notifications for Workflow Steps

You can configure notifications to be sent when a workflow step becomes eligible or after a workflow step is completed. Notifications can inform a user of a task (workflow step) to perform, such as review and approve a new request. Notifications can also inform a group of users of the results of a task (workflow step). You configure notifications on the **Notifications** tab in the Workflow Step window.

Review the Workflow Step Worksheet for notification information.

To add a notification to a workflow step:

1. From the Workbench shortcut bar, select **Configuration > Workflows**.
2. Open a workflow.
3. In the Workflow window, click the **Layout** tab.
4. Double-click a workflow step.
5. In the Workflow Step window, click the **Notifications** tab.
6. Click **New**.

   The Add Notification for Step window opens to the **Setup** tab.
7. From the **Event** list, select an event to trigger the new notification.
8. From the **Interval** list, select the time interval at which the notification is to be sent (after the trigger event occurs).
9. In the **Recipients** section, do one of the following:
   - Click **New,** and then use the Add New Recipient window to select the notification recipients (users, security groups, or tokens).
   - To specify the users or groups listed on the **Security** tab for the step as notification recipients, click **Copy Security**.
10. Click the **Message** tab.
11. Configure the body of the notification, and then click **OK**.

    The **Notifications** tab lists the new notification details. To send a different notification to other recipients for a different event, click **New,** and then repeat this process.

    You might want to send different notifications for a single workflow step if, for example:
    - A step has several possible results, which require different responses.
    - The notification content depends on the type of error encountered.
    - Depending on the type of step error that occurs, you want to notify recipients at different time intervals.
12. Click **OK**.

Configuring Setup Tabs

You can configure a workflow step to send notifications at different times, different intervals, following different events, and to different recipients.

Sending Notifications When Workflow Steps Become Eligible

To send a notification when a workflow step becomes eligible:

1. In the Workflow Step window, click the **Notifications** tab.

   See "Configuring Notifications for Workflow Steps " on page 54.

2. Click **New**.

   The Add Notification for step window opens to the **Setup** tab.

3. From the **Event** list, select **Eligible**.

4. To determine the frequency with which the notification is sent, from the **Interval** list, select a value.

   > **Note:** If you select **8:00 AM Daily M-F,** the notification will go out every morning at 8:00 AM from Monday through Friday after the step becomes eligible.
   >
   > If you select **8:00 AM Daily M-F** or **Hourly M-F,** you can send multiple notifications to a single recipient in a batch.

5. To send recipients a reminder if the event is still in effect after a given number of days:

   a. For the **Send Reminder?** option, select **Yes**.

   b. In the **Reminder Days** field, type the number of days after which, if the event is still in effect, a reminder is to be sent.

6. For **Enabled,** leave **Yes** selected.

7. To stop notification transmission after the step is no longer eligible, select the **Don't send if obsolete** checkbox.

8. In the **Recipients** section, do one of the following:

   - Click **New,** and then use the Add New Recipient window to select the notification recipients (users, security groups, or tokens).

- To specify the users or groups listed on the **Security** tab for the step as notification recipients, click **Copy Security**.

9. Click the **Message** tab.
10. Configure the body of the notification, and then click **OK**.
11. In the Workflow Step window, click **OK**.

Sending Notifications when Workflow Steps Have Specific Results

You can configure a notification to be sent when a workflow step has a specific decision or execution result.

To send notification when a workflow step has a specific result:

1. In the Workflow Step window, click the **Notifications** tab.

   See "Configuring Notifications for Workflow Steps " on page 54.
2. Click **New**.
3. In the Add Notification for Step window, click the **Setup** tab.
4. From the **Event** list, select **Specific Result**.
5. From the **Value** list, select the workflow step result to trigger the notification.

   > **Note:** The available values are determined by the workflow step source validation.
6. To determine the frequency with which the notification is sent, from the **Interval** list, select a value.

   > **Note:** If you select **8:00 AM Daily M-F,** the notification will go out every morning at 8:00 AM from Monday through Friday after the step becomes eligible.
   >
   > If you select **8:00 AM Daily M-F** or **Hourly M-F,** you can send multiple notifications to a single recipient in a batch.
7. To send recipients a reminder if the event is still in effect after a given number of days:
   a. For the **Send Reminder?** option, select **Yes**.
   b. In the **Reminder Days** field, type the number of days after which, if the event is still in effect,

a reminder is to be sent.

8. For **Enabled,** leave **Yes** selected.
9. To stop notification transmission after the step is no longer eligible, select the **Don't send if obsolete** checkbox.
10. In the **Recipients** section, do one of the following:
    - Click **New,** and then use the Add New Recipient window to select the notification recipients (users, security groups, or tokens).
    - To specify the users or groups listed on the **Security** tab for the step as notification recipients, click **Copy Security**.
11. Click the **Message** tab.
12. Configure the body of the notification, and then click **OK**.
13. In the Workflow Step window, click **OK**.

Sending Notifications When Workflow Steps Result in Specific Errors

You can configure the notification to be sent when a workflow step has a specific error. "Table 3-2. Specific errors for workflow steps" on the next page lists the possible workflow step errors.

Table 3-2. Specific errors for workflow steps

Specific Error	Meaning
No consensus	All users of all security groups, or users linked to the workflow step need to vote, and there is no consensus.
No recipients	None of the security groups linked to the workflow step have users linked to it. No user can act on the workflow step.
Timeout	The workflow step timed out. (Used for execution steps and decision steps.)
Invalid token	Invalid token used in the execution.
ORACLE error	Failed PL/SQL execution.
NULL result	No result is returned from the execution.
Invalid integer	Validation includes an invalid value in the **Integer** field.
Invalid date	Validation includes an invalid value in the **Date** field.
Command execution error	Execution engine has failed or has a problem.
Invalid Result	Execution or subworkflow has returned a result not included in the validation.
Parent closed	For wf_receive or wf_jump steps, a request expects a message from a package line that is cancelled or closed.
Child closed	For wf_receive or wf_jump steps, a package line expects a message from a request that is cancelled or closed.
No parent	For wf_receive or wf_jump steps, a request expects a message from a package line that has been deleted.
No child	For wf_receive or wf_jump steps, a package line expects a message from a request that has been deleted.
Multiple jump results	For wf_jump steps in a package Line, different result values were used to transition to the step.
Multiple Return Results	The package-level subworkflow received multiple results from package lines that traversed it.

To send a notification when a workflow step has a specific error:

1. From the Workbench shortcut bar, select **Configuration > Workflows**.
2. Open a workflow.
3. In the Workflow window, click the **Layout** tab.
4. Double-click a workflow step.

5. In the Workflow Step window, click the **Notifications** tab.
6. Click **New**.

   The Add Notification for Step window opens to the **Setup** tab.
7. From the **Event** list, select **Specific Error**.
8. From the **Error** list, select the error that you want to trigger the notification.
9. To determine the time at which the notification is sent, from the **Interval** list, select a value.

   > **Note:** If you select **8:00 AM Daily M-F,** the notification will go out every morning at 8:00 AM from Monday through Friday after the step becomes eligible.
   >
   > If you select **8:00 AM Daily M-F** or **Hourly M-F,** you can send multiple notifications to a single recipient in a batch.
10. To send recipients a reminder if the event is still in effect after a given number of days:
    a. For the **Send Reminder?** option, select **Yes**.
    b. In the **Reminder Days** field, type the number of days after which, if the event is still in effect, a reminder is to be sent.
11. For **Enabled,** leave **Yes** selected.
12. To stop notification transmission after the step is no longer eligible, select the **Don't send if obsolete** checkbox.
13. In the **Recipients** section, select the notification recipients (users, security groups, or tokens). For detailed instructions, see "Configuring Notification Recipients" on page 62.
14. Click the **Message** tab, and the configure the body of the notification. For details on how to do this, see "Configuring Message Content" on page 64.
15. Click **OK**.
16. In the Workflow Step window, click **OK**.

Specifying the Time Notifications Are Sent

Use the **Interval** field in the workflow step to specify when to send the notification. The interval determines how frequently the notification is sent.

To send the time notification are sent:

1. From the Workbench shortcut bar, select **Configuration > Workflows**.
2. Open a workflow.
3. In the Workflow window, click the **Layout** tab.
4. Double-click a workflow step.
5. In the Workflow Step window, click the **Notifications** tab.
6. Click **New**.

   The Add Notification for Step window opens.
7. Click the **Setup** tab.
8. Configure the **Interval** field as follows:
   - **8:00 AM Daily M-F.** This notification is sent every workday at 8:00 a.m. starting on the next workday after the notification event occurs.
   - **Hourly M-F.** This notification is sent every hour, starting on the next available workday after the notification event occurs.
   - **Immediate.** This notification is sent immediately.
9. Click **OK**.
10. In the Workflow Step window, click **OK**.

Sending Follow-Up Notifications (Reminders)

A reminder notification can be sent if the notification event is still true after a period of time.

For example, a reminder can be sent if a step is still Eligible after a number of days. A reminder cannot be sent if the notification event is All.

To send follow-up notifications:

1. In the Workflow Step window, click the **Notifications** tab.

   See "Configuring Notifications for Workflow Steps " on page 54. The **Notifications** tab opens.
2. Click **New**.

   The Add Notification for Step window opens.
3. Click the **Setup** tab.

4. Configure the **Interval** field, as listed in the following table.

Field Name	Description
Event	Selects any value except for All.
Send Reminder	To enable the **Reminder Days** field, select **Yes**.
Reminder Days	Type the number of days to wait before sending a reminder notification

5. Click **OK**.
6. In the Workflow Step window, click **OK**.

Configuring Notification Recipients

You must specify at least one recipient for a notification. The recipient can be a specific user, all members of a security group, or any email address.

To add a recipient to a notification:

1. In the Workflow Step window, click the **Notifications** tab.
2. Click **New**.

   The Add Notification for Step window opens.
3. On the **Setup** tab, click **New**.

   The Add New Recipient window opens.
4. Do one of the following:
   - To designate the recipients as the primary addressees, select the **To** option.
   - To copy the recipient on the notification, select the **Cc** option.
   - To blind copy the recipient on the notification, select the **Bcc** option.
5. From the list at the top right, select one of the following to specify the method of notification for recipients:
   - **Enter a Username**
   - **Enter an Email Address**
   - **Enter a Security Group**
   - **Enter a Standard Token**

- **Enter a User Defined Token**

  Selecting a value updates the value displayed in the **Recipient Type** field. For example, selecting **Enter a Security Group** changes the value to **Security Group**.

6. Type the specific value that corresponds to the recipient type.
   - To select one or more users to receive the notification, use the **Username** auto-complete list. You can use the `Ctrl` and `Shift` keys to select multiple users. Each user must have an email address specified.
   - To specify a recipient by his or her email address, in the **Email Address** box, type the email address.
   - To select one or more security groups, use the **Security Group** auto-complete list. (You can use the `Ctrl` and `Shift` keys to select multiple groups.) All enabled group members who have an email address in the database will receive the notification.
   - To select a standard token from a list of system tokens that correspond to a user, security group, or email address, use the **Standard Token** auto-complete list. The value displayed in the **Recipient Type** field indicates whether the token resolves to a user (name or ID), security group (name or ID), or email address.
   - To specify a user-defined token, in the **User Defined Token** field, type any field token that corresponds to a user, security group, or email address. Then, from the **Recipient Type** list, select the item that the token resolves to (user name or ID, security group name or ID, or email address).

   Tip: Use security groups or dynamic access (distributions) to specify the notification recipients whenever possible. Avoid specifying a list of users or an individual email address. If the list of users changes (due to a departmental or company reorganization), you would have to update it manually. If you specify a security group instead, any changes to group membership are automatically propagated throughout the workflow steps.

   Use distributions to send a notification to an unnamed resource. For example, to configure the notification to be sent to the assigned users, specify the `[REQ.ASSIGNED_TO_USERID]` token as the recipient.

7. Click **OK**.
8. From the **Setup** tab, click **OK**.

   The Workflow Step window opens.

9. Click **OK**.

The changes are added to the workflow.

Configuring Message Content

You can construct the notification's message to ensure that it contains the correct information in the format you want. For example, if a notification is sent to instruct you that a request requires your approval, the message should instruct you to log onto PPM and update the request's status. The notification should include a link (URL) to the referenced request.

To make them easier to configure and use, notifications include the following:

- Preconfigured notification templates to quickly compose your message.
- You can compose the body of message as plain text or as HTML.
- The notification can include multiple tokens that resolve to information relevant to the recipient. For example, you can include tokens for the URL to the request approval page, information on request status and priority, and emergency contacts.

To configure the message in a notification:

1. In the Workflow Step window, click the **Notifications** tab.
2. Click **New**.

   The Add Notification for Step window opens.
3. Click the **Message** tab.
4. From the **Notification Template** list, select a template to use for the notification.

   The **Body** field content is updated based on the selected template.
5. From the **Notification Format** list, select **HTML** or **Plain Text**.

   The HTML format allows more flexibility in the look and feel of the notification. You can use any HTML editor to write and test the HTML code, and then copy and paste this content to the **Body** field.
6. Select values for the **From** and **Reply to** fields.
7. Construct the body of the message.

   When constructing the body, consider using the following:

- Token for the URL to the Request Detail page. See "Table 3-3. Smart URL tokens" on the next page for a list of these tokens.
- Token for the URL to the package (Workbench or standard interface). See "Table 3-3. Smart URL tokens" on the next page for a list of these distributions.
- Tokens in the body of the message. Click **Tokens** to access the Token Builder window where you can add tokens to the message body.
- Tokens related to specific package lines or request detail fields. Add tokens that resolve information related to the individual package line or request detail field to the **Linked Token** field.

8. Click **OK**.
9. From the **Notifications** tab, click **OK**.

Using Tokens in the Message Body

You can select any of the available tokens available in the Token Builder window to include in the body of your message. However, not all tokens will resolve in all situations. As a general rule, tokens associated with the request or workflow will resolve.

Including URLs (Smart URLs)

When you receive a notification, it is often helpful to have a link to the item that requires attention. Notifications can be configured in the body of a notification to include the Web address (URL) for the following entities:

- Packages
- Requests
- Request Types
- Projects
- Tasks
- Workflows
- Validations

- Object Types
- Environments

If you are viewing your email with a Web-based mail reader (such as Microsoft Outlook), you can click the URL in the notification and be taken directly to the referenced entity.

For workflows, request types, validations, object types and environments, the notification can use the entity ID or the entity name as the parameter in the URL. This will bring you to the correct window in the Workbench and open the detail window for the specified entity.

"Table 3-3. Smart URL tokens" below lists the most commonly-used smart URL tokens for packages and requests.

Table 3-3. Smart URL tokens

Smart URL Token	Description
PACKAGE_URL	Provides a URL that loads the package details page in the standard interface.
WORKBENCH_PACKAGE_URL	Provides a URL that loads the package window in the Workbench.
REQUEST_URL	Provides a URL that loads the request details page in the standard interface.

If you use an HTML-formatted message, you must use an alternate token to provide a link to requests. (You can also use this token in plain-text formatted notifications.) The smart URL token (for HTML format) for requests is `REQUEST_ID_LINK`.

The `REQUEST_ID_LINK` token provides a link that loads the request detail page in the standard interface. This token resolves to the following format:

```
<a href="http://URL">Request Name</a>
```

In the notification, the link is displayed as a linked entry.

Configuring Timeouts for Workflow Steps

Timeouts determine how long a workflow step can remain eligible before generating an error. The **Timeout** tab in the Workflow Step window is used to set a timeout for the workflow step. See the **Timeout** field in the Workflow Step Worksheet for information on how to set the timeout.

To set timeouts for a workflow step:

1. From the Workbench shortcut bar, select **Configuration > Workflows**.
2. Open a workflow.

   The Workflow window opens.
3. Click the **Layout** tab.
4. Right-click a workflow step, and then click **Edit** on the shortcut menu.

   The Workflow Step window opens.
5. Click the **Timeout** tab.
6. Configure the timeout as follows:
   - **Use Workflow Step Source.** This setting determines the timeout for the workflow step. The **Timeout** and **Interval** fields are disabled.
   - **Specific Value.** You can type a timeout value for the workflow step based on the **Timeout Type** field value.
7. Click **Apply**.

Configuring Transitions for Workflow Steps

Transitions are the rules that logically connect workflow steps. You add transitions to a workflow to establish the direction a process should take, based on the results of a workflow step. For example, a request is typed into a request resolution system. The first step in the workflow is Review Request. From this workflow step, the request might be Approved or Not Approved. Both Approved and Not Approved are transitions from the Review Request workflow step.

Transitions are added to a workflow after a workflow step has been dragged and dropped from the Workflow Step Source window to the **Layout** tab in the Workflow window. You can choose a transition between workflow steps, based on the following workflow step results:

- The **Specific result** follows this transition, and is the default workflow step result. Specific results are based on the workflow step validation.
- All **Other results** that do not have transitions set follow this transition.
- **All results** follow this transition.
- The **Specific Error** follows this transition.

- All **Other Errors** that do not have transitions set follow this transition.
- **All Errors** follow this transition.

Adding Transitions Based on Specific Results

To add a Specific Result transition:

1. From the Workbench shortcut bar, select **Configuration > Workflows**.

   The Workflow Workbench opens.

2. Open a workflow.

   The Workflow window opens.

3. Click the **Layout** tab.
4. Right-click a workflow step, and then select **Add Transition** on the shortcut menu.
5. Select the destination workflow step for the transition.

   On the **Layout** tab, a line with an arrow tip is displayed between the workflow steps. The Define Transition and Step Transitions windows opens. The Define Transition window contains many options for defining the transition. The most common transition is Specific Results. For information on other transitions, see "Adding Transitions Not Based on Specific Results" below.

6. From the **Specific Results** list, select the transition.
7. Click **OK**.
8. In the Step Transitions window, click **Apply** or **OK**.

   To add another validation to the transition, click **New,** and then add another transition value. Click **OK** to add the transition value and close the Step Transitions window. The defined transition name is added to the transition line.

9. Click **Save**.

Adding Transitions Not Based on Specific Results

Transitions are added to a workflow after a workflow step has been dragged and dropped from the Workflow Step Source window to the **Layout** tab of the Workflow window. Specific results is the default transition value for the transition. The following lists other transition values:

- Other results
- All results
- Specific Events
- Specific Error
- Other Errors
- All Errors

Adding Transitions Based on Data in Tables

You can transition based on information stored in a table. To transition using this method, use a workflow execution step with an execution type of SQL.

When transitioning from a properly configured execution step (Execution Type = SQL Statement), transition based on a specific result. The possible results are defined in the workflow step source's validation. The values in this field are determined by an SQL query of a database table.

As with any execution step, configure this transition as an immediate or a manual step.

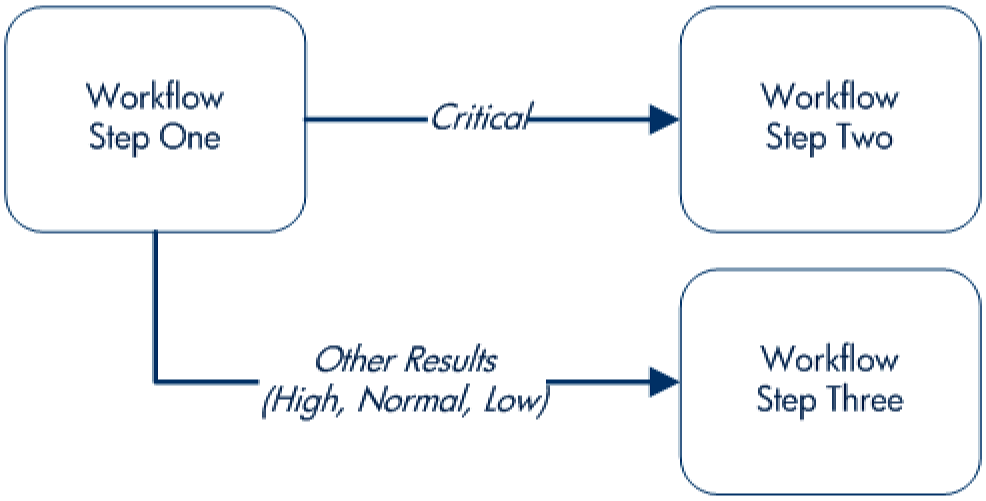
Adding Transitions Based on All But One Specific Value

You can transition based on all but one specified value. You can use Other Results when multiple transitions exit a single step. Other Results acts as the transition if none of the other explicit transition conditions are satisfied.

For example, you might want to transition all Critical requests one way and all other results (High, Normal, Low) in a different way.

To add a transition based on all but one specific value, create a transition from a workflow step based on a value in Specific Results. Create a second transition from the same workflow step. For the second transition, specify Other Results in the Define Transition window.

Figure 3-10. Transitions using other results



Adding Transitions Based on All Results

You can define a request to transition regardless of the step's actual results. For example, you may want to run a subworkflow to perform server maintenance after the on-call server contact is identified. To do this, add a transition from the Specify Contact step to the subworkflow. Because the next step in the process does not depend on the result of the step, it is appropriate to use the All Results transition. To do this, define a transition from the step, and then select **All Results**.

Consider using an All Results transition to start a subprocess. Note that you can still define transitions based on Specific Results or errors when you select **All Results**. Later, you can use an AND condition workflow step to bring the process together.

Adding Transitions Based on Errors

You can transition based on a specific error that occurs during an execution step. You can then branch the business process based on likely execution errors, such as Command Execution Error or Invalid Token (see "Table 3-4. Workflow transition errors" below). As you add a transition, select **Specific Error** option in the Define Transition window, and then select the error.

Table 3-4. Workflow transition errors

Transition Option	Meaning
Multiple Return Results	The package level subworkflow receives multiple results from package lines that traversed it.
No consensus	All users of all security groups, or users linked to the workflow step need to vote,

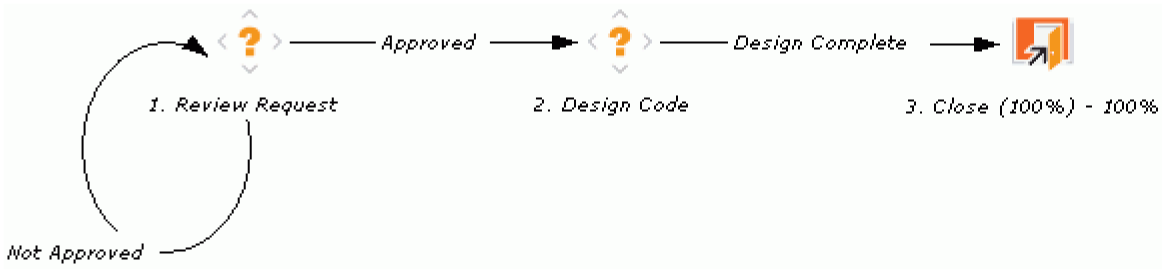
Table 3-4. Workflow transition errors, continued

Transition Option	Meaning
	and there is no consensus.
No recipients	None of the security groups linked to the workflow step has users linked to it. No user can act on the workflow step.
Timeout	The workflow step times out. Used for executions and decisions.
Invalid token	Invalid token used in the execution.
ORACLE error	Failed PL/SQL execution.
NULL result	No result is returned from the execution.
Invalid integer	Validation includes an invalid value in the **Integer** field.
Invalid date	Validation includes an invalid value in the **Date** field.
Command execution error	Execution engine has failed or has a problem.
Invalid Result	Execution or subworkflow has returned a result not included in the validation.
Parent closed	For wf_receive or wf_jump steps, a package line is expecting a message from a request that is cancelled or closed.
Child closed	For wf_receive or wf_jump steps, a request is expecting a message from a package line that is cancelled or closed.
No parent	For wf_receive or wf_jump steps, a package line is expecting a message from a request that has been deleted.
No child	For wf_receive or wf_jump steps, a request is expecting a message from a package line that has been deleted.
Multiple jump results	For wf_jump steps in a package line, different result values were used to transition to the step.

Adding Transitions Back to Same Step

You can keep the option of resetting failed execution workflow steps, rather than immediately transitioning along a failed path. This is often helpful when troubleshooting the execution ("Figure 3-11. Transitioning back to the same step" below).

Figure 3-11. Transitioning back to the same step



If the commands run successfully, they follow the Success transition path. However, if the commands fail, they do not transition out of the step because no transition has been defined for the FAILED result. The user must manually select the workflow step, and then select FAILED - RETRY. The execution is rerun.

Do not use an immediate execution workflow step if a FAILED result is feeding directly back into the execution workflow step. This results in a continual execution-failure loop.

To transition a request or package line based on a value in a field, you must:

- Configure an execution workflow step.
- Configure the transition for the execution workflow step.

To transition back to the same execution step:

1. From the Workbench shortcut bar, select **Configuration > Workflows**.

   The Workflow Workbench opens.
2. Open a workflow.
3. In the Workflow window, click the **Layout** tab.
4. Configure an immediate execution workflow step, as follows:
   a. In the Workflow Step Source window, copy an existing immediate execution workflow step.

      The Execution window opens.

b. Complete the fields, as described in the following table.

Field Name	Description
Workflow Scope	Requests for request tracking and resolution processes, Packages for deployment processes, or Release Distributions for release processes.
Execution Type	Select **Token**.
Processing Type	Select **Immediate**. Immediate steps are automated. They run the commands that are configured automatically, and move the workflow to the next eligible step without user intervention.
Validation	Create a validation with the following values. • **Succeeded** • **Failed** • **Failed - Reset** • **Failed - Rejected** For details on how to create a validation, see the HPE.
Enabled	Yes

c. Click **OK**.

5. Add the new execution workflow step to the workflow.
6. Right-click the immediate execution workflow step, and then select **Add Transition**.
7. Select several points near the execution workflow step, and then select the source workflow step.

   The Define Transition and Step Transitions window opens. The Define Transition window provides many options for defining the transition.
8. From the **Specific Results** list in the Define Transitions window, select the transition.

   The validations in the **Specific Results** field are the validations created for the execution workflow step. For example, select **Failed - Reset**.
9. Click **OK**.
10. In the Step Transitions window, click **OK**.

    The defined transition name is added to the transition line.
11. Click **Save**.

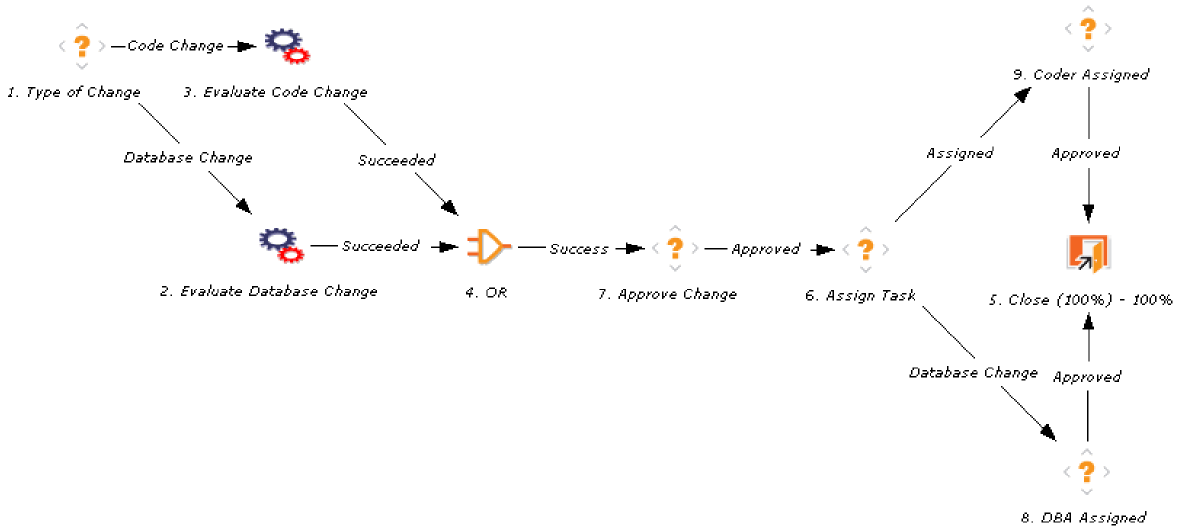
Adding Transitions Based on Previous Workflow Step Results

You can use workflow parameters to store the result of a workflow step. This value can then be used later to define a transition. The basic steps for adding a transition based on a previous workflow step result are:

1. In the Workflow window, on the **Workflow** tab, create a workflow parameter.
2. Create a token execution step to resolve the value in the workflow parameter.
3. For a workflow step, on the **Properties** tab of the Workflow Step window, in the **Workflow Parameter** field, provide the workflow parameter name.

"Figure 3-12. Add a transition based on a previous workflow step" below shows an example process. One step requires the user to route the request based on the type of change (code or database). The decision made at this step is considered later in the process to correctly route rework of the specific type.

Figure 3-12. Add a transition based on a previous workflow step



To add a transition based on a previous workflow step:

1. From the Workbench shortcut bar, select **Configuration > Workflows**.

   The Workflow Workbench opens.

2. Open a workflow.

   The Workflow window opens to the **Workflow** tab.

3. Create a workflow parameter, as follows:

   a. In the parameters section, click **Add**.

      The Workflow Parameters window opens.

   b. Complete the fields.

   c. Click **OK**.

4. Click the **Layout** tab.

5. Configure an execution workflow step with a token that resolves the value in the workflow parameter.

   > **Note:** The validation used in this step must contain the same values as the validation specified in the Type of Change decision step.

   a. From the Workflow Step Source window, copy an existing execution workflow step.

      The Execution window opens.

   b. Configure the workflow step.

   c. Click **OK**.

6. Add the new execution workflow step to the workflow, as follows:

   a. Add a workflow step to the workflow.

      The Workflow Step window opens.

   b. In the Workflow Step window, on the **Properties** tab, select the workflow parameter from the **Workflow Parameter** field.

   c. Click **OK**.

7. Add the steps and transitions, as shown in "Figure 3-12. Add a transition based on a previous workflow step" on the previous page.

8. Click **OK**.

Adding Transitions to and Removing Them from Subworkflows

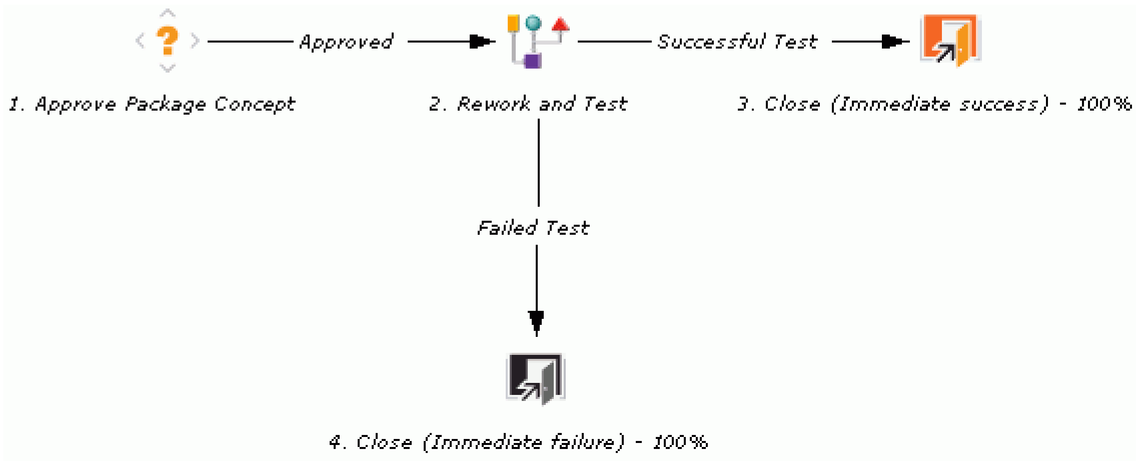
A transition to a subworkflow step is made in the same way as a transition to any other workflow step (execution, decision, or condition). The transition is graphically represented by an arrow between steps. The package line or request proceeds to the first step designated in the subworkflow definition.

When the package or request reaches the subworkflow step, it follows the path defined in that subworkflow. It either closes within that workflow (at a Close step) or returns to the parent workflow.

For a package line or request to transition back to the parent workflow, the subworkflow must contain a return step. The transitions leading into the return step must match the validation established for the subworkflow step.

In the following example, the transitions exiting the Rework and Test step (Successful Test and Failed Test) match the possible transitions entering the subworkflow's return step.

Figure 3-13. Transitioning to and from subworkflows


Approved
Successful Test
1. Approve Package Concept
2. Rework and Test
3. Close (Immediate success) - 100%
Failed Test
4. Close (Immediate failure) - 100%

Users must verify that the validation defined for the subworkflow step is synchronized with the transitions entering the return step. The subworkflow validation is defined in the Workflow window.

Users typically define the possible transitions from the subworkflow step during the subworkflow definition.

The subworkflow step validation cannot be edited if the subworkflow is used in another workflow definition. You cannot edit the subworkflow field if the subworkflow is used in another workflow definition.

Configuring Validations for Workflow Steps

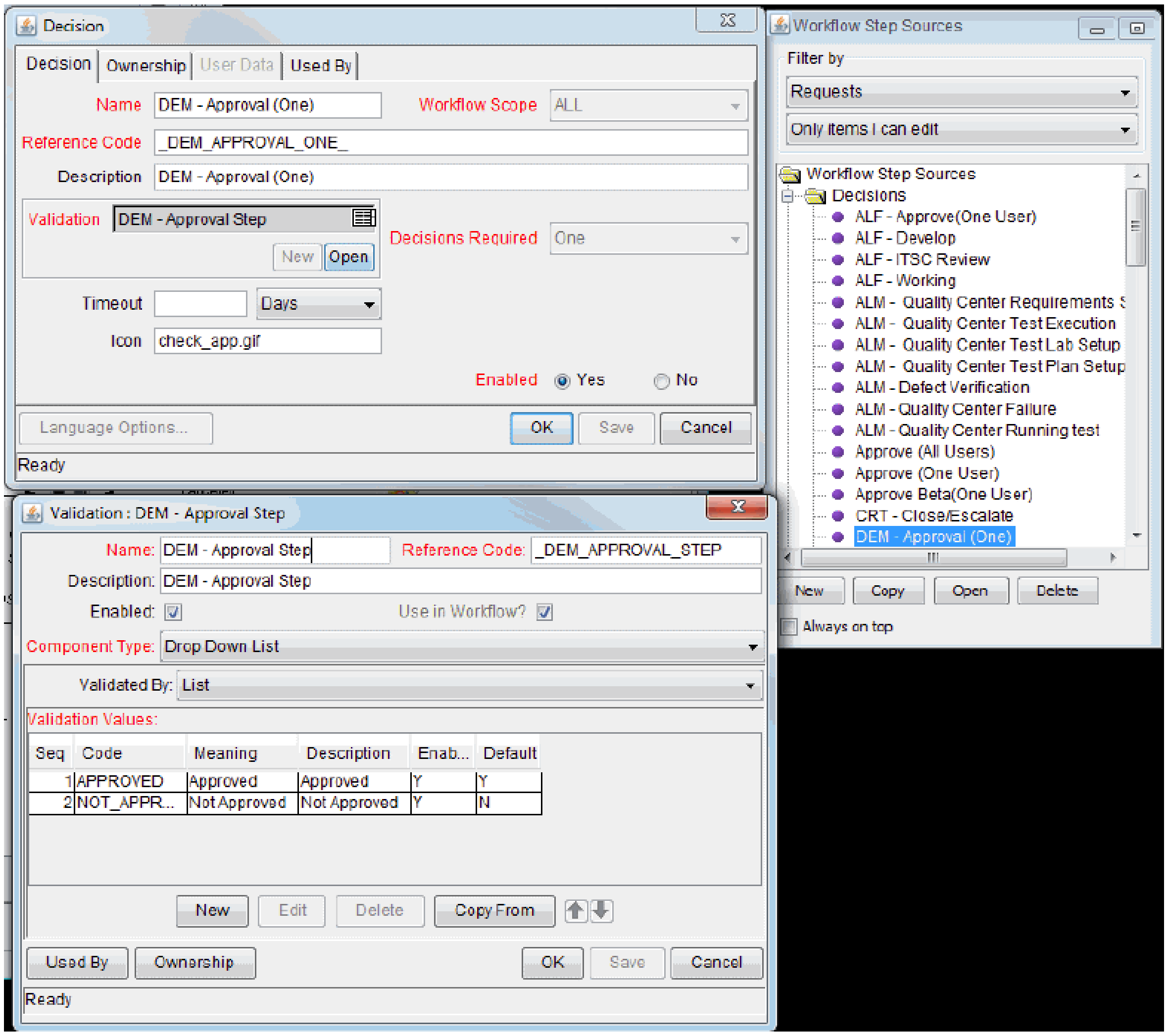
Validations determine the acceptable values for fields. Validations maintain data integrity by ensuring that the correct information is typed in a field before it is saved to the database. For workflow steps, validations ensure the correct transitions are associated with the correct workflow step.

Validations are defined for each workflow step found in the Workflow Step Source window. Opening a workflow step in the Workflow Step Sources window opens the Decision window. The Decision window contains the workflow step's default information. One piece of the default information is the validation. " Figure 3-14. Workflow step sources and validations" below shows the Decision window of the Approve (One User) decision workflow step and the validation listed in the Approval Step. In this example, the validation is WF - Approval Step. By checking the validation, WF - Approval Step has two validation values:

- Approved
- Not Approved

After a workflow step is added to a workflow, the transition can be added. Opening the Define Transition window for the workflow step, the validation values are displayed as the **Specific Results** field.

Figure 3-14. Workflow step sources and validations


Decision
Decision | Ownership | User Data | Used By
Workflow Step Sources
Language Options... OK Save Cancel
New Copy Open Delete
Validation : DEM - Approval Step
New Edit Delete Copy From
Used By Ownership OK Save Cancel

Name: DEM - Approval (One)
Workflow Scope: ALL
Reference Code: _DEM_APPROVAL_ONE_
Description: DEM - Approval (One)
Validation: DEM - Approval Step
Decisions Required: One
Timeout: Days
Icon: check_app.gif
Enabled: [✓ Yes] No
Ready

Filter by: Requests
Only items I can edit

Workflow Step Sources
- Decisions
  - ALF - Approve(One User)
  - ALF - Develop
  - ALF - ITSC Review
  - ALF - Working
  - ALM - Quality Center Requirements S
  - ALM - Quality Center Test Execution
  - ALM - Quality Center Test Lab Setup
  - ALM - Quality Center Test Plan Setup
  - ALM - Defect Verification
  - ALM - Quality Center Failure
  - ALM - Quality Center Running test
  - Approve (All Users)
  - Approve (One User)
  - Approve Beta(One User)
  - CRT - Close/Escalate
  - DEM - Approval (One)

Always on top

Name: DEM - Approval Step
Reference Code: _DEM_APPROVAL_STEP
Description: DEM - Approval Step
Enabled: ✓
Use in Workflow? ✓
Component Type: Drop Down List
Validated By: List
Validation Values:

Seq	Code	Meaning	Description	Enab...	Default
1	APPROVED	Approved	Approved	Y	Y
2	NOT_APPR...	Not Approved	Not Approved	Y	N

Ready

Validations and Execution Type Relationships

There is a correlation between the validation and the execution type. For data-dependent transitions (token, SQL, PL/SQL), the validation must contain all possible values of the query or token resolution. Otherwise, the execution step could result in a value that is not defined for the process, and the request or package line could become stuck in a workflow step.

For most built-in workflow events and executions that run commands, the validation often includes the standard workflow results (Success or Failure). If the commands or event run without error, the result of Success is returned. Otherwise, Failure is returned.

"Table 3-5. Relationship between validation and execution types" on the next page summarizes this relationship.

Table 3-5. Relationship between validation and execution types

Execution Types	Validation Notes
Built-in workflow event and workflow step commands	Typically use a variation of the WF - Standard Execution Results validation (Succeeded or Failed). A few of the workflow events have specific validation requirements: • wf_return • wf_jump • wf_receive
PL/SQL function	Validation must contain all possible values returned by the function.
Token	Validation must contain all possible values for the token.
SQL statement	Validation must contain all possible values for the SQL query. You can use the same SQL in the validation (drop-down or auto-complete list) minus the WHERE clause.

Adding Color to Workflow Steps

To make it easier to distinguish between steps in a workflow, or simply change the look and feel of a workflow in graphical view, you can apply fill color to one or more steps.

To add fill color to workflow steps:

1. In the Workflow window for an open step, click the **Layout** tab.
2. Do one of the following:
   - To select a single workflow step to fill with color, right-click the step, and then click **Edit** on the shortcut menu.
   - To select multiple steps to fill with the same color, press `Ctrl,` click all of the steps to which you want to add color, and then click **Edit** on the shortcut menu.
3. Select the **Display Settings** tab.
3. Use the color selection features on the **Swatches** tab, the **HSB** tab, or the **RGB** tab to specify a fill color for the step.
4. Click **OK**.
5. Click anywhere on the **Layout** tab.

   The selected steps fill with the color you specified.

Configuring Segregation of Duties for Workflow Steps

To comply with government or corporate policies, PPM allows you to segregate duties for workflow steps.

To set segregation of duties for a workflow step:

1. From the Workflow Workbench, open a workflow.

   The Workflow window opens.

2. Right-click a workflow step, and then click **Edit** on the shortcut menu.

   The Workflow Step window opens.

3. Click the **Segregation of Duties** tab.
4. Click **New**.

   The SOD - Source Step window opens.

5. Define a segregation source for the current workflow step, using one of the following methods:
   - To segregate this step from another workflow step, leave the **Workflow Step** option selected, and then, from the list, select the other step.
   - To prevent the user who created the package or release from acting on this step, select the **Workflow Instance Creation** option.
6. To add the segregation source to the **Segregation of Duties** tab, click **OK**.
7. In the Workflow Step window, click **OK**.

Note: All users who can act on a segregated step are prevented from acting on the current workflow step.

Integrating Object Type Commands and Workflows

Object type commands are tightly integrated with the PPM workflow engine. The commands contained in an object type are run at execution workflow steps.

It is important to note the following concepts regarding command and workflow interaction:

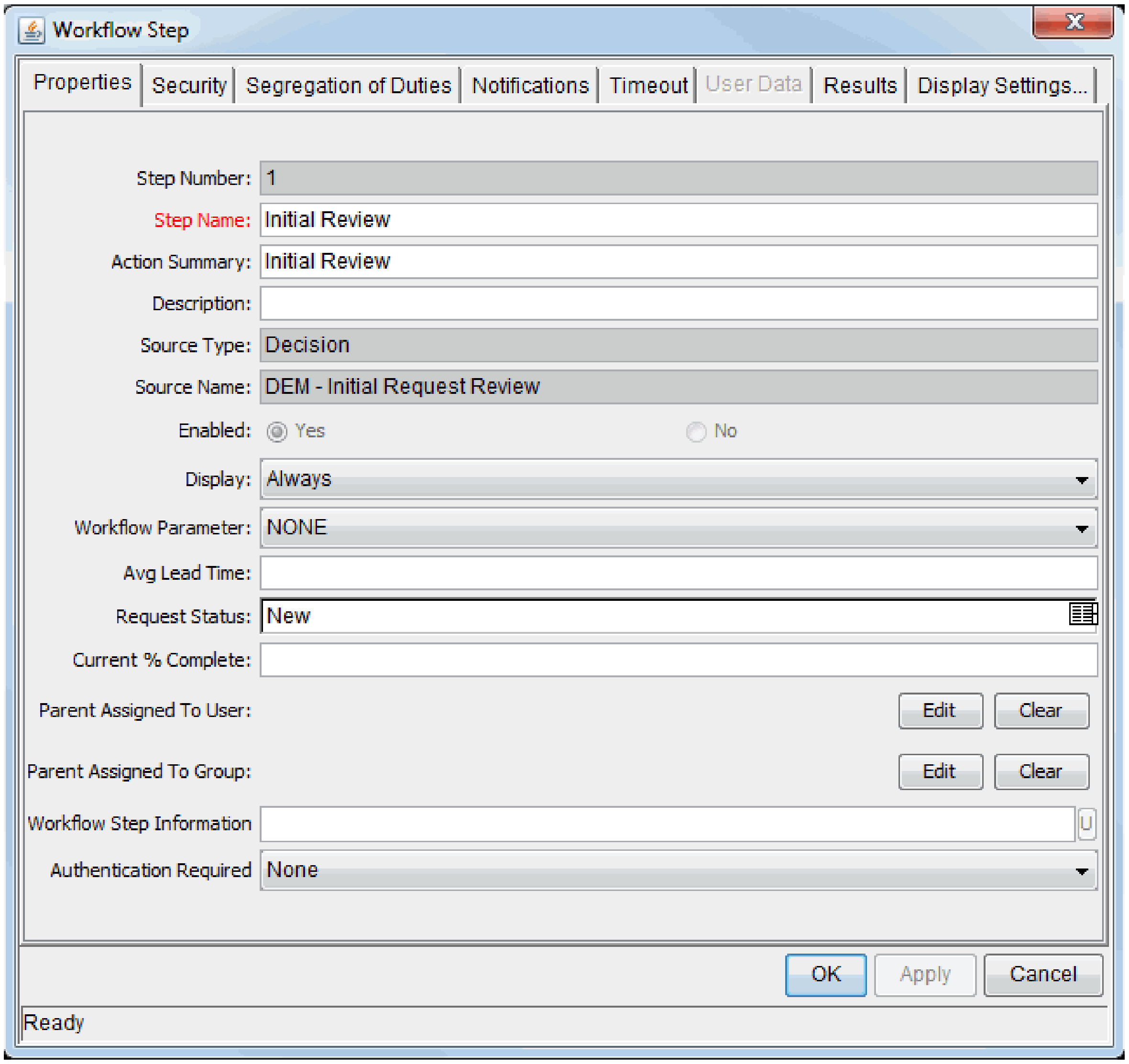
- To run object type commands at a particular workflow step, the workflow step must be configured as follows:
    - The workflow step must be an execution type step.
    - Set the workflow scope to Packages.
    - Set the execution type to Built-in Workflow Event.
    - Use the execute_object_commands workflow command.
- When the object reaches the workflow step (with Workflow Command = execute_object_commands), all object type commands whose conditions are satisfied are run in the order in which they are provided in the object type command field (**Commands** tab).
- You can configure the object type to run only certain commands at a given step.
- You can use conditional statements in the command to configure each object type command so that only certain steps (within a command) are run at a particular workflow step.

Integrating Environments and Workflows

Environments must be linked to execution workflow steps that require connection, communication, or transfer between the clients, servers and databases used in the deployment system.

Environments are specified in the Workflow Step window, accessible from the **Layout** tab in the Workflow window. You right-click a workflow step, and then click **Edit**. Select the source and destination environment, or source and destination environment groups from the fields shown in "Figure 3-15. Workflow step window for environments" on the next page.

Figure 3-15. Workflow step window for environments



Step Number:	1
Step Name:	Initial Review
Action Summary:	Initial Review
Description:	
Source Type:	Decision
Source Name:	DEM - Initial Request Review
Enabled:	[✓ Yes] No
Display:	Always
Workflow Parameter:	NONE
Avg Lead Time:	
Request Status:	New
Current % Complete:	
Parent Assigned To User:	
Parent Assigned To Group:	
Workflow Step Information	
Authentication Required	None

Ready

Choosing Source Environments Based on Application Code

Environment groups can be used to dynamically determine the source environment based on the application code for a package line. The application codes are selected based on the environments associated with the environment groups. All application codes associated with an environment are inherited by the environment group.

To enable the dynamic selection of a source environment:

1. From the Workbench shortcut bar, select **Environments > Environment Groups**.

   The Environments Workbench opens.

2. Select an environment group or create a new environment group.
3. Select the **Application Codes** tab.
4. For each application code, select **Primary Source Environment.**

   The primary source environment is selected automatically as the source environment in the workflow step when the associated application code is used in a specified package line.

5. Click **OK**.

Integrating Request and Package Workflows

Requests (Demand Management) and package workflows (Deployment Management) can be configured to work together to communicate at key points in the request and package processes. A request workflow step can *jump* to a preselected package workflow step. The package workflow step can receive the request workflow step, and then act on it to continue on to the next step in the process.

To integrate packages and requests at a level that does not rely on the workflow configuration, you can attach them to each entity as references, and then set dependencies on these references to control the behavior of the request or package. For example, you might specify a request as a predecessor to a package, so that the package cannot continue along its workflow until the request closes.

Two built-in workflow steps facilitate this cross-product workflow integration:

- Jump workflow steps (wf_jump)
- Receive workflow steps (wf_receive)

These steps are used at the points of interaction between workflows. Workflows can communicate through these jump and receive workflow step pairs.

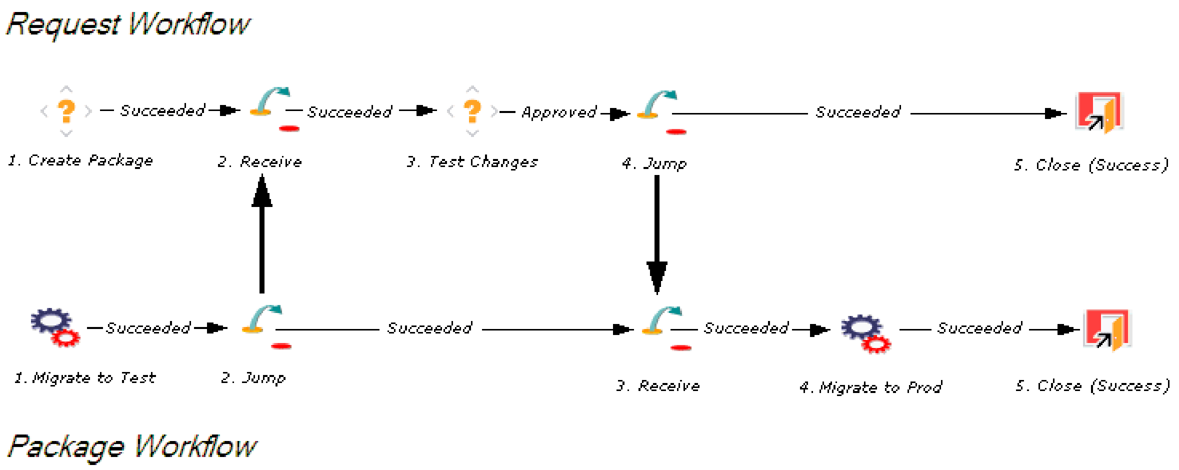
The following example shows how this cross-product workflow integration can be useful:

1. A request spawns a package for migrating new code to the production environment.
2. The newly-spawned package must go through an Approval step.
3. After the Approval step succeeds, the process jumps back to, and is received by, the request. The request then undergoes more testing and changes in the QA Environment.

4. After successfully completing the QA Test, the process jumps from the request and is received by the package.
5. Because the QA Test step was successful, the process can now migrate the code changes to the production environment.

"Figure 3-16. Jump and receive workflow step pairs" below shows this process.

Figure 3-16. Jump and receive workflow step pairs



The jump and receive workflow step pair must be carefully coordinated. Each jump workflow step must have an associated receive workflow step, linked together by a common jump and receive workflow step label defined in the Workflow Step window. The transition values used to provide and exit the jump and receive workflow steps must also be coordinated.

To establish communication between request and package workflows:

1. Set up the WF - Jump/Receive Step Labels validation for use in the Workflow Step window.

   This validation is used to join a jump and receive workflow step pair. The selected **WF - Jump/Receive Step Labels** must match in the paired jump and receive Workflow Step windows. See "Step 1. Setting Up WF - Jump/Receive Step Label Validations " on the next page.

2. Use the **wf_jump Built-in Workflow Event** to create a jump workflow step.

   See "Step 2. Generating Jump Step Sources " on the next page.

3. Use the **wf_receive Built in Workflow Event** to create a receive workflow step.

   See "Step 3. Generating Receive Step Sources" on page 86.

4. Verify that both the jump and receive workflow steps specify the same entry in the **WF - Jump/Receive Step Labels** field and that the entry matches the transition value between the two

steps.

See "Step 4. Including Jump and Receive Workflow Steps in Workflows" on page 88.

Step 1. Setting Up WF - Jump/Receive Step Label Validations

To set up the WF - Jump/Receive Step Labels validation:

1. From the Workbench shortcut bar, select **Configuration > Validations**.

   The Validation Workbench opens.

2. Click **List,** and then open the **WF - Jump/Receive Step Labels** validation.

   The Validation window opens.

3. To define a new validation value to use to link two workflows, click **New**.

   The Add Validation Value window opens.

4. Type the code, its meaning, and a description.
5. Click **OK**.
6. In the Validation window, click **Ownership**.
7. In the **Ownership** window, specify the security groups whose members can edit this validation.
8. Click **OK**.

The new validation value is now included in the **Jump/Receive Step Label** field in the Workflow Step window.

For more information about how to configure validations, see the *Commands, Tokens, and Validations Guide and Reference*.

Step 2. Generating Jump Step Sources

To create a jump step using the wf_jump built-in workflow event:

1. From the Workbench shortcut bar, select **Configuration > Workflows**.

   The Workflow Workbench opens.

2. Open a workflow.

   The Workflow window opens and the Workflow Step Sources window opens.

3. In the Workflow Step Sources window, select the **Executions** folder, and then click **New**.

   The Execution window opens.

4. In the **Name** field, type a name for the jump step.
5. From the **Workflow Scope** list, select **Requests**.

   > **Note:** Package-level subworkflows and Release Distribution workflows cannot include jump and receive steps.

6. In the **Execution Type** list, leave **Built-in Workflow Event** selected.
7. From the **Workflow Event** list, select **wf_jump**.
8. Use the **Validation** auto-complete list to select a validation to use to transition out of this workflow step.

   > **Note:** The validation values that exit the jump workflow step must match the possible validation values used to enter the jump workflow step.

9. Type all required information and any optional information you want to provide.
10. Click the **Ownership** tab.
11. Specify the security groups whose members can edit this execution workflow step.
12. Click **OK**.

    The **Executions** folder in the Workflow Step Sources window now includes the new jump workflow step.

This workflow step can now be used in any new or existing workflow within the defined workflow step scope. Remember that every jump step must have a paired receive step in another workflow.

Step 3. Generating Receive Step Sources

To create a receive step using the wf_receive built-in workflow event:

1. From the Workbench shortcut bar, select **Configuration > Workflows**.

   The Workflow Workbench opens.

2. Open a workflow.

   The Workflow window opens and the Workflow Step Sources window opens.

3. In the Workflow Step Sources window, select the **Executions** folder, and then click **New**.

   The Execution window opens.

4. In the **Name** field, type a name for the new execution step.
5. From the **Workflow Scope** list, select either **Packages** or **Requests,** depending on how you plan to apply the workflow.
6. In the **Execution Type** list, leave **Built-in Workflow Event** selected.
7. From the **Workflow Event** list, select **wf_receive**.
8. Use the **Validation** auto-complete list to select a validation to use to transition out of this workflow step.

   > **Note:** The validation values that exit the receive workflow step must match the possible validation values used to enter and exit the jump workflow step.

9. Type all required information and any optional information you want to provide.
10. Click the **Ownership** tab.
11. Select the security groups whose members can edit this execution workflow step.
12. Click **OK**.

    The **Executions** folder in the Workflow Step Sources window now includes the new receive workflow step.

This workflow step can be used in any new or existing workflow within the defined workflow scope. Keep in mind that every receive step must correspond to a jump step in another workflow.

Step 4. Including Jump and Receive Workflow Steps in Workflows

After you complete "Step 2. Generating Jump Step Sources " on page 85 and "Step 3. Generating Receive Step Sources" on page 86 you can now include them in a workflow. The **Jump/Receive Step Label** field is the key communication link between separate workflows. The communicating jump and receive workflow steps must have a matching **Jump/Receive Step Label** field entry. The **Jump/Receive Step Label** field entry must be unique for any given jump and receive workflow step pair.

To include a jump and a receive workflow step pair in a workflow:

1. From the Workbench shortcut bar, select **Configuration > Workflows.**

   The Workflow Workbench opens.

2. Open a workflow.

   The Workflow window opens and the Workflow Step Sources window opens.

3. Shrink the Workflow window, so that the Workflow window and the Workflow Step Sources window are open next to each other.

4. In the Workflow Step Sources window:

   a. Expand the **Executions** folder.

   b. Drag your jump workflow step to the **Layout** tab in the Workflow window.

      The Workflow Step window opens.

   c. From the **Jump/Receive Step Label** list, select an item.

      For example, **Migrate to Production.** This item must be the same for a paired jump and receive workflow step. The **Jump/Receive Step Label** field is the key communication link between separate workflows. The communicating jump and receive workflow steps must have a matching Jump/Receive Step Label field. The Jump/Receive Step Label field must be unique for any jump and receive pair.

      From the **Jump/Receive Step Label** list, select an item.

   d. Type any additional workflow step information you want to provide, and then click **OK**.

5. In the Workflow Step Sources window:

   a. Expand the **Executions** folder.

   b. Drag your receive workflow step to the **Layout** tab in the workflow window.

      The Workflow Step window opens.

   c. In the **Jump/Receive Step Label** field, select an item.

      For example, **Migrate to Production**. This item must be the same for a paired jump and receive workflow step. The **Jump/Receive Step Label** field is the key communication link between separate workflows. The communicating jump and receive workflow steps must have matching jump/receive step labels. The **Jump/Receive Step Label** field value must be unique for any jump and receive pair.

   d. Type any additional workflow step information you want to provide, and then click **OK**.

6. Add a transition between the jump workflow step and the receive workflow step.

   The transition must be set to the **Jump/Receive Step Label** field value you selected (for example **Migrate to Production**).

7. To save and close the workflow, click **Save**.

Chapter 4: Configuring Workflow Components

- "Overview of Workflow Step Sources" below
- "Overview of Creating Workflow Step Sources " on page 92
- "Creating Decision Workflow Step Sources " on page 93
- "Creating Execution Workflow Step Sources" on page 96
- "Creating Subworkflow Workflow Step Sources " on page 108
- "Using Workflow Parameters" on page 109
- "Modifying Workflows Already In Use" on page 111

Overview of Workflow Step Sources

PPM includes a number of standard workflow step sources that you can add to a workflow. These sources are preconfigured with standard validations (transition values), workflow events, and workflow scope. These available steps specify the following common attributes, which are expected to remain consistent across all workflows which use that step source:

- Validation associated with the step (and, thus, the list of valid transition values out of the step).
- Voting requirements of the step.
- Default timeout value for the step. (You can configure a unique timeout value for each step.)
- Icon used for the step within the graphical layout.

Browse through all of the workflow step sources using the Available Workflow Steps window in the Workflow Workbench. If a step source that meets the process requirements is not available, one needs to be created.

If PPM has a workflow step source that meets the process requirements, you can copy and rename it. This can save configuration effort and avoid user processing errors. For example, if you need a step to route a request based on whether it needs more analysis, you could copy and use the preconfigured Request Analysis workflow step source.

Copy the step source so that it can be used uniquely for the processes. This allows you to control who can edit the step source, ensuring that the process will not be inadvertently altered by another user.

Create a new step source when the step requires any of the following:

- A unique validation (transition values) leaving the step
- A unique execution in the step: PL/SQL function, token, SQL function, or workflow step commands
- A different processing type: immediate or manual
- A specific workflow scope
- A unique combination of the above settings

Configuring and Using Workflow Step Source Restrictions

The following restrictions apply to workflow step sources:

- You cannot delete a step source that is in use in a workflow.
- You cannot change a validation for a step source that is in use. If you must change the validation, copy the associated step source, and then configure a new validation.
- You must enable the workflow step source before you can add it to a workflow.
- Only add step sources to a workflow if the workflow has a matching workflow scope, or the step source scope is set to All.
- You cannot delete a workflow step in a workflow that has processed a request, package line, or release. Deleting the step would compromise data integrity. Instead, remove all transitions to and from the workflow step, and then disable the step.

Opening Workflow Workbench

To open the Workflow Workbench:

1. Log on to PPM.
2. From the menu bar, select **Open > Administration > Open Workbench**.

The Workbench opens.

3. From the shortcut bar, select **Configuration > Workflows**.

   The Workflow Workbench opens.

Overview of Creating Workflow Step Sources

You can create new decision and execution workflow step sources from the Workflow Step Sources window. Subworkflow workflow steps are created by configuring a standard workflow to be a subworkflow (see "Creating Subworkflow Workflow Step Sources " on page 108). You cannot add to, delete, or modify condition steps.

To create a new workflow step source:

1. From the Workbench shortcut bar, select **Configuration > Workflows**.
2. Open a workflow.

   The Workflow window opens and the Workflow Step Sources window opens.
3. Select the Workflow Step Sources window.
4. In the first **Filter by** field, select **Requests, Packages,** or **Release Distributions**, depending on the type of workflow.
5. In the second **Filter by** field, select **Only items I can edit**.
6. Under Workflow Step Sources, select **Decisions** or **Executions**.
7. Click **New**.

   A window that corresponds to the selected workflow step opens.
8. Type required and any optional information to define the workflow step.

   For information about how to configure a specific workflow step source, see "Creating Decision Workflow Step Sources " on the next page or "Creating Execution Workflow Step Sources" on page 96.
9. Configure the ownership of the workflow step source. For information on configuring the ownership of a workflow step source, see "Configuring Ownership of Workflow Step Sources" on the next page.
10. For **Enabled,** click **Yes**, and then click **OK**.

The new workflow step source is ready for use in any new or existing workflows with the corresponding workflow scope.

Configuring Ownership of Workflow Step Sources

As you configure a workflow step source, you can specify who can edit the workflow step source.

To configure ownership of a new workflow step source:

1. From the Workbench shortcut bar, select **Configuration > Workflows**.
2. Open a workflow.

   The Workflow window opens and the Workflow Step Sources window opens.
3. Open a decision or execution workflow step source window.

   A window that corresponds to the selected workflow step source type opens.
4. Click the **Ownership** tab.

   Use the **Ownership** tab to select the security groups that can edit this workflow step. The default is to allow all security groups who can edit workflows to edit a workflow step source.
5. Select **Only groups listed below that have the Edit Workflows Access Grant**.
6. Click **Add**.

   The Add Security Group window opens.
7. Select a security group.
8. Click **OK**.

   Only users who belong to a listed security group that can edit workflows can now edit this workflow step source.
9. From the **Ownership** tab, click **OK**.

   The new workflow step source is now listed in the Workflow Step Sources window. You can use it in any new or existing workflow with the corresponding workflow scope.

Creating Decision Workflow Step Sources

Before creating a decision workflow step source, see "Decision Workflow Step Worksheets " on page 218. It contains the information required to properly configure the workflow step source.

To create a new decision workflow step source:

1. From the Workbench shortcut bar, select **Configuration > Workflows**.
2. Open a workflow.

   The Workflow window opens and the Workflow Step Sources window opens.
3. Select the Workflow Step Sources window.
4. In the first **Filter by** field, select **Requests, Packages,** or **Release Distributions,** depending on the type of workflow.
5. Under Workflow Step Sources, select **Decisions**.
6. Click **New**.

   The Decision window opens.
7. From the **Decision** tab, type the information, as listed in the following table.

Field Name	Description
Name	The name that describes the workflow step source. The step can be renamed when added to the workflow.
Workflow Scope	Describes the type of workflow that will be using this step source. Use the list to select a workflow scope. The following lists the possible values: ◦ **ALL.** For all workflow types. ◦ **Requests.** For Demand Management request workflows. ◦ **Packages.** For Deployment Management package workflows. ◦ **Release Distributions.** For Deployment Management release workflows.
Reference Code	Automatically populated with information entered in the **Name** field.
Description	Description of the workflow step source.
Validation	Validations determine the transition values for the workflow step. Use the list to select a validation (New or Open).
Decisions Required	Defines the number of decisions required for the workflow step. Select from the following possible values: ◦ **One.** If selected, the workflow step can progress if any user who is eligible to act on this step makes a decision. ◦ **At Least One.** If selected, the workflow step waits for the voters to

Field Name	Description
	vote, and requires that your set a timeout period. If all voters designate the same value decision before the timeout period, the decision is accepted and the workflow proceeds forward. If any of the voters designate different values before the timeout period, the step immediately results in a No consensus. If at the end of the timeout, only a few voters (or only one voter) have cast their vote, the cumulative decision of the voters that voted is used. If at the end of the Timeout no one has voted, the step results in a Timeout. ◦ **All.** If selected, the workflow step waits for all of the voters to vote, makes it mandatory that all voters vote, and requires that you set a timeout period. The workflow step waits until the timeout period for the voters to vote. If all voters vote, the decision is considered. If some or none of the voters voted, the step remains open or closes due to a timeout, depending on the configuration. You can define Specific Errors in workflow steps such as Timeout and No Consensus as either Success or Failure in the Define Transition window. When using **All** or **At Least One**, all users must unanimously approve or not approve one of the validation's selections. Otherwise, the result is No Consensus.
Timeout	A timeout specifies the amount of time that a step can stay eligible for completion before completing with an error (if **Decisions Required** is **All, One,** or **At Least One)**. Timeouts can be by minute, hour, weekday or week. Timeout parameters for executions and decisions are a combination of a numerical timeout value and a timeout unit (such as weekdays). If this workflow step remains eligible for the value typed in the timeout value, the request, package, or release can be configured to send an appropriate notification. This field is often used with the **At Least One** and **All** settings for **Decisions Required**. Timeouts can be uniquely configured for each workflow step in the **Layout** tab. The timeout value specified in the workflow step source acts as the default timeout value for the step. When adding a workflow step to the workflow using this workflow step source, you can specify a different timeout value for the workflow step.
Icon	A different graphic can be specified to represent steps of this source for use on the workflow **Layout** tab. The graphic needs to exist in the icons subdirectory. All icons are in .gif format.

Field Name	Description
Enabled	The workflow step source must be enabled in order to add the workflow step to the workflow layout.

8. Click the **Ownership** tab.

9. From the **Ownership** tab, specify the security groups that can edit this workflow step.

   For detailed information about how to configure the **Ownership** tab, see "Configuring Ownership of Workflow Step Sources" on page 93.

10. Click the **User Data** tab.

    Product entities, such as packages, workflows, requests and projects, include a set of standard fields that provide information about those entities. While these fields are normally sufficient for day-to-day processing, user data fields provide the ability to capture additional information specific to each organization. User data is defined under the **User Data** tab. If there are no user data fields, the **User Data** tab is disabled.

11. Click the **Used By** tab.

The **Used By** tab displays reference information concerning the workflow step.

12. Click **OK**.

    The new workflow step source is now included in the Workflow Step Sources window. It can be used in any new or existing workflow with the corresponding workflow scope.

Creating Execution Workflow Step Sources

Before creating an execution workflow step source, see "Execution Workflow Step Worksheets " on page 216. It contains the information required to properly configure the workflow step source.

To create a new execution workflow step source:

1. From the Workbench shortcut bar, select **Configuration > Workflows.**

2. Open a workflow.

   The Workflow window opens and the Workflow Step Sources window opens.

3. Select the Workflow Step Sources window.

4. In **Filter by** field, select **Requests, Packages,** or **Release Distributions,** depending on the type of workflow.

5. Select the **Executions** folder.
6. Click **New**.

   The Execution window opens.
7. Type the information as listed in the following table.

Field Name	Description
Name	The name of the workflow step source. The step can be renamed when added to the workflow.
Workflow Scope	Describes the type of workflow that will be using this step source. Use the list to select a workflow scope. The following lists the possible values: ◦ **ALL.** For all workflow types. ◦ **Requests.** For Demand Management request workflows. ◦ **Packages.** For Deployment Management package workflows. ◦ **Release Distributions.** For Deployment Management release workflows.
Reference Code	Automatically populated with information entered in the **Name** field.
Description	Description of the step source.
Execution Type	Used to select the type of execution to be performed. Use the list to select an execution type. The following lists the possible values: ◦ **Built-in Workflow Event.** Runs a predefined command and returns its result as the result of the step. ◦ **SQL Statement.** Runs an SQL statement and returns its result as the result for the workflow step. ◦ **PL/SQL Function.** Runs a PL/SQL function and returns its result as the result for the workflow step. ◦ **Token.** Calculates the value of a token and returns its value as the result for the workflow step. ◦ **Workflow Step Commands.** Runs a set of commands, independent of an object, at a workflow step.
Workflow Event	For Execution Type Built-in Workflow Event, the specific event to perform must be selected. The available choices depend on the workflow scope selected and include: ◦ **execute_object_commands.** Runs the object type commands for a package line.

Field Name	Description
	◦ **execute_request_commands.** Runs the request type commands for a request. ◦ **create_package.** Generates an Deployment Management package. ◦ **rm_ready_for_release.** Adds the current package to an existing release. Note that the existing release must be in an Open state. ◦ **create_package_and_wait.** Generates an Deployment Management package. The create workflow step that generates the package holds it until the package is closed. ◦ **create_request.** Generates another request. ◦ **wf_close_success.** Sets the request or package line as closed with an end status of Success. ◦ **wf_close_failure.** Sets the request or package line as closed with an end status of Failed. ◦ **wf_jump.** (Deployment Management and Demand Management) Instructs the workflow to proceed to a corresponding Receive Workflow Step in another workflow. ◦ **wf_receive.** (Deployment Management and Demand Management) Instructs the workflow to receive a Jump Workflow Step and continue processing a request or package line initiated in another workflow. ◦ **wf_return.** (Deployment Management and Demand Management) Used to route a subworkflow process back to its parent workflow.
PL/SQL Function	For Execution Type PL/SQL Function, the actual function to run. The results of the function determine the outcome of the step. The results of the function must be a subset of the validation values for that workflow step.
Token	For Execution Type Token, the token to be resolved. The results determine the outcome of the workflow step.
SQL Statement	For Execution Type SQL Statement, the actual query to run. The results of the query determine the outcome of the workflow step. The results of the query must be a subset of the validation values for that step.
Workflow step commands	For Execution Type Workflow Step Commands, the actual commands to run. The commands result is either Succeeded or Failed. Use a validation with those values to enable transitioning out of the step based on the execution results.

Field Name	Description
Processing Type	Determines when the execution step is performed. Use the list to select one of the following processing types: ◦ **Immediate.** Runs the workflow step upon workflow step eligibility. ◦ **Manual.** A user is to manually perform the workflow step.
Validation	Validations determine the transition values for the workflow step. Select a validation from the list.
Timeout	The amount of time that a step is eligible before completing with an error. Timeout intervals can be in minutes, hours, weekdays or weeks. Timeout parameters for executions are a combination of a numerical timeout value and a timeout unit, such as weekdays. If this workflow step remains eligible for the value typed in the timeout value, the request, package line, or release can be configured to send a notification to interested parties. Timeouts can be uniquely configured for each workflow step in the **Layout** tab. The timeout value specified in the workflow step source acts as the default timeout value for the step. When adding a workflow step to the workflow using this workflow step source, you can specify a different timeout value for the workflow step. For executions, timeouts can be uniquely configured for the amount of time that an execution is allowed to run before completing with an error. This applies to the workflow step commands and object type commands only. Command level timeouts are set in the Command window of an object type.
Icon	You can select a different graphic to represent this steps of this workflow step source. This graphic needs to exist in the icons subdirectory. All icons are in .gif format.
Enabled	The workflow step source must be enabled in order to add it to the workflow layout.

8. Click the **Ownership** tab.

   The **Ownership** tab configures which security groups will have the ability to edit this workflow step. The default is to allow all security groups who can edit workflows to edit a workflow step source. For complete instructions on how to configure the **Ownership** tab, see "Configuring Ownership of Workflow Step Sources" on page 93.

9. Click the **User Data** tab.

   Product entities such as packages, workflows, requests and projects include a set of standard fields that provide information about those entities. While these fields are normally sufficient for day to day processing, user data fields provide the ability to capture additional information specific to each organization. User data is defined under the **User Data** tab. If there are no user data fields, the **User Data** tab is disabled.

10. Click the **Used By** tab.

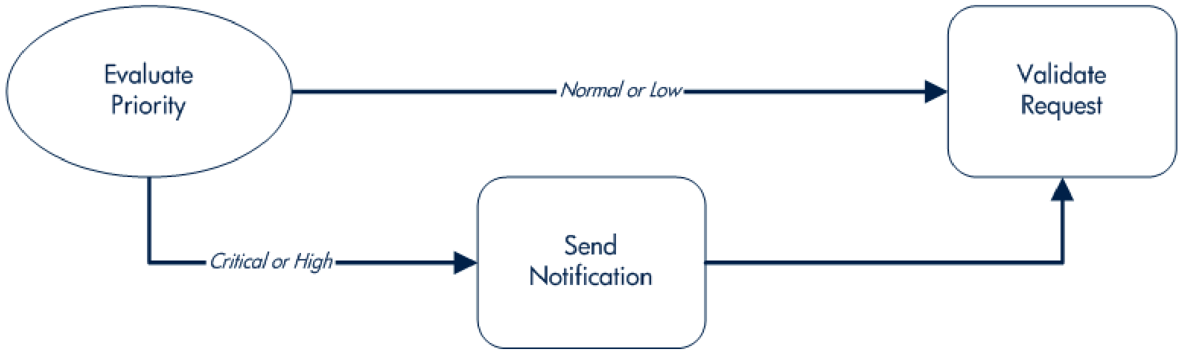
    The **Used By** tab displays reference information concerning the workflow step.

11. Click **OK**.

    The new workflow step source is now included in the Workflow Step Sources window. It can be used in any new or existing workflow with the corresponding workflow scope.

Setting Up Execution Steps

When setting up execution workflow steps, be sure to include workflow events (transitions) for both Success and Failure. If a workflow step has failed and users cannot select Failure as one of the workflow events, the workflow cannot proceed.



Defining Execution Types

Execution workflow steps are used to perform specific actions. Deployment Management provides a number of built in workflow events for processing common execution events, such as running request type commands, object type commands, and closing a request. You can also create custom executions based on SQL, PL/SQL, token resolution, and custom commands.

Executing Object Type Commands

Different objects stored in the package line require unique processing at different points in a process. For example, the commands needed to migrate a file are different than the commands needed to migrate data. Therefore, it is possible to program the commands on an object type basis. The workflow can then be configured to run the object type commands at a specific step in the process. Each package line will run its own commands, ensuring the correct execution for that object type.

Deployment Management includes the execution workflow step source DLV Execution w/ Reset that runs object type commands. Use this step source unless it does not meet the required specifications, such as validation or processing type.

To create this execution step source, make a copy of the execution workflow step source DLV Execution w/ Reset, and change the field values as defined in "Table 4-1. Execution window values to run object type commands" below.

Table 4-1. Execution window values to run object type commands

Field Name	Description
Name	Type a descriptive name for the step source
Workflow Scope	Packages
Execution Type	Built-in Workflow Event
Workflow Event	execute_object_commands
Processing Type	Manual or Immediate
Validation	WF - Standard Execution Results (This is the default selection. You can select another existing or create a new validation.)
Enabled	Yes

Closing Packages as Successful

It is possible to create an execution workflow step that closes a package line and marks the package line as a **Success.** When all package lines are closed, the package will close. Each package workflow should resolve with a closed package. Later, the packages that were closed successfully can be reported on.

This type of step is also required when integrating request and package workflows. If a package has been created using the request workflow step Create Package and Wait, the request workflow will not proceed until the package workflow has closed.

Deployment Management includes the execution workflow step sources Close (Immediate success) and Close (Manual success) that performs this task. Use one of these step sources unless they do not meet the required specifications, such as validation or processing type.

To create the execution step source, make a copy of the execution workflow step source Close (Immediate success) or Close (Manual success), and then change the field values as defined in "Table 4-2. Execution window values for closing packages as success" below.

Table 4-2. Execution window values for closing packages as success

Field Name	Description
Name	Type a descriptive name for the workflow step source
Workflow Scope	Packages
Execution Type	Built-in Workflow Event
Workflow Event	wf_close_success
Processing Type	Manual or Immediate
Validation	WF - Standard Execution Results (This is the default selection. You can select another existing or create a new validation.)
Enabled	Yes

Closing Packages as Failed

It is possible to create an execution step that closes a package and marks the package as **Failed.** Each package workflow should resolve with a closed package.

This type of step is also required when integrating request and package workflows. If a package has been created using the request workflow step Create Package and Wait, the request workflow will not proceed until the package workflow has closed.

Deployment Management includes the execution workflow step source Close (Immediate failure) that performs this task. Use this step source unless it does not meet the required specifications, such as validation or processing type.

To create the execution step source, make a copy of the execution workflow step source Close (Immediate failure), and then change the field values as defined in "Table 4-3. Execution window values for closing packages as failed" below.

Table 4-3. Execution window values for closing packages as failed

Field Name	Description
Name	Type a descriptive name for the workflow step source
Workflow Scope	Packages
Execution Type	Built-in Workflow Event
Workflow Event	wf_close_failure
Processing Type	Manual or Immediate
Validation	WF - Standard Execution Results (This is the default selection. You can select another existing or create a new validation.)
Enabled	Yes

Marking Packages Ready for Release

You can configure an execution workflow step source to feed a package into HPE.

Deployment Management includes the execution workflow step source Ready for Release that performs this task. Use this step source unless it does not meet the required specifications, such as validation or processing type.

To create the execution step source, make a copy of the execution workflow step source Ready for Release, and then change the field values listed and described in "Table 4-4. Execution window values for marking packages as Ready for Release" on the next page.

Table 4-4. Execution window values for marking packages as Ready for Release

Field Name	Description
Name	Type a descriptive name for the workflow step source
Workflow Scope	Packages
Execution Type	Built-in Workflow Event
Workflow Event	rm_ready_for_release
Processing Type	Manual or Immediate
Validation	RM - Ready for Release
Enabled	Yes

Executing PL/SQL Functions and Creating Transitions Based on Results

PL/SQL function execution workflow steps are used when a workflow needs to be routed based on the results of the PL/SQL function. A PL/SQL function execution workflow step runs a PL/SQL function and returns its results as the result of that workflow step.

Create a new execution step source with the field values as defined in "Table 4-5. Execution window values for executing PL/SQL functions" below.

Table 4-5. Execution window values for executing PL/SQL functions

Field Name	Description
Name	Type a descriptive name for the workflow step source.
Workflow Scope	Packages
Execution Type	PL/SQL Function
Processing Type	Manual or Immediate
Validation	Selects or creates a validation that includes all the possible values of the SQL query. You can create a validation validated by SQL. Use the same SQL from the execution minus the WHERE clause.
Execution	Type the SQL query.
Enabled	Yes

Executing SQL Statements and Creating Transitions Based on Results

SQL statement execution workflow steps are used when a workflow needs to be routed based on the result of a query. An SQL statement execution workflow step runs an SQL query and returns its results as the result of that workflow step.

When creating the SQL statement, you must obey the following rules:

- Use only SELECT statements
- Tokens can be used within the WHERE clause
- A query must return only one value

Create a new execution step source with the field values described in "Table 4-6. Execution window values for executing SQL statements" below.

Table 4-6. Execution window values for executing SQL statements

Field Name	Description
Name	Type a descriptive name for the workflow step source.
Workflow Scope	Packages
Execution Type	SQL Statement
Processing Type	Manual or Immediate
Validation	Selects or creates a validation that includes all the possible values of the SQL query. You can create a validation validated by SQL. Use the same SQL defined for the execution minus the WHERE clause.
Execution	Type the SQL query.
Enabled	Yes

Evaluating Tokens and Creating Transitions Based on Results

Deployment Management includes workflow execution steps that may be used to set up data-dependent rules for the routing of workflow processes. Token execution workflow steps enable a workflow to be routed based on the value of any field within a particular entity. A token execution workflow step references the value of a given token and uses that value as the result of the workflow step. A transition can be made based on the value stored in the product by using tokens in the execution workflow step.

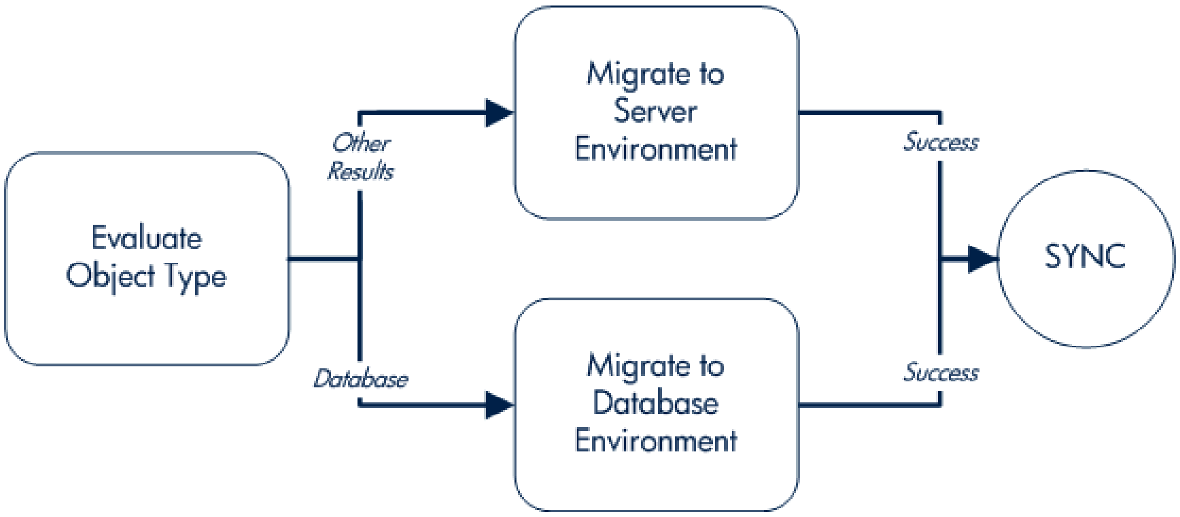
Create a new execution step source with the field values as defined in "Table 4-7. Execution window values for evaluating tokens" below.

Table 4-7. Execution window values for evaluating tokens

Field Name	Description
Name	Type a descriptive name for the workflow step source.
Workflow Scope	Packages
Execution Type	Token
Processing Type	Manual or Immediate
Validation	Selects or creates a validation that includes all of the possible values of the resolved token. For example, if the token is for the **Priority** field, use the validation for the **Priority** field here as well.
Execution	Type the token for the value that the transition will be based on.
Enabled	Yes

For example, IT needs to deploy changes to different servers depending on the type of object being deployed.

Figure 4-1. Transitioning based on a token


Evaluate Object Type
Other Results
Database
Migrate to Server Environment
Migrate to Database Environment
Success
Success
SYNC

IT decides to use an execution workflow step to automatically evaluate the object type and route the package line accordingly. To accomplish this, an execution workflow step source, Evaluate Object Type, is configured with the parameters listed in "Table 4-8. Example of execution window values for evaluating tokens" below.

Table 4-8. Example of execution window values for evaluating tokens

Field Name	Description
Name	Evaluate Object Type
Workflow Scope	Packages
Execution Type	Token
Processing Type	Immediate
Validation	DLV - Object Type - Enabled
Execution	[PKGL.OBJECT_TYPE]
Enabled	Yes

Executing Multiple System Level Commands

System level commands can be run for execution steps of the following execution type:

- Built-in Workflow Event (execute_object_commands)
- Workflow Step Commands

When either the workflow or the object type commands run at this step, the commands either succeed or fail. It may be preferable to retain the option of resetting failed execution steps, rather than immediately transitioning along a failed path. This is often helpful when troubleshooting the execution.

Creating Subworkflow Workflow Step Sources

A subworkflow is any workflow that is referenced from within another workflow. Use subworkflows to model complex business processes into logical, more manageable, and reusable subprocesses.

You can drag a subworkflow from the Workflow Step Sources window and drop it onto the **Layout** tab. When the package, request, or release reaches the subworkflow step, it follows the path defined in that subworkflow. The subworkflow either closes within that workflow or returns to the parent workflow.

Subworkflows are defined in the Workbench using the same process as when configuring a workflow. When creating a subworkflow, be sure to set the following:

- Set the **Sub-workflow** option to **Yes**.
- Make sure that the validation for the step leaving the subworkflow layout matches the subworkflow step in the parent workflow.

Subworkflows Returning to Deployment Management Workflows

Execution workflow steps can be configured to automatically return from a subworkflow to its parent Deployment Management workflow.

For a request to transition back to the parent workflow, the subworkflow must contain a return step. The transitions leading into the return step must match the validation established for the subworkflow step. You must verify that the validation defined for the subworkflow step is synchronized with the transitions entering the return step.

Deployment Management includes the Return from Subworkflow execution workflow step source, which performs this task. Use this step source unless it does not meet the required specifications, such as validation or processing type.

To create the execution step source, make a copy of the execution workflow step source Return from Subworkflow and change the field values, as defined in "Table 4-9. Execution window values for returning from subworkflows" on the next page.

Table 4-9. Execution window values for returning from subworkflows

Field Name	Description
Name	Type a descriptive name for the work step source.
Workflow Scope	Packages
Execution Type	Built-in Workflow Event
Workflow Event	wf_return
Processing Type	Manual or Immediate
Validation	WF - Standard Execution Results (This is the default selection. You can select a different validation or create a new one.)
Enabled	Yes

Using Workflow Parameters

Use workflow parameters to store the results of a workflow step. This value can then be used later to define a transition. The following lists the rules concerning workflow parameters:

- You can use the `WFI.P` token prefix to reference workflow parameters.
- You can use workflow parameters in PL/SQL and SQL workflow step executions.

Creating Workflow Parameters

To create a workflow parameter:

1. From the Workbench shortcut bar, select **Configuration > Workflows**.
2. Open a workflow.

   The Workflow window opens.
3. From the **Workflow** tab, click **Add**.

   The Workflow Parameter window opens.

4. Type information in the Workflow Parameter window as specified in the following table.

Field Name	Description
Prompt	The name of the workflow parameter.
Token	The name of the token. For example, LOOP_COUNTER.
Description	A description of the workflow parameter.
Default Value	The initial value given to the workflow parameter.

5. In the **Parameters** section of the **Workflow** tab, click **Add**.
6. Click **OK**.
7. From the **Workflow** tab, click **OK**.

Example: Building a Loop Counter Using Workflow Parameters

A workflow parameter can be used to generate a counter for the number of times a workflow step enters a state.

To build a loop counter using workflow parameters:

1. From the Workbench shortcut bar, select **Configuration > Workflows**.
2. Open a workflow.

   The Workflow window opens.
3. From the **Workflow** tab, click **Add**.

   The Workflow Parameter window opens.
4. Type the information as listed in the following table.

Field Name	Description
Prompt	Loop Counter
Token	LOOP_COUNTER

Field Name	Description
Description	Stores count
Default Value	0

5. In the Parameters section of the **Workflow** tab, click **Add**.
6. Click **OK**.
7. From the **Workflow** tab, click **OK**.
8. Create a new immediate SQL execution workflow step.

   For details on how to create an SQL execution workflow step, see "Creating Execution Workflow Step Sources" on page 96.

   Note about the new step definition:

   - The result of the SQL execution workflow step returns the result `LOOP_COUNTER + 1.` This return value is linked back into the parameter when the workflow step is generated on a workflow.
   - A validation for a numeric text field is used. This allows you to use <=, <, >=, and > comparisons in transitions off this step.
9. Add the workflow step to a workflow and choose the new workflow parameter Loop Counter.

   By choosing Loop Count, the workflow engine is told to assign the result of "select loop counter val + 1" from dual back into the Loop Counter parameter.

You can now add transitions to and from the new loop counter step. For example, you add the loop counter each time an execution fails. If the execution fails three times, a notification is sent to the user. If the execution fails five times, management is notified.

Modifying Workflows Already In Use

Workflows can be modified while they are going through their workflow steps after a package or request has been initiated. These modifications include adding new workflow steps, as well as changing the transitions, security assignments and notifications from within the workflow.

You can make changes to workflows that are currently in use with the same procedures and windows that you used to define the workflows. All of these procedures are performed in the Workflow Workbench window.

When modifying workflows that are being used, rules exist for which entities can be added, changed, deleted or renamed. These rules are described in "Table 4-10. Rules for modifying production workflows" below.

Table 4-10. Rules for modifying production workflows

Entity	Procedure
Transitions Security Notifications Workflow Steps Workflow Parameters	You can change any of these entities or add them to a workflow that is in use.
Transitions Security Notifications Workflow Parameters	You can delete any of these entities from a workflow in use.
Workflow Steps	You cannot delete this entity from a workflow in use, but you can rename it. You can delete transitions coming into or going out of a workflow step to effectively remove it from the workflow.

If a workflow that is in use is changed and saved, the changes take effect immediately. Any changes made to workflow steps are applied to all open package lines, requests, releases, and distributions.

Changes to a workflow can have undesirable effects on requests or packages currently in progress and are using that workflow.

When modifying a workflow that is in use, this can disrupt the normal flow in and out of the workflow and prevent it from reaching completion. For example, removing a transition from a workflow step may result in the requests or package lines being stuck in that workflow step.

Performance Considerations

Updating the security in an existing workflow step with a specific configuration can impact system performance. For example, when adding dynamic security to a step based on a standard or user defined token, product database tables are updated to handle this new configuration. Due to the scope of these database changes, Database Statistics need to be rerun on your database.

> **Note:** This also applies when migrating a workflow with these types of changes into an instance of the PPM.

Migrating a workflow with these types of changes into an instance of the PPM can also impact system performance. Product database tables must be updated to handle this new workflow and, due to the scope of these database changes, Database Statistics need to be rerun on your database.

For information about how to run database statistics on your database, see the document *Installation and Administration Guide*. For help with this procedure, contact your application administrator.

Copying and Testing Trial Versions of Workflows

Before modifying a workflow that is being used, do the following:

1. Make a copy of the original workflow.
2. Modify the copied version of the workflow with the changed workflow steps.
3. Test the modified version of the workflow to make sure it works correctly.
4. Determine if the workflow step is in use.To determine which steps are currently eligible, remove the incoming transition to the step that will be deleted and run the following reports. The reports will indicate if the step to be deleted is eligible for action by package lines or requests.
    - To determine when the requests have flowed out of a workflow step, run the Workflow Detail Report. This report indicates if the step to delete is eligible for user action or has been completed.
    - To determine if any package lines are eligible for user action in a workflow, run the Packages Pending Report.

    You are ready to make the same changes to the original workflow.

Modifying Production Workflows

The final step in modifying workflows already in use is to modify the production workflow. The following sections offer guidance on how to modify the production workflow.

Disabling Workflow Steps

As mentioned in "Table 4-10. Rules for modifying production workflows" on page 112, a step cannot be deleted from a workflow when it is in use. It can only be disabled. However, you may want to change the process. Any changes to the process must be reflected in the workflow. This may require disabling existing steps and adding new steps.

1. Remove transitions to the existing workflow step you no longer want to use.
2. Add a new workflow step to the workflow.
3. Redirect the transitions to the new workflow step.

Redirecting Workflows

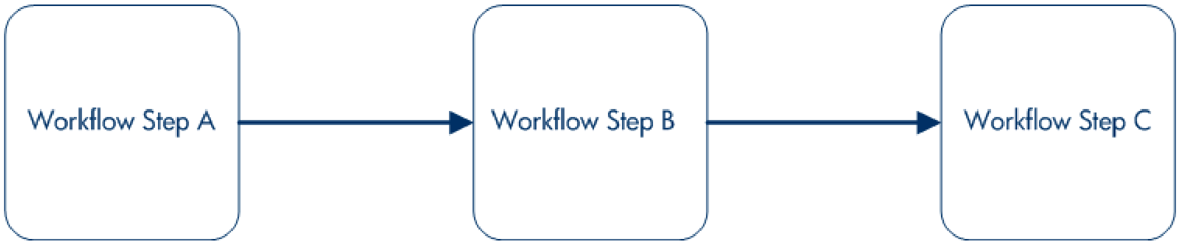
When disabling a workflow step that is currently **Eligible** for user action, the requests or package lines in that step will become stuck. Because the step is now disabled, the user cannot take action on it and will not be able to progress any further through the workflow.

The outgoing transition to be deleted is still intact, so the eligible package lines and requests will eventually be acted upon and flow out of the workflow step.

Add a new workflow step to the workflow and redirect the transitions to that new workflow step so that the movement of package lines and requests avoids the disabled step and is not interrupted.

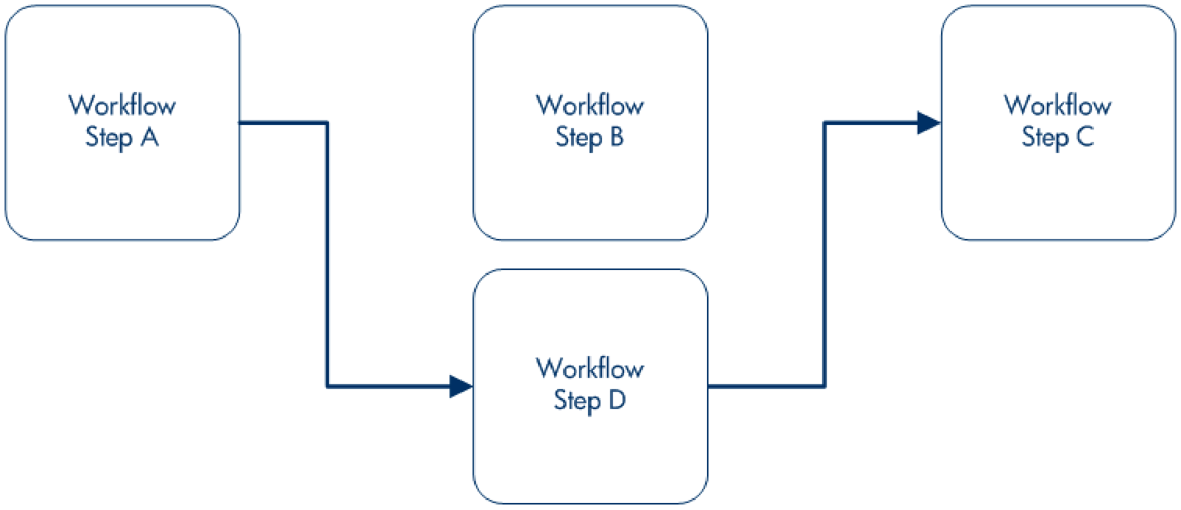
For example, consider a workflow where you wanted to disable workflow step B in the sequence, as shown in "Figure 4-2. Redirecting the workflow, Step 1" below.

Figure 4-2. Redirecting the workflow, Step 1



After removing the incoming and outgoing transitions to B, add a new workflow step D which will connect steps A and C, letting the workflow continue to process requests or package lines (see "Figure 4-3. Redirecting the workflow, Step 2" on the next page).

Figure 4-3. Redirecting the workflow, Step 2



Run the applicable reports again to be sure there are no entities eligible for action by the user in the step that was disabled.

Moving Requests or Packages Out of Steps

If the requests or packages are stuck in a step after a transition has been removed from a workflow in use, add the deleted transition back to the workflow. After the requests or packages have flowed out of the step, delete the transition again.

Chapter 5: Configuring Object Types

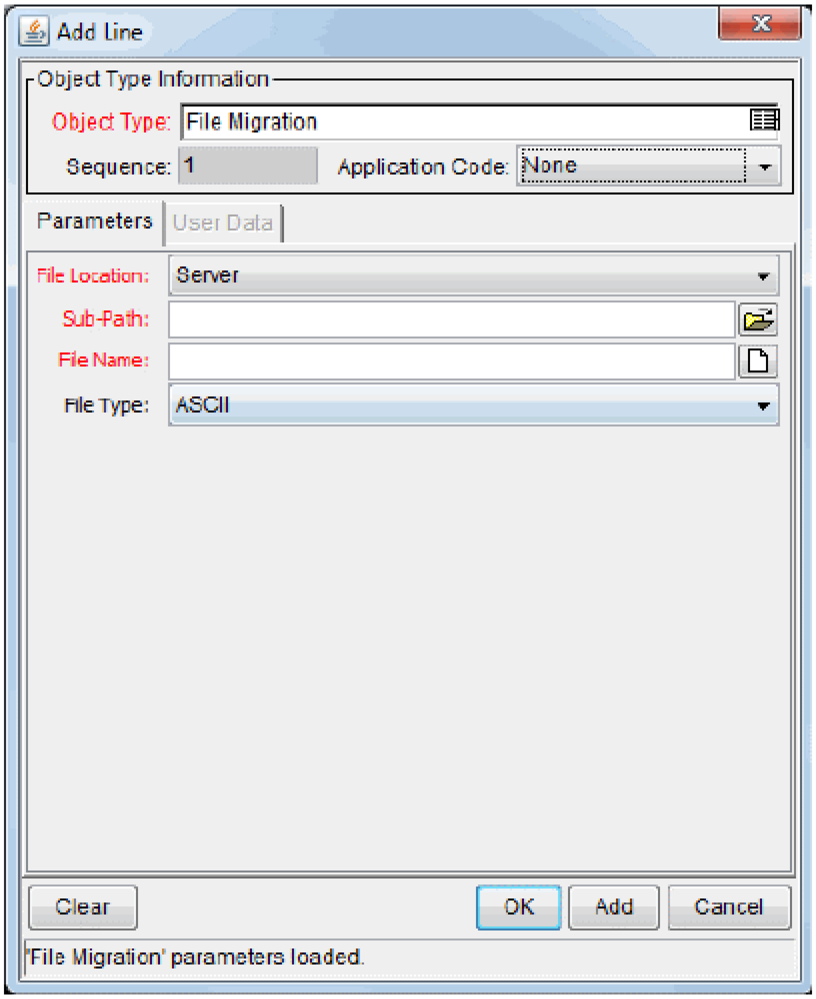
- "Overview of Object Types" below
- "Opening Object Type Workbench" on page 119
- "Configuring General Information for Object Types" on page 119
- "Creating Object Type Fields" on page 120
- "Configuring Layouts for Object Types " on page 129
- "Configuring Commands for Object Types " on page 132
- "Configuring Ownership for Object Types " on page 135

Overview of Object Types

Deployment Management automates complex software deployment processes. While PPM workflows define the process, object types are used to define the technical steps required to deploy a particular object. For example, a File Migration object type may contain the information and commands required to transfer a file from one machine to another, while an SQL script object type might address the migration and execution of database scripts.

Object types are used to create and process packages. Each package line in a package consists of one object of a specific type. To define a package line, the user selects an object type in the Add Line window in the package screen (see "Figure 5-1. Example of an object type" on the next page). The fields required to process that object type are dynamically displayed.

Figure 5-1. Example of an object type

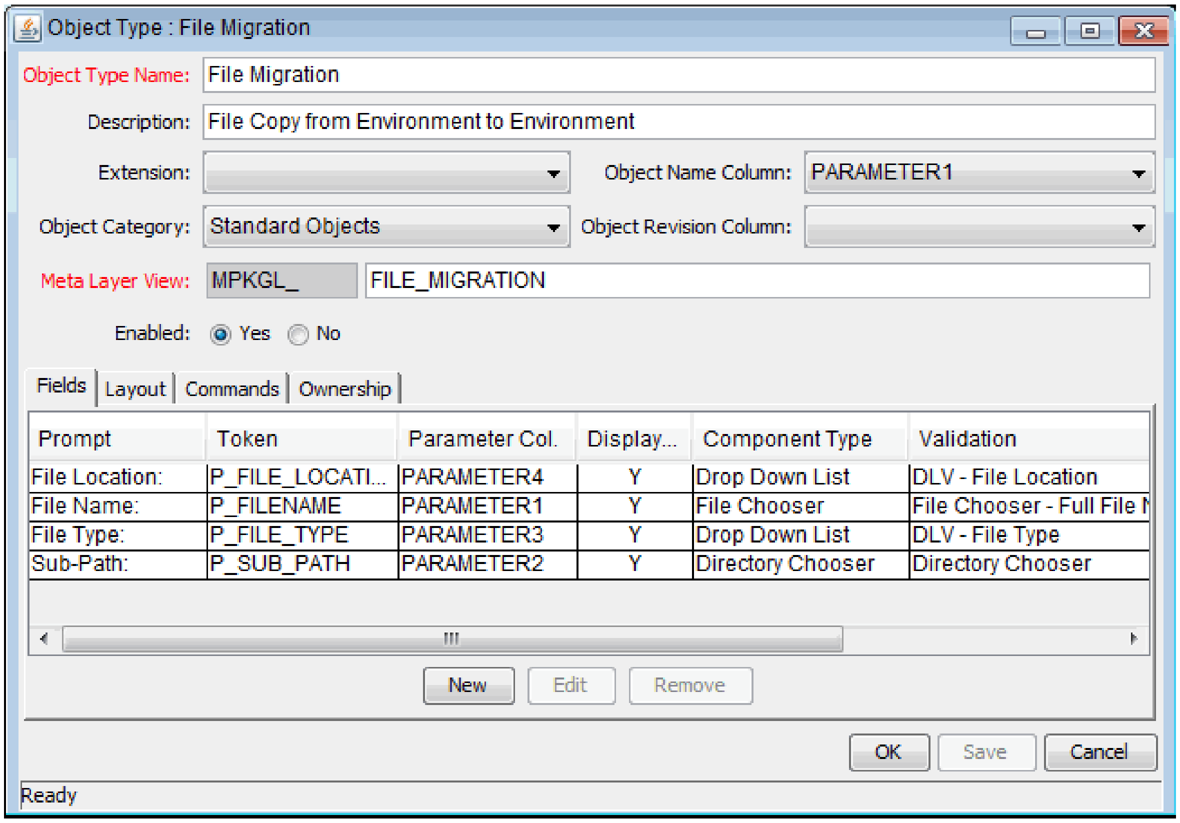


Object Type: File Migration
Sequence: 1
Application Code: None
File Location: Server
Sub-Path:
File Name:
File Type: ASCII

'File Migration' parameters loaded.

You use the Object Type window to create and configure object types (see "Figure 5-2. Object Type window" on the next page).

Figure 5-2. Object Type window



Object Type Name: File Migration

Description: File Copy from Environment to Environment

Extension:

Object Name Column: PARAMETER1

Object Category: Standard Objects

Object Revision Column:

Meta Layer View: MPKGL_ FILE_MIGRATION

Enabled: [✓ Yes] No

Prompt	Token	Parameter Col.	Display...	Component Type	Validation
File Location:	P_FILE_LOCATI...	PARAMETER4	Y	Drop Down List	DLV - File Location
File Name:	P_FILENAME	PARAMETER1	Y	File Chooser	File Chooser - Full File N
File Type:	P_FILE_TYPE	PARAMETER3	Y	Drop Down List	DLV - File Type
Sub-Path:	P_SUB_PATH	PARAMETER2	Y	Directory Chooser	Directory Chooser

Ready

The following is a list of the main components of an object type:

- **General information**: Basic information concerning the object type, such as the object type name and the object type category. See "Configuring General Information for Object Types" on the next page.
- **Fields**: Every object type includes fields. The **Fields** tab is used to create fields for the object type. See "Creating Object Type Fields" on page 120.
- **Layout**: After all the fields are created for an object type, the layout of those fields can be configured using the **Layout** tab. See "Configuring Layouts for Object Types " on page 129.
- **Commands**: Can be used to control certain behavior of object type fields. At specific workflow execution steps in a deployment process, it is possible to run commands stored in the object type. These commands can manipulate data inside an object type field. This provides an advantage over the defaulting features on the **Fields** tab, which can only default based on a single parameter stored on the same object type. See "Configuring Commands for Object Types " on page 132.
- **OraApps**: Deployment Management Extension for Oracle E-Business Suite requires specially-configured object types. If Deployment Management Extension for Oracle E-Business Suite is not installed, the **OraApps** tab is disabled.

- **Ownership**: Configures who can edit the object type. See "Configuring Ownership for Object Types " on page 135.

Opening Object Type Workbench

To open the Object Type Workbench:

1. Log on to PPM.
2. From the menu bar, select **Open > Administration > Open Workbench**.

   The PPM Workbench opens.
3. From the shortcut bar, select **Deployment Mgmt > Object Types**.

   The Object Type Workbench window opens.

Configuring General Information for Object Types

To configure the general information of an object type:

1. From the Workbench shortcut bar, select **Deployment Mgmt > Object Types**.

   The Object Type Workbench opens.
2. Open an object type.

   The Object Type window opens.
3. Type the information listed in the following table.

Field Name	Description
Object Type Name	The name of the object type.
Description	A useful description of how the object type is used.
Extension	For object types created for an Deployment Management extension. Select the extension.
Object Category	Object categories provide a way of categorizing object types in Deployment Management. The default values are **Custom Object**

Field Name	Description
	and **Standard Objects**. You can configure the values by changing the validation. Validation: DLV - Object Category - Enabled
Object Name Column	When defining an object type, this field represents the name of the object in a package line. This field should link to a field defined for the object type. Typically, selecting an entry for the **Object Name** column is done after all of the object type fields are created. For more information, see "Setting Object Names" on page 131.
Object Revision Column	If integrating with a version control system using this object type, indicates which parameter field represents the revision number for the object. The object revisions column should be set after defining the object type fields. It is also possible to create a field in the object type to represent the revision number of the object. This field will often be a numeric text field. The deployment process can then be configured to consider the object revision number when processing the package. Typically, selecting an entry for the **Object Revision** column is done after all the object type fields are created. For more information, see "Setting Object Names" on page 131.
Meta Layer View	Meta layer views relate information specific to PPM.
Enabled	Indicates whether or not the object type is available to PPM. Selecting **Yes** makes the object type available to the system.

4. To save the changes and close the Object Type window, click **OK**. To save the changes and leave the Object Type window open, click **Save**.

Creating Object Type Fields

You can configure each field to behave in a certain way using the Field configuration window in the Object Type window, click a field to open the Field window (see "Figure 5-3. Field window" on page 122). The Field window contains three tabs:

- **Attributes**: Used to set basic display, edit, and requirement field properties.
- **Default**: Used to set the value in the field.

- **Dependencies**: Used to set clearing, display, and requirement field properties based on values in other object type fields.

From the Field window, configure whether the field:

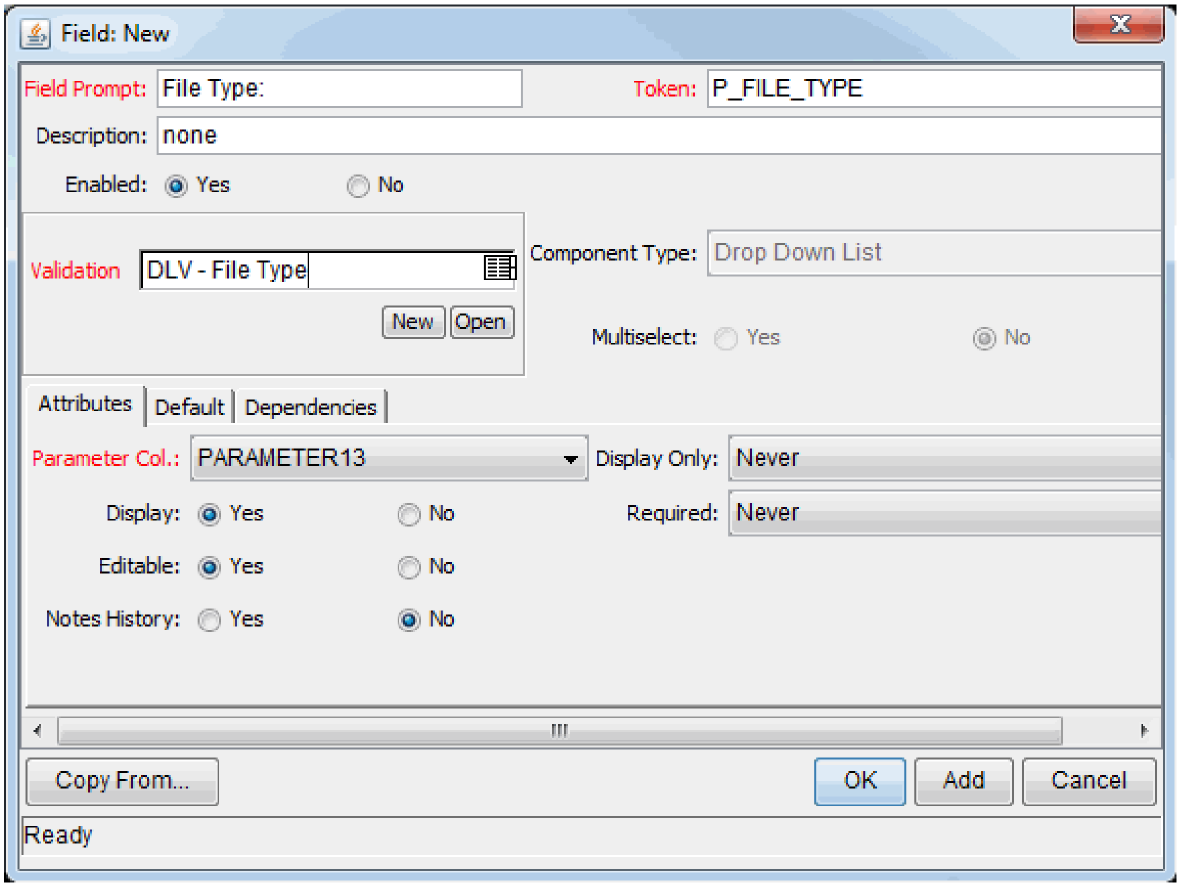
- Is displayed (for example, you might need to store a value for later use in commands, but do not want to clutter the package line)
- Can be edited under different circumstances
- Is required under different circumstances
- Defaults to a certain value
- Is dependent on values in other fields in the object type
  - Clear the field's value when another field changes
  - Display only when another field has a specific value
  - Required only when another field has a specific value

Overview of Object Type Field Validations

When configuring the object type, you can specify a different validation for each field. The validation dictates the possible values that can be provided in the field and the field type (such as text field, drop-down list, or date field).

For example, XYZ Corporation requires a **File Type** field to capture the objects to be deployed. On their object type, they set up the field as follows.

Figure 5-3. Field window



Field: New

Field Prompt:	File Type:	Token:	P_FILE_TYPE
Description:	none		
Enabled:	[✓ Yes] No		
Validation	DLV - File Type	Component Type:	Drop Down List
		Multiselect:	Yes [✓ No]

Attributes | Default | Dependencies

Parameter Col.:	PARAMETER13	Display Only:	Never
Display:	[✓ Yes] No	Required:	Never
Editable:	[✓ Yes] No		
Notes History:	Yes [✓ No]		

Ready

The validation is validated by a list. This is an appropriate choice because the selection is not expected to change.

Selecting Validations

If no validation meets the necessary requirements (such as having the applicable values), you must create one. You can also select a validation that has been configured for use at your site.

Be careful if you use a validation that is configured for use in another process. If the owner of the other process changes the validation, the validation also changes for the items in your process. Consider copying the existing validation to create a new one. You can then control who can alter the validation values by setting validation ownership.

Creating Object Type Fields

To create an object type field:

1. From the Workbench shortcut bar, select **Deployment Mgmt > Object Types**.

   The Object Type Workbench window opens.

2. Open an object type.

   The Object Type window opens.

3. Click **New**.

   The Field window opens.

4. Complete the fields in the Field window as specified in the following table.

Field Name	Description
Field Prompt	Type the name of the new field.
Token	Type a unique name for the token.
Description	Type a description of the new field.
Enabled	Select **Yes** to make the field available to the system.

5. Select a **Validation**.

   See "Overview of Object Type Field Validations " on page 121 for information on choosing a validation. If no existing validation meets the requirements, create one that does.

6. Configure field behavior.

   This consists of setting options in the field's **Attributes, Default,** and **Dependencies** tabs. Note that some field behavior is dependent on other object type fields. You may need to revisit this step after you create the other fields in the object type.

   To configure field behavior:

a. Complete the fields in the **Attributes** tab, as follows:

Field Name	Description
Parameter Col.	Indicates the internal database column that the field value will be stored in. These values are stored in the corresponding column in the package lines table for each line of the given object type. Information can be stored in up to 30 columns to allow for up to 30 fields/parameters. No two fields in an object type can use the same column.
Display Only	Indicates whether to display a field using the following options: • **Always** • **Never** • **Use Dependency Rules** Select **Use Dependency Rules** to use the logic defined on the **Dependencies** tab.
Display	Indicates if this field is visible in the package line region of the package window.
Required	Indicates if a value needs to be specified for this field using the following options: • **Always** • **Never** • **Use Dependency Rules** Select **Use Dependency Rules** to use the logic defined in the **Dependencies** tab.
Editable	After a package line has been typed and submitted, it starts moving through its workflow. This attribute determines if the field can still be updated. For example, it may be necessary to ensure that a **Filename** field is not editable after package lines of file object type start getting processed.
Notes History	Notes stored for historical purposes.

b. Complete the fields in the **Default** tab.

Field Name	Description
Default Type	Determines whether the field has a default value. Either

Field Name	Description
	default the field with a constant value, default it from the value in another field, or default to a parameter.
Visible Value	If a default type of constant is selected, type the constant value here.
Depends On	If defaulting from another field, type the token name of that field. At runtime, when using this object type, every time a value is typed or updated in the source field, it is automatically typed or updated in this destination field.

c. Complete the fields on the **Dependencies** tab. You must have selected the **Use Dependency Rules** from the **Attributes** tab.

Field Name	Description
Clear When The Following Changes	Indicates that the current field is to be cleared if the specified field changes.
Display Only When	Indicates that the current field is to be editable only if certain logical criteria are met. This field functions with the following adjacent fields: • A list containing the logical qualifier • A text field
Required When	Indicates that the current field should be required when certain logical criteria are satisfied. This field functions with the following adjacent fields: • A list containing logical qualifier • A text field

7. Click **OK**.

   The changes to the object type are saved.

Configuring Field Dependencies

Field behavior and properties can be linked to the value of other fields defined for that entity. For example, an object type field can become required when the value in another field in that object type equals the text Critical.

A field can be configured to:

- Clear when another field changes.
- Become read only when another field meets a logical condition defined in "Table 5-1. Field dependencies" below.
- Become required when another field meets a logical condition defined in "Table 5-2. Example conditions" on page 135.

Table 5-1. Field dependencies

Logical qualifier	Description
like	Looks for the specified value to find any matching values in the chosen field.
not like	Looks for values in the chosen field that are not equal to the specified value.
is equal to	Looks for an exact match of the specified value to the contents of the field chosen.
is not equal to	Is true when there are no results exactly matching the value of the field contents.
is null	Is true when the field selected is blank.
is not null	Is true when the field selected is not blank.
is greater than	Looks for a numerical value in excess of the value typed in the value field.
is less than	Looks for a numerical value below the value typed in the value field.
is less than equal to	Looks for a numerical value below, or the same as, the value typed in the value field.
is greater than equal to	Looks for a numerical value in excess of, or the same as, the value typed in the value field.

To configure a field dependency:

1. From the Workbench shortcut bar, select **Deployment Mgmt > Object Types.**

   The Object Type Workbench window opens.

2. Open an object type.

   The Object Type window opens.

3. Select and edit an existing field or create a new field.

   The Field window opens.

4. Click the **Dependencies** tab.

5. Set the field dependencies, using one of the following options:

- In the **Clear When the Following Changes** field, select a field name to indicate that the current field is to be cleared if the selected field changes.
- In the **Display Only When** field, select a field name to indicate that, if certain logical criteria are satisfied, the current field is to be read-only.

  This field functions with two adjacent lists that contain logical qualifier, and a field which dynamically changes to a date field, list, or text field, depending on the validation of the selected field.
- In the **Required When** field, select a field name to indicate that the current field is to be required if certain logical criteria are satisfied.

  This field functions with two adjacent lists that contain a logical qualifier, and a field that dynamically changes to a date field, list, or text field, depending on the validation of the selected field.

6. Click **OK**.

   This adds the field dependencies to the **Fields** tab of the Object Type window and closes the field window.
7. Click **OK**.

Copying Object Type Fields

To copy an object type field:

1. From the Workbench shortcut bar, select **Deployment Mgmt > Object Types**.

   The Object Type Workbench opens.
2. Open an object type.

   The Object Type window opens to the **Fields** tab.
3. Click **New**.

   The Field window opens.
4. Click **Copy From**.

   The Field Selection window opens.
5. Query for the field you want to copy.

You can use several criteria, including token name or field prompt, to query fields. You can perform more complex queries. For example, you can list all fields that reference a given validation or that a specific entity uses. Because of the number of PPM fields, limit the list of fields by one or more of the query criteria.

6. In the Field Selection window, select the desired field, and then click **Copy**.

   The Field Selection window closes and the definition of the selected field is copied to the New Field window.

7. In the New Field window, modify the fields as required.
8. Click **OK**.

   The new field is added to the **Fields** tab.

9. Click **OK**.

Editing Object Type Fields

Changes to fields for object types already used by existing package lines can have a significant impact. For example, if the column where a field value gets stored is changed, all existing package lines for this object type will now have incorrect data. Also, remember that tokens can be used in object type commands and notifications. Any changes to these could disrupt the system.

Changing information like the field prompt or description should not affect the behavior of existing package lines.

To edit an existing field:

1. From the Workbench shortcut bar, select **Deployment Mgmt > Object Types**.

   The Object Type Workbench opens.

2. Open an object type.

   The Object Type window opens.

3. From the **Fields** tab, select a field, and then click **Edit**.

   The Field window opens.

4. Modify the field as required, and then click **OK**.
5. Click **OK**.

Removing Fields

Removing a field from an object type does not change the historical information for existing package lines using the given object type. Any values for the deleted field remain in the package lines table in the column specified in the field definition.

To remove a field permanently from an object type:

1. From the Workbench shortcut bar, select **Deployment Mgmt > Object Types**.

   The Object Type Workbench opens.

2. Open an object type.

   The Object Type window opens.

3. From the **Fields** tab, select a field, and then click **Remove**.
4. Click **OK**.

Configuring Layouts for Object Types

You can change the look of an object type using the **Layout** tab of the Object Type window. Fields can be wide or narrow. Wide fields span two columns. Narrow fields span a single column. You can also move wide fields up and down. Narrow fields can move up and down, and side to side.

Changing Field Widths

To change the width of an object type field:

1. From the Workbench shortcut bar, select **Deployment Mgmt > Object Types**.

   The Object Type Workbench opens.

2. Open an object type.

   The Object Type window opens.

3. Click the **Layout** tab.

4. Select a field.
5. In the **Field Width** field, select **1** or **2**.

   The Layout editor does not let you make changes that conflict with another field in the layout. For fields of component type Text Area, you can determine the number of lines the text area displays by clicking the Text Area type field and changing the value in the Component Lines attribute. If the selected field is not of type Text Area, this attribute is blank and cannot be changed.
6. Click **OK**.

Moving Fields

To move an object type field:

1. From the Workbench shortcut bar, select **Deployment Mgmt > Object Types**.

   The Object Type Workbench opens.
2. Open an object type.

   The Object Type window opens.
3. Click the **Layout** tab.
4. Select a field.

   To select a range of fields, use the `Shift` key. You cannot use the `Ctrl` key to select nonadjacent field names.
5. Use the arrow pointers to reposition the fields in the layout builder.
6. To switch the positions of two fields:
   a. Select the first field, and then select the **Swap Mode** option. An **S** is displayed in the option section of the selected field.
   b. Double-click the second field.

      After the fields change positions, the **Swap Mode** option is cleared.
7. To open a window that displays a preview of your layout, click **Preview**.

   Note that:
   - If all the fields are one column wide, then all displayed columns automatically span the available section whenever a user views a package line of the given object type.

   - Rows that contain no fields are ignored. They are not displayed as blank lines.
   - Any fields that are not displayed do not affect the layout. They are considered the same as a blank field.

8. Click **OK**.

   The changes to the object type are saved.

Setting Object Names

When defining an object type, it is important to choose one field to represent the name of this object in a package line. This field is the object name. To designate a field as the object name, select that field's Parameter Column in the **Object Name Column** field.

For example, to designate the File Name field as the object name for a File Migration object type, select the File Name field's Parameter Column in the **Object Name Column** field.

In the package window, the object name for each package line is displayed in the **Object Name** column, on the **Status** tab.

The object name field drives additional functionality:

- If the object name field is a file chooser or an auto-complete list, multi-selection is automatically enabled on this field when users add a line to a package with this object type. If multiple values are selected, a new package line for each value will be created, allowing users to add multiple lines to a package simultaneously.
- All migrations are tracked in the database tables KENV_ENV_CONTENTS and KENV_ENV_CONTENTS_HIST. The value of a package line's object name field is stored in these tables (along with other relevant data) whenever a migration occurs.
- You can use the Object Type Workbench to query the Object Name.

Setting Object Revisions

You can create a field on the object type to represent the revision number of the object. This field is often a numeric text field. You can then configure the deployment process to consider the object revision number during package processing.

Configuring Commands for Object Types

Object types can have many commands and each command can have many command steps. A command can be viewed as a particular function for an object type. Copying a file can be one command and checking that file into version control can be another. To perform these functions, a series of events needs to take place, and these events are defined in the command steps.

An additional level of flexibility is introduced if some commands can only be run under certain circumstances. This is powered by the **Condition** field of the commands (see "Command Conditions" on page 135).

For more information about how to create commands and special commands, see the *Commands, Tokens, and Validations Guide and Reference*.

Adding Commands to Object Types

To add commands to object types:

1. From the Workbench shortcut bar, select **Deployment Mgmt > Object Types**.

   The Object Type Workbench opens.

2. Open an object type.

   The Object Type window opens.

3. Click the **Commands** tab.

4. Click **New Cmd.**

   The New Command window opens.

5. Complete the fields in the New Command window, as follows.

Field Name	**Description**
Command	A simple name for the command.
Condition	A condition that determines whether the steps for the command are run or not. For more information, see "Command Conditions" on page 135.

Field Name	Description
Description	A description of the command.
Timeout	The amount of time the command can run before its process is terminated. This mechanism is used to abort commands that are hanging or taking an abnormal amount of time to run.
Enabled?	Indicates whether the command is enabled for execution.

6. Click **OK**.

   The **Commands** tab lists the new command.

7. Click **OK**.

Editing Commands of Object Types

To edit a command on an object type:

1. From the Workbench shortcut bar, select **Deployment Mgmt > Object Types**.

   The Object Type Workbench opens.

2. Open an object type.

   The Object Type window opens.

3. Click the **Commands** tab.

4. Select a command to edit, and then click **Edit Cmd.**

   The Edit Command window opens.

5. Type the information specified in the following table.

Field Name	Description
Command	A simple name for the command.
Condition	A condition that determines whether the steps for the command are run or not. (See "Command Conditions" on page 135 for more information.)
Description	A description of the command.
Timeout	The amount of time the command will be allowed to run before its

Field Name	Description
	process is terminated. This mechanism is used to abort commands that are hanging or taking an abnormal amount of time.
Enabled?	Indicates whether the command is enabled for execution.

6. Click **OK**.

   The **Commands** tab lists the edited command.

7. Click **OK**.

Copying Commands in Object Types

To copy a command in an object type:

1. The Object Type Workbench opens.

   From the Workbench shortcut bar, select **Deployment Mgmt > Object Types**.

2. Open an object type.

   The Object Type window opens.

3. Click the **Commands** tab.

4. Select the command to copy, and then click **Copy Cmd**.

   The command is copied to another line in the **Commands** tab.

5. Click **OK**.

   The changes to the object type are saved.

Deleting Commands in Object Types

To delete a command in an object type:

1. From the Workbench shortcut bar, select **Deployment Mgmt > Object Types**.

   The Object Type Workbench opens.

2. Open an object type.

The Object Type window opens.

3. Click the **Commands** tab.
4. Select the command to delete, and then click **Remove**.
5. Click **OK**.

Command Conditions

Depending on the execution context, it might be necessary to run a different set of commands. This flexibility is achieved through the use of conditional commands. The **Condition** field for a command is used to define the situations when the associated command steps run.

Conditions are evaluated as boolean expressions. If the expression evaluates to true, the command is run. If false, the command is skipped and the next command is evaluated. If no condition is specified, the command is always run. The syntax of a condition is identical to the WHERE clause of an SQL statement, which allows enormous flexibility when evaluating scenarios. Some example conditions are detailed in "Table 5-2. Example conditions" below. Be sure to place single quotes around string literals or tokens that are to evaluate strings.

Table 5-2. Example conditions

Condition	Evaluates to
BLANK	Command is run in all situations.
`[P.P_VERSION_LABEL]' IS NOT NULL	Command is run if the parameter with the token P_VERSION_ LABEL in the package line is not null.
`[DEST_ENV.ENVIRONMENT_ NAME]' = `Archive'	Command is run when the destination Environment is named "Archive."
`[AS.SERVER_TYPE_CODE]' = `UNIX'	Command is run if the application server is installed on a UNIX machine.

The condition can include tokens. For more information concerning tokens, see the HPE.

Configuring Ownership for Object Types

To define ownership groups, add security groups to the **Ownership** tab. If no ownership groups are associated with the entity, the entity is treated as global and any user who can edit an object type can edit, copy, or delete this object type.

If a security group is disabled or loses the ability to edit object types, its members also can no longer edit an object type. For more information on access grants, see the *Security Model Guide and Reference*.

Adding Ownerships to Object Types

To add an ownership:

1. From the Workbench shortcut bar, select **Deployment Mgmt > Object Types**.

   The Object Type Workbench opens.

2. Open an object type.

   The Object Type window opens.

3. Click the **Ownership** tab.

4. Select the ownership option.

   - **All users with the Edit Object Type Access Grant**
   - **Only groups listed below that have the Edit Object Type Access Grant**

   Do the following:

   a. Click **Add**.

      The Add Security Group window opens.

   b. Select the security group, and then click **OK**.

      The **Security Group** field lists the selected security group.

   c. Click **OK**.

      The security group is added to the **Ownership** tab.

5. Click **OK**.

Deleting Ownerships from Object Types

To delete an ownership:

1. From the Workbench shortcut bar, select **Deployment Mgmt > Object Types**.

   The Object Type Workbench opens.

2. Open an object type.

   The Object Type window opens.

3. Click the **Ownership** tab.

4. Select an ownership.

   The **All users with the Edit Object Type Access Grant** option assigns ownership of the object type to users who can edit object types.

5. Click **Remove**.

6. Click **OK**.

Using Commands to Change Field Values

Commands can be used to control certain behavior of object type fields. At specific points (execution workflow steps) in the deployment process, you can run the commands stored in the object type. These commands can then manipulate the data inside an object type field. For example, you can construct a command to consider a number of parameters, and then default a field based on those parameters. This can be better than the default features for the Field window, which default, based on a single field located in the same object type.

The `ksc_store` special command can perform this function. For information on using this and other commands, see the *Commands, Tokens, and Validations Guide and Reference*.

You can use commands to control field values in the following situations:

- To store a value from an execution in a custom field
- To clear a field after you evaluate several parameters

Chapter 6: Configuring Releases and Distributions

- "Overview of Releases and Distributions" below
- "Overview of Configuring Releases " on page 148
- "Opening Release Workbench" on page 148
- "Creating Releases " on page 149
- "Adding Packages to Releases " on page 150
- "Adding Requests to Releases " on page 152
- "Verifying Releases " on page 153
- "Creating Distributions " on page 154

Overview of Releases and Distributions

A release is a group of packages and related requests that need to be deployed together. Release management provides an interface through which users can group, view and execute these packages. You can add packages to a release either by including a Ready for Release step in the package workflow or by the release manager through the Release window.

For example, a product update release is scheduled five months from now. To ensure a smooth product delivery, all changes are tracked to the original code using release management. As developers complete packages, those packages are included in a release and processed together. By grouping every required change in the release, you can quickly assess the state of the product delivery.

Workflow Scope

Each workflow has an associated workflow scope. The workflow scope determines which release management entities can be processed through that workflow. The workflow scope can be one of the following:

- Packages
- Requests
- Release Distributions

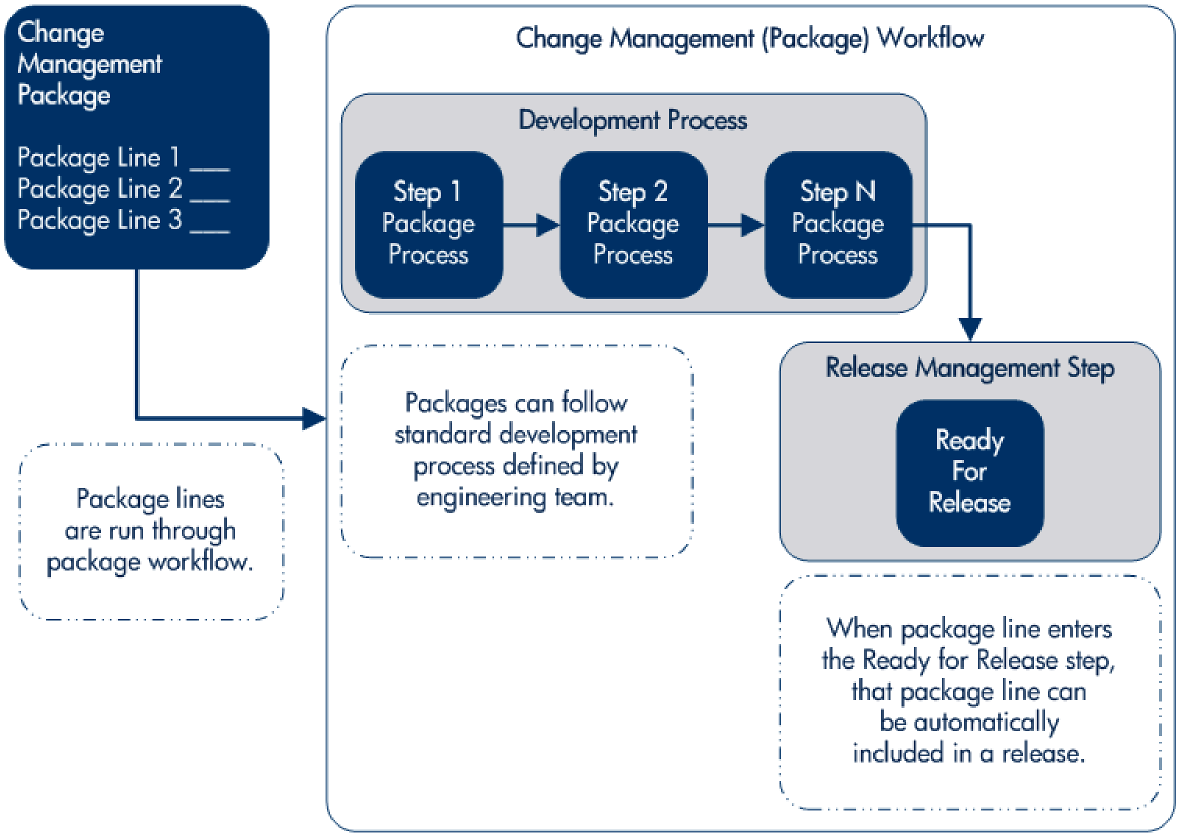
Release management uses workflows with packages and release distribution scopes. Workflow scope enforces certain workflow configuration restrictions. For example, release distribution workflows cannot include the wf_jump and wf_receive workflow events.

Release Management and Package Workflows

You can configure your standard package workflows to feed packages into a release. A Ready for Release workflow step can be included in the package workflow. When a package line enters the Ready for Release step, the developer (or other release management user responsible for that package) can select which release they would like to add the package to. The user selects the release and adds the package and its associated package lines to the release. When all the package lines are confirmed in the Ready for Release step, the package is ready to be used in the release.

"Figure 6-1. Ready for release step in workflow" on the next page shows the process by which developers can add packages to a release.

Figure 6-1. Ready for release step in workflow


Change Management Package
Package Line 1 ___
Package Line 2 ___
Package Line 3 ___
Package lines are run through package workflow.
Change Management (Package) Workflow
Development Process
Step 1 Package Process
Step 2 Package Process
Step N Package Process
Packages can follow standard development process defined by engineering team.
Release Management Step
Ready For Release
When package line enters the Ready for Release step, that package line can be automatically included in a release.

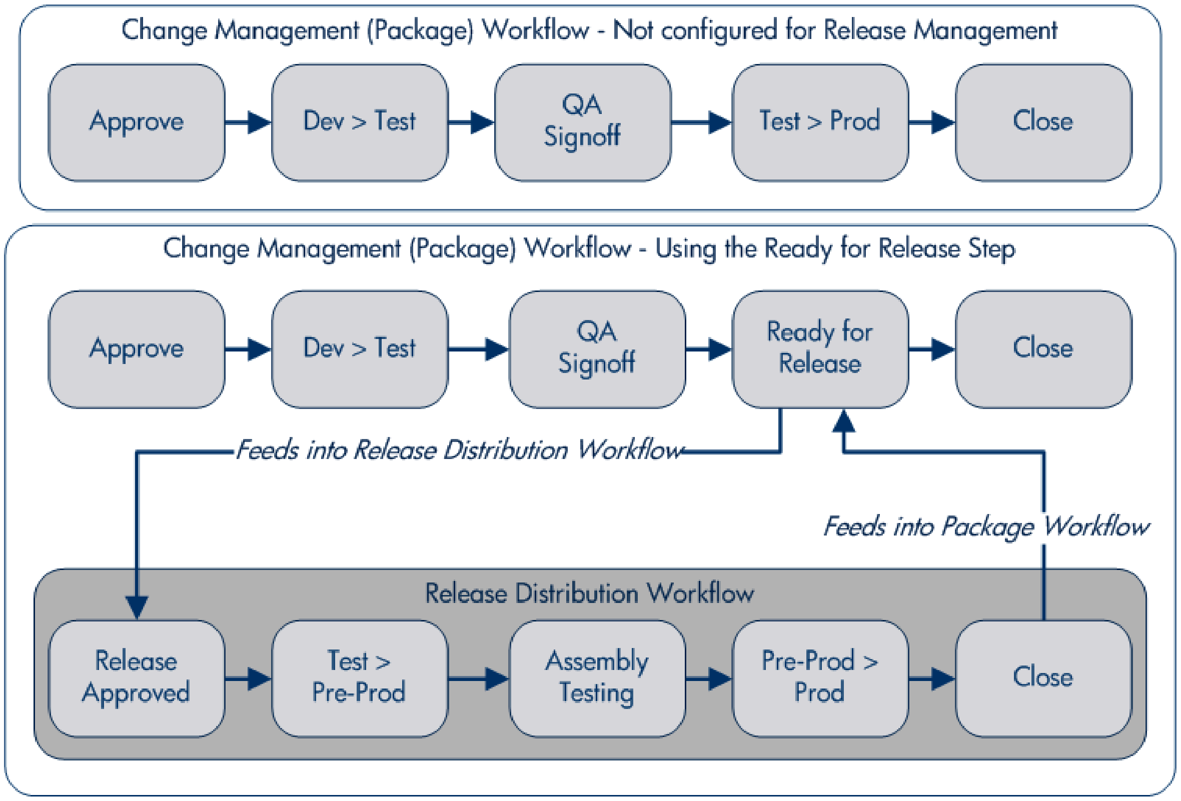
Release Distribution Workflows

Just as the inclusion of applicable packages and requests is integral to the release definition, so is the process by which the packages are processed in a release distribution. Distribution workflows are used to define the process by which the release's packages are properly tested, approved, and run against any required environments.

Release distribution workflows need to include package level subworkflows to perform key package level processing. All package line (object type) executions will occur in the subworkflow.

"Figure 6-2. Role of the distribution workflow" on the next page shows the relationships between packages and release workflows.

Figure 6-2. Role of the distribution workflow

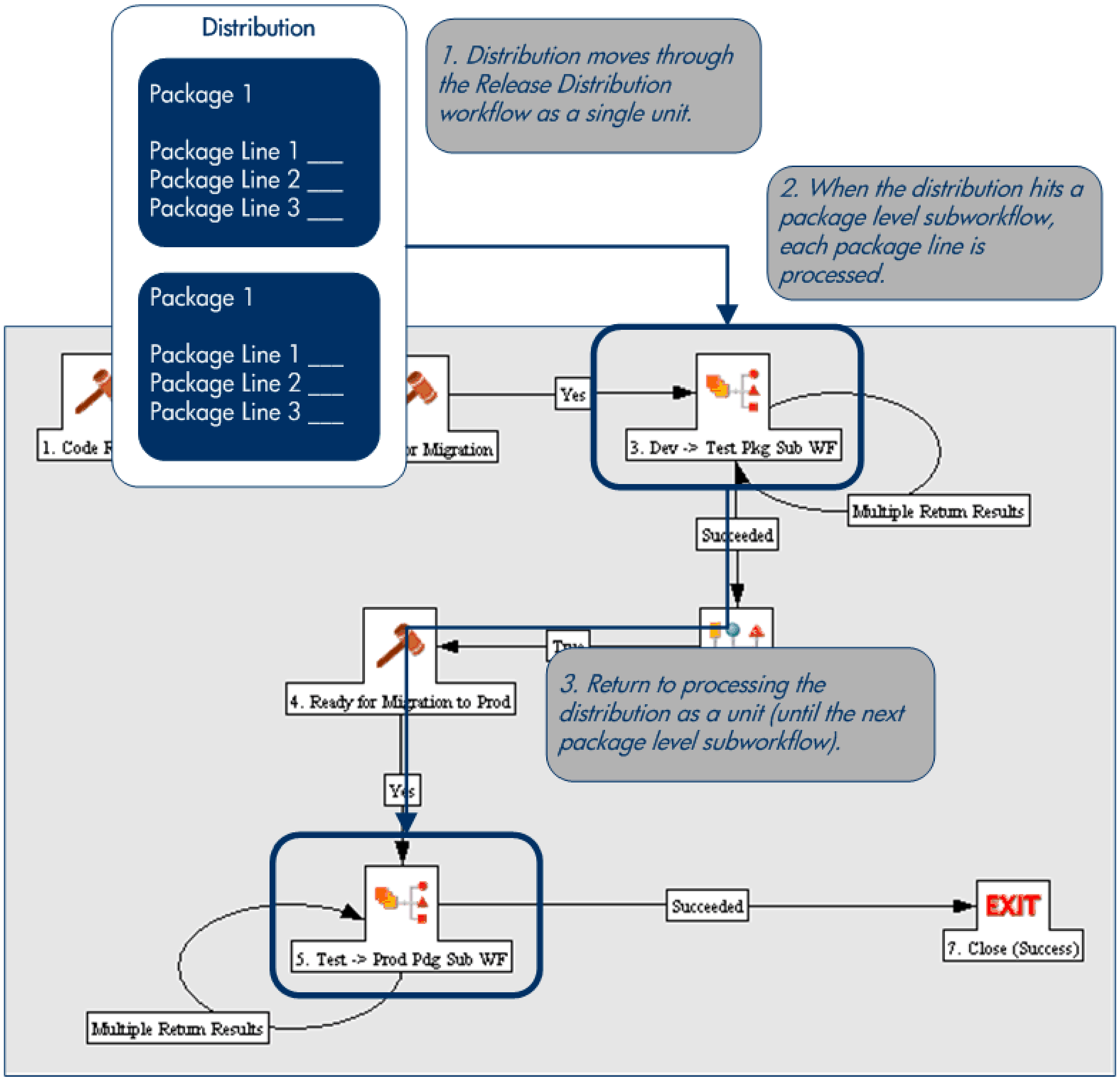

Change Management (Package) Workflow - Not configured for Release Management
Approve
Dev > Test
QA Signoff
Test > Prod
Close
Change Management (Package) Workflow - Using the Ready for Release Step
Approve
Dev > Test
QA Signoff
Ready for Release
Close
Feeds into Release Distribution Workflow
Feeds into Package Workflow
Release Distribution Workflow
Release Approved
Test > Pre-Prod
Assembly Testing
Pre-Prod > Prod
Close

The release distribution workflow provides a way in which the release manager can ensure that all files associated with the release deploy properly. As with any workflow, the distribution workflow can be configured to model your existing or best-practice release processes.

Package Level Subworkflows

Release distributions include package level subworkflows, which are used to perform key package level processing. Package level subworkflows are any package subworkflows that have the Use in Release Distributions flag set to **Yes.** All package line (object type) execution occurs in these subworkflows. Also, all package and package line tokens are resolved as these workflows are traversed.

Figure 6-3. Distribution workflow


Distribution
Package 1
Package Line 1 ___
Package Line 2 ___
Package Line 3 ___
Package 1
Package Line 1 ___
Package Line 2 ___
Package Line 3 ___
1. Distribution moves through the Release Distribution workflow as a single unit.
2. When the distribution hits a package level subworkflow, each package line is processed.
1. Code R
r Migration
Yes
3. Dev -> Test Pkg Sub WF
Multiple Return Results
Succeeded
True
4. Ready for Migration to Prod
3. Return to processing the distribution as a unit (until the next package level subworkflow).
Yes
5. Test -> Prod Pdg Sub WF
Succeeded
EXIT
7. Close (Success)
Multiple Return Results

Package level subworkflows are used within release distribution workflows. The release distribution workflow is typically used for release approvals and executing system commands (such as starting or stopping servers). The distribution is processed as a single unit, as it proceeds through the release distribution workflow. When the distribution hits a package-level subworkflow, each package line within the distribution is processed. The package subworkflows are used to process package lines and run object type commands.

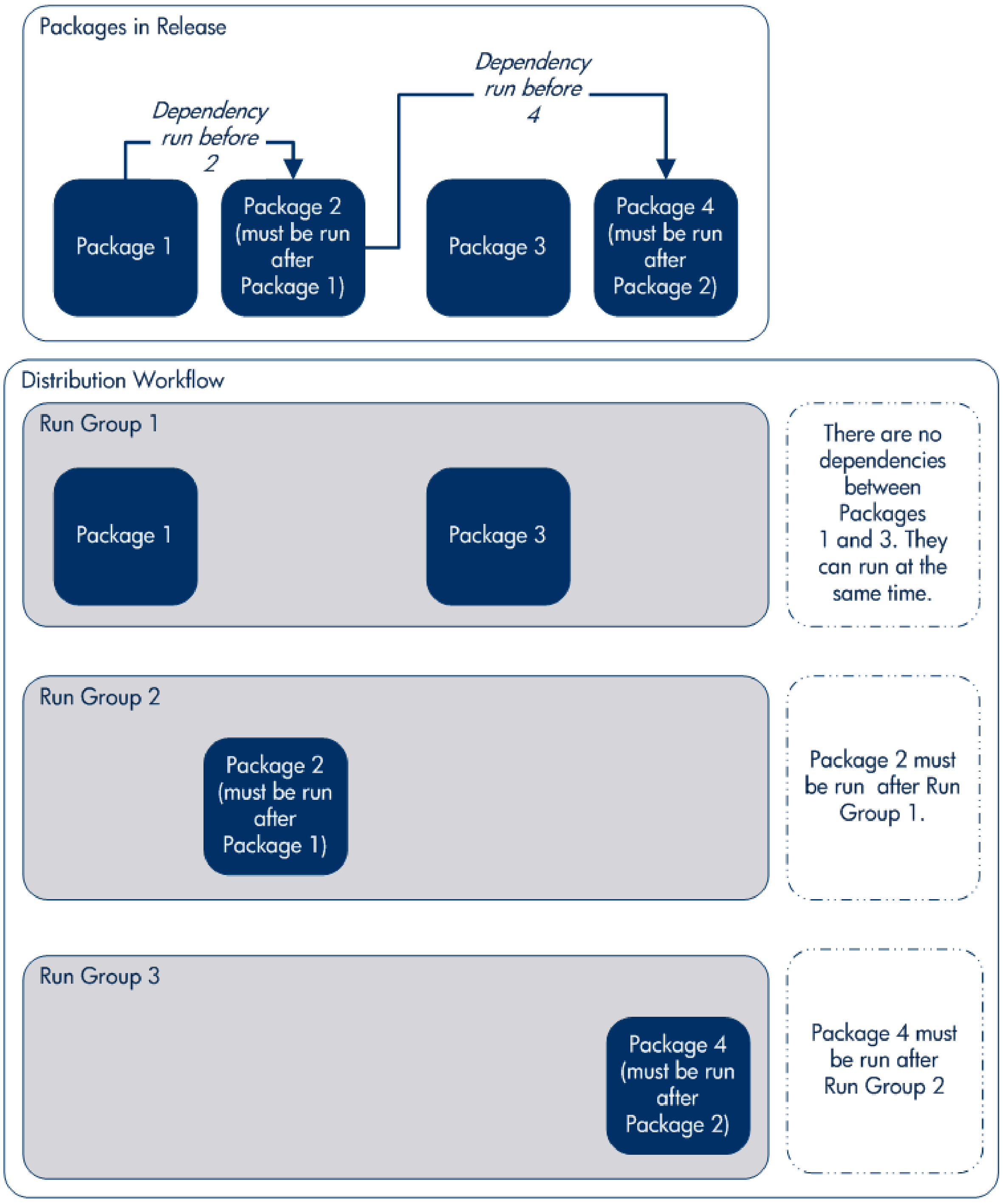
Dependencies and Run Groups

Within a release, the release manager can configure the order in which the packages are processed. The release manager can select certain packages to run before or after other packages in the release. The ordering of packages segregates them into run groups. When a distribution enters an execution step in a package-level subworkflow, all package lines in the first run group will be run before the package lines in the next run group can begin.

Run groups are automatically determined as you set dependencies between packages. Run groups present an efficient way to process packages, which can be run in parallel without having to serially wait for unrelated dependencies. "Figure 6-4. Dependencies and run groups" on the next page shows how package dependencies result in different run groups.

Run groups are automatically determined when a release distribution is created, based on package dependencies specified in the release.

Figure 6-4. Dependencies and run groups


Packages in Release
Dependency run before 2
Dependency run before 4
Package 1
Package 2 (must be run after Package 1)
Package 3
Package 4 (must be run after Package 2)
Distribution Workflow
Run Group 1
Package 1
Package 3
There are no dependencies between Packages 1 and 3. They can run at the same time.
Run Group 2
Package 2 (must be run after Package 1)
Package 2 must be run after Run Group 1.
Run Group 3
Package 4 (must be run after Package 2)
Package 4 must be run after Run Group 2

Opening Releases

When a release manager first creates a release, only release management users with permission to edit the release window can add packages. The release manager can enable other users to add packages to the release by clicking **Open Release** in the release window. By creating an open release, developers processing a Ready for Release workflow step will have the option of adding the package to that release. If a release is not opened, it is not displayed in the list used for that Ready for Release step.

Submitting Releases

When a release manager submits a release, the release enters a code freeze state. In this state, packages cannot be added or removed from the release by anyone other than the release manager. When the release is submitted, a distribution is automatically created. You can then process the distribution, or cancel it and create a new one later.

Overview of Using Release Management - Process

Release management introduces repeatable, reliable processes surrounding software and application releases. Release management provides an interface for grouping and processing the packages and requests associated with a specific release.

Release Management Preconfiguration

Planning for an application or software release should begin immediately upon recognition that a release is pending. Before the release manager creates a release, it is often necessary to preconfigure the following in release management:

1. Modify package workflows to include the Ready for Release workflow step.
2. Create all required distribution workflows (including all package-level subworkflows).

By adding the Ready for Release workflow step to workflows, you provide developers with the ability to add a required package to a release at the appropriate time. Development package workflows typically address necessary approval and execution steps directly related to that package. The Ready for Release workflow step indicates that the developer has signed off on the package and the package is ready to be integrated and shipped with other packages related to the release.

The release manager should also create the distribution workflows. This includes defining any subworkflows (package level or distribution level) that will be used in the release distribution workflow. These workflows define the process by which the release's packages are properly tested, approved, and run against any required environments. The release distribution workflow and associated subworkflows ensure that all files associated with the release are properly deployed. As with any workflow, the distribution workflow can be configured to model your existing or best-practice release processes.

Creating Releases

To create a release in release management:

1. Create a new release in the Release window (see "Creating Releases " on page 149).
2. Open the release by clicking **Open Release** (see "Opening Release Workbench" on page 148).

   This allows developers working in the Deployment Management Package window to add packages to the release through the Ready for Release workflow step.
3. Add packages and requests to the release (see "Adding Packages to Releases " on page 150 and "Adding Requests to Releases " on page 152).

   Packages can be added through the Ready for Release workflow step in the Packages window or directly by the release manager through the Release window.
4. Click **Dependencies** in the **Package** tab to set package dependencies.
5. Configure dependencies between packages in the release.
6. Verify the release.

Processing Releases

When the applicable data is collected in the release (packages, requests and dependencies) and the applicable workflows have been created, the release manager can then process the release.

To process a release the release manager will:

1. Submit the release.
   a. Create a distribution. This consists of selecting a workflow for the distribution and disabling any package lines that should not run with that distribution.
   b. Submit the distribution.
2. Send feedback to packages. Send feedback to the packages at any time from the Distribution window. Select the value from the feedback drop-down list and click **Feedback**. This value is sent back to the package line in the Ready for Release workflow step and is used to transition out of that step.
3. Close the release only if there is no need to create additional distributions for use at a later date.
   - When the release is closed, any in-progress distributions are cancelled.
   - If a distribution is not submitted, it is not cancelled when the release is closed. After the release is closed the distribution cannot be edited.

Distributions

A distribution is a deployment of a release. In a distribution, the release manager specifies which workflow will control the release process and which of the release's packages will be included.

For example, a software company has a product update release scheduled five months from now. As a part of their release process, they need to update their testing, production, and training instances of the product. The processes required for delivering the product to these environments differs in the following ways:

- Not all the packages in the release need to be applied to each instance. For example, the training instance requires custom code to establish additional product security, which is not required in the production instance.
- There is a different review process for each instance. For example, the testing instance does not require the department head sign-off for each iteration of the release.

The software company creates a distribution for each of these release instances. For each distribution, the release manager defines which:

- Packages are included in the specific release instance.
- Workflow the release follows.

Overview of Configuring Releases

Setting up a successful software or application release requires a comprehensive view of the release process. Release management provides the tools for capturing the entire release process. For an overview of the items and processes involved in creating a release, see "Overview of Using Release Management - Process " on page 145.

Establishing a controlled release starts with creating a new release in the Release Workbench.

The following lists the sections found in the Release window:

- **General information**: Includes basic information concerning the environment, such as the environment name and description. See "Creating Releases " on the next page.
- **Packages**: Allows packages to be added to the release. See "Adding Packages to Releases " on page 150.
- **Requests**: Allows requests to be added to the release. See "Adding Requests to Releases " on page 152.
- **Distributions**: Creates distributions for releases. See "Creating Distributions " on page 154.
- **Notes**: Lets you add notes to a release.
- **References**: Tracks information regarding the release.

Opening Release Workbench

To open the Release Workbench:

1. Log on to PPM.
2. From the menu bar, select **Open > Administration > Open Workbench**.

   The PPM Workbench opens.
3. From the shortcut menu, select **Deployment Mgmt > Releases**.

   The Release Workbench opens.

Creating Releases

To configure general information and create a release:

1. From the Workbench shortcut bar, select **Deployment Mgmt > Releases**.

   The Release Workbench opens.

2. Click **New Release**.

   The Release window opens.

3. Complete the fields as specified in the following table.

Field Name	Description
Release Name	The name of the release.
Release Status	The current status of the Release. ◦ **New** ◦ **Open** ◦ **Code Freeze** ◦ **Closed**
Release Manager	The name of the release management user who has control over the particular release. Only release management users who can manage releases are listed.
Release Team	The users who have access to the particular release. This is a validated list of security groups and is used for informational purposes only.
Release Group	A generic grouping of releases which allows the release manager to group releases into logical categories such as customization.
Description	The description of the release.

4. Add packages to the release.

   For information on how to add packages, see "Adding Packages to Releases " on the next page.

5. Add requests to the release.

   For information on how to add requests, see "Adding Requests to Releases " on page 152.

6. To allow other release management users to add packages and requests to the release, click **Open Release**.
7. Click **OK**.

Adding Packages to Releases

Adding Packages Through Release Window

When defining a release in the Release window, the release manager can decide to manually add packages to the release.

To manually add a package to a release:

1. From the Workbench shortcut bar, select **Deployment Mgmt > Releases**.

   The Release Workbench opens.
2. Open a release.

   The Release window opens to the **Packages** tab.
3. Click **Add**.

   The Package Selection window opens.
4. Specify your search criteria, and then click **List**.

   The **Query Results** field displays the packages that match the search criteria.
5. In the **Query Results** field, select the packages to add to the release.
6. Click **Add**.

   If the packages reference any other packages or requests, you are prompted to indicate whether to include or exclude them.
7. Click **Close**.

   The Release window now lists the packages you added.
8. Click **Save**.

Adding Packages Through Package Window

To add packages to a release from the Package window, you reference the release on the package **References** tab.

Adding Packages by Ready for Release Workflow Step

Release management provides a Ready for Release workflow step source, which can add significant value to your release process. When a package line reaches the Ready for Release workflow step and is run, the status of the package line changes to Confirmed. As soon as all the lines in a package reach this status, the entire package status changes to Ready for Release. The release distribution can then feed back to each of its associated packages, so that each package can progress to the next workflow step.

To reject the Ready for Release workflow step, select **Bypass Execution** or **Override Status.** This stops package release, but allows the package to continue through its own workflow.

To act on a Ready for Release workflow step:

1. From the Workbench shortcut bar, select **Deployment Mgmt > Packages**.

   The Package Workbench opens.

2. Open a package.

   The Package window opens.

3. Click the **Status** tab.

4. Select the workflow step to act on.

   Workflows that are available for action are displayed in bold text. After you select an available workflow step, the **Action** button label changes to the workflow step name.

5. Click **Ready for Release**.

   The Package Action window opens.

6. In the **Results** field, select the workflow step result.

   You can also type any relevant notes in this window.

7. Use the **Add to Release** field to select the release to associate with the package.

   This field may be required, depending on the workflow step configuration.

8. Click **OK**.

9. In the Package window, click **OK**.

   The package is now ready for release.

Adding Packages from Requests

When a request that is included in a release spawns a package, that package is included in the release automatically. This becomes a powerful method for including packages in a release.

For example, a development manager can include a request to fix a software bug in the release. You can configure that request workflow to automatically create a package to migrate changes into production. That package is then included in the release.

Adding Requests to Releases

Add requests to a release to track information for that release. For example, the release manager may want to track the software bugs or enhancements (captured using a request) that were implemented during the release time frame.

Adding Requests Through Release Window

To add a request to a release from the Release window:

1. From the Workbench shortcut bar, select **Deployment Mgmt > Releases**.

   The Release Workbench opens.

2. Open a release.

   The Release window opens.

3. Click the **Requests** tab.
4. Click **Add**.

   The Request Selection window opens.
5. Type search criteria, and then click **List**.
6. In the **Query Results** field, select the request to add to the release.
7. Click **Add**.

   If there are any referenced entities, you are prompted to indicate whether to include or exclude them.
8. Click **Close**.
9. In the Release window, click **Save**.

Adding Requests Through Requests Window

Users can only add requests to an open release. After the release reaches the code freeze or closed state, requests can only be added through the Release window by the release manager.

To add a request to a release:

1. Log on to PPM.
2. From the menu bar, select **Dashboard > My Requests**.
3. Select the References section.
4. In the **New Reference** field, select **Release**.
5. Click **Add**.

   The Reference Release window opens.
6. Use the **Release** field to select a release.

   The **Release** field displays only the releases that the release manager has opened.

Verifying Releases

After you assemble a release (packages, requests, and dependencies), verify that the release is properly configured.

To verify your release:

1. From the Workbench shortcut bar, select **Deployment Mgmt > Releases**.

   The Release Workbench opens.

2. Open a release.

   The Release window opens to the **Packages** tab.

3. Click **Verify**.

   The Verify Release Properties window opens.

4. In the list at the top of the window, select one of the following options:

   - **Check for Packages without Lines**
   - **Check for Packages without Associated Requests**
   - **Check for Packages without Associated Packages**
   - **Check for Dependencies not in the Release**
   - **Check for Circular Dependencies**

5. Click **Submit**.

   Any errors are reported in the window.

6. After you examine the results, click **OK**.

Creating Distributions

A distribution is a deployment of a release. In a distribution, the release manager specifies the workflow to control the release process and specifies which release packages to include.

To create a distribution:

1. From the Workbench shortcut bar, select **Deployment Mgmt > Releases**.

   The Release Workbench opens.

2. Open a release.

   The Release window opens.

3. Click **Open Release**.

   This enables the **Distributions** tab.

4. Click the **Distributions** tab.
5. Click **New**.

   The Distribution window opens.
6. Type the new distribution name and description.
7. Select the workflow for this distribution to follow.
8. Select any packages that you want to disable, and then click **Enable/Disable**.

   The names of disabled packages are displayed in italic text.
9. Click **Submit Release**.

   The distribution runs through the workflow specified. The release begins to run through the assigned workflow.

Enabling or Disabling Package Lines in a Distribution

To disable a package line in a distribution:

1. From the Workbench shortcut bar, select **Deployment Mgmt > Releases**.

   The Release Workbench opens.
2. Open a release.

   The Release window opens.
3. Click the **Distributions** tab.
4. Click the **Package Status** tab.
5. Locate the package line that you want to disable.
6. To display all the packages, click **Run Groups**.

   Note that you may have to change your filter to see the package you want.
7. Select the package line to disable.

   You can disable an entire package. Disabling a package line in an active run group (within a package-level subworkflow) cancels the package line.

8. Click **Disable.**

   The name of the disabled package line is displayed in italic text.

If you disable a package line in an active run group, you cannot reenable the package line until the run group completes. If the disabled package line was not in an active run group, you can reenable it immediately.

Running Distributions through a Workflow

The last step involved in creating a release is running the release through a Deployment Management workflow. This involves running any decisions, commands, token evaluations, or other tasks for the entire distribution.

Processing the distribution requires that you process steps on both the **Distribution Status** and **Package Status** tabs.

Processing Distribution Steps

Active workflow steps appear in bold text. Select the active line and click **Action** to process that workflow step. On the **Distribution Status** tab, you can expand and act on all distribution steps (including distribution-level subworkflow steps). To process package lines, you must use the **Package Status** tab.

Processing Package Lines

Package lines can be processed individually or in groups. Package lines that are available for your action appear in bold text. Select an active package line and click **Action** to process that individual workflow step.

Release management provides a convenient interface for processing groups of package lines (in the same workflow step) simultaneously. This is done by viewing and selecting the package statuses.

To select all package lines within a workflow step of a particular status:

1. From the Workbench shortcut bar, select **Deployment Mgmt > Releases**.

   The Release Workbench opens.

2. Open a release.
3. Select the **Distributions** tab.
4. Click the **Package Status** tab.
5. Expand all run groups and workflows.
6. To display a status summary of all the package lines in each workflow step, in the Description column, click the plus icon (+).
7. Select the summary.

   Again, items that are available for your action appear in bold text. When you select the summary, all package lines in that state are automatically selected. You can deselect individual items using `Ctrl + click.`
8. Click **Action** to process all the selected package lines.
9. Proceed to the next workflow step in the package process or distribution process (depending on your preconfigured process).

   To view updates in the **Distribution Status** tab, click in the tab, and then click **Refresh**.

Completing Distributions

When the distribution completes and the workflow closes, a value can be returned to the Ready for Release workflow steps. Those packages can then continue to process, based on that validation. The value is sent by clicking **Feedback** in the Distribution window.

Chapter 7: Configuring Environments

- "Overview of Environments " below
- "Overview of Configuring Environments " on page 165
- "Overview of Environment Groups " on page 179
- "Overview of Environment Refresh " on page 185

Overview of Environments

When migrating objects, Deployment Management logs onto remote computers the same way any other user would (using FTP, SSH, or Telnet). Deployment Management can log on using any existing username and password, but HPE recommends that you generate a new user on each computer that Deployment Management is to access. This helps to clarify the setup and relieve some administrative burden.

Make sure that you have full access to the <*PPM_Home*> directory on the PPM Server and the correct read and write permissions on other required directories. On Windows® servers, the Administrators group must have read access to the PPM Server home directory. Any Windows server that Deployment Management accesses should be configured as described in the *Installation and Administration Guide*.

Environment Connection Protocols

The communication protocol used to connect to the server or client must be specified in the environment. This protocol is used by commands to connect to source and destination environments in the deployment system. Work with the application administrator to determine which connection protocols are supported at your site for the machines housing the deployment environments. The following connection protocols are supported:

- Telnet
- SSH

- SSH2 (Legacy)
- SSH2

> **Note:** SSH2 is a new client introduced in PPM Center version 9.30 to support FIPS 140.2 compliant encryption algorithms. To differentiate it from the old SSH2 introduced in earlier versions, the old SSH2 is renamed SSH2 (Legacy) in version 9.30.
>
> For details about using the SSH2 client, see "Configuring to Use the SSH2 Client" below.

Configuring to Use the SSH2 Client

To use the SSH2 client, you should configure the following server configuration parameters from the Administration Console.

Parameter Name	Description, Usage	Default and Valid Values
SSH2_JSCH_DISABLE_STRICT_HOST_KEY_CHECKING	When set to `true`, the client will connect to the remote host even if its key is not in the list of trusted hosts (`known_hosts` file). **Caution:** This parameter should not be set to `true` on a production environment.	Default: `false` Valid values: `true`, `false`
SSH2_JSCH_KNOWN_HOSTS_FILE_PATH	When a value is defined in this parameter (valid file path), PPM Center will use it as `known_hosts` file to validate keys of trusted hosts it connects to. The remote servers you will connect to should be included in the `known_hosts` file (using OpenSSH format), otherwise the client cannot connect to them (unless the `SSH2_JSCH_DISABLE_STRICT_HOST_KEY_CHECKING` parameter is set to `true`, in which case PPM Center does not search for a `known_hosts` file). If this parameter is left empty, PPM Center first checks if there is a `<PPM_HOME>/known_hosts` file to use. If no, PPM Center then checks known standard locations for `known_hosts` file: • `.../.ssh/known_hosts` and `.../etc/ssh/ssh_known_hosts` under UNIX	Default: N/A

	• `%USERPROFILE%\ssh\known_hosts` and `%USERPROFILE%\.ssh\known_hosts` under Windows **Note:** When a Linux user connects to a remote server using the `ssh` command on the command line and then accepts the host key when prompted, this remote machine key will be automatically added to the trusted hosts list in `.../.ssh/known_hosts`.	

Enabling Logging for SSH2 Library

If the SSH2 connection fails with no useful information in the PPM logs, you can route the logs of the JSch library to a dedicated file and set its logging level to a more verbose setting by adding the following text in your `logging.conf` file. You can change the red parts as you demand.

```
# SSH2/JSch logging.
log4j.logger.com.kintana.core.net.ssh2.jsch=DEBUG, JSCH_LOG

# JSCH_LOG has a dedicated log file to easily pinpoint any error during SSH2
connection.
log4j.appender.JSCH_LOG=org.apache.log4j.RollingFileAppender
log4j.appender.JSCH_LOG.File=${jboss.server.home.dir}/log/ssh2_jsch_log.txt
log4j.appender.JSCH_LOG.Append=true
log4j.appender.JSCH_LOG.MaxFileSize=250KB
log4j.appender.JSCH_LOG.MaxBackupIndex=20
log4j.appender.JSCH_LOG.layout=org.apache.log4j.PatternLayout
log4j.appender.JSCH_LOG.layout.ConversionPattern=%x:%t:%c:%d{yyyy/MM/dd-
HH:mm:ss.SSS z}: %m%n
```

Note: Make sure that the value of the parameter `com.kintana.core.logging.SYSTEM_THRESHOLD` in the `logging.conf` file is set according to the logging level chosen for JSch logs.

Adding Support for Ciphers of Unlimited Strength

If you find that the SSH2 client cannot use ciphers of unlimited strength such as AES-256-CTR, it is most probably because you have not installed the Java Cryptography Unlimited Strength Jurisdiction Policy Files on your JVM-running PPM Center.

The default Java installation is provided with limited cryptography capabilities (for example, keys over 128 bit cannot be used), as certain countries have regulations in place regarding authorized level of cryptography that can be freely used. In order to remove this limitation, you need to:

1. Download the new policy files for Java 7 from the following address:
   http://www.oracle.com/technetwork/java/javase/downloads/jce-7-download-432124.html

2. Install the policy in the `<java.home>/lib/security` directory.

Once this is done, all algorithms should become available.

> **Note:** Make sure you are installing the files into the JVM you are running PPM Center with.

SSH2 Client FAQ

This section lists the questions you may have regarding the SSH2 client. Each question is followed by an answer.

1. **Q**: What is the meaning of the following error when I use PPM Legacy SSH2 client:
   `ERROR: java.io.IOException: Unsupported encoding algorithms requested: aes256-ctr,aes192-ctr,aes128-ctr`?

   A: The PPM Legacy SSH2 client only supports Triple DES cipher as an encoding algorithm. If the server you try to connect to does NOT support it, you will encounter this error, and all the algorithms the server supports will be listed here.

2. **Q**: When trying to connect to a remote server with the PPM SSH2 client, why do I have the following error:
   `java.lang.RuntimeException: com.jcraft.jsch.JSchException: <host_name>. RSA key fingerprint is xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx:xx`?

   A: If you encounter this error message when trying to connect to a remote host, it means that PPM Center cannot find the remote host in the list of trusted hosts (also known as `known_hosts` file). Refer to the descriptions of parameters `SSH2_JSCH_DISABLE_STRICT_HOST_KEY_CHECKING` and `SSH2_JSCH_KNOWN_HOSTS_FILE_PATH` in the parameter table to address this problem.

3. **Q**: I can use aes128-ctr, but I cannot use aes256-ctr or aes192-ctr. Why?

   A: You need to enable support for ciphers of unlimited strength in your Java Virtual Machine running PPM Center. See the section "Adding Support for Ciphers of Unlimited Strength" on the previous page for details.

Environment Transfer Protocols

The transfer protocol that will be used to transfer files to the server or client must be specified in the environment.

Deployment Management supports the following transfer protocols:

- FTP
- FTP (active)
- FTP (passive)
- Secure Copy
- Secure Copy 2 (Legacy)
- Secure Copy 2

> **Note:** Secure Copy 2 is a new transfer protocol added in version 9.30. To differentiate it from the old Security Copy 2 introduced in earlier versions, the old Security Copy 2 is renamed Secure Copy 2 (Legacy).

Transfer Protocol Configuration Notes

Choose the transfer protocol best suited to the business and technology needs. Consider factors related to security and performance when selecting the transfer protocol. Work with the application administrator to determine which connection protocols are supported for the machines housing the deployment environments. The following list provides some suggestions for when to use the above protocols.

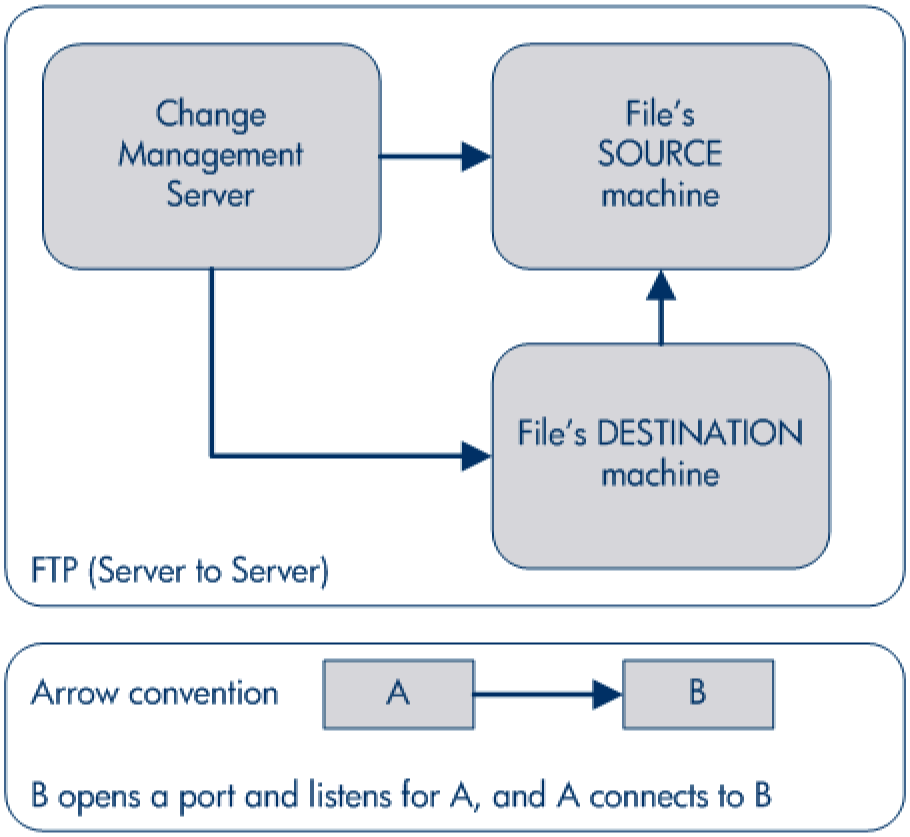
No additional product configuration is required to enable one FTP mode over another. Application administrators need to consider their FTP server configuration (particularly as they relate to security and firewall settings) when selecting an FTP protocol for transferring data.

Selecting FTP Protocol

The following capabilities must be enabled on the source and destination machines for the following FTP protocol selection to function properly.

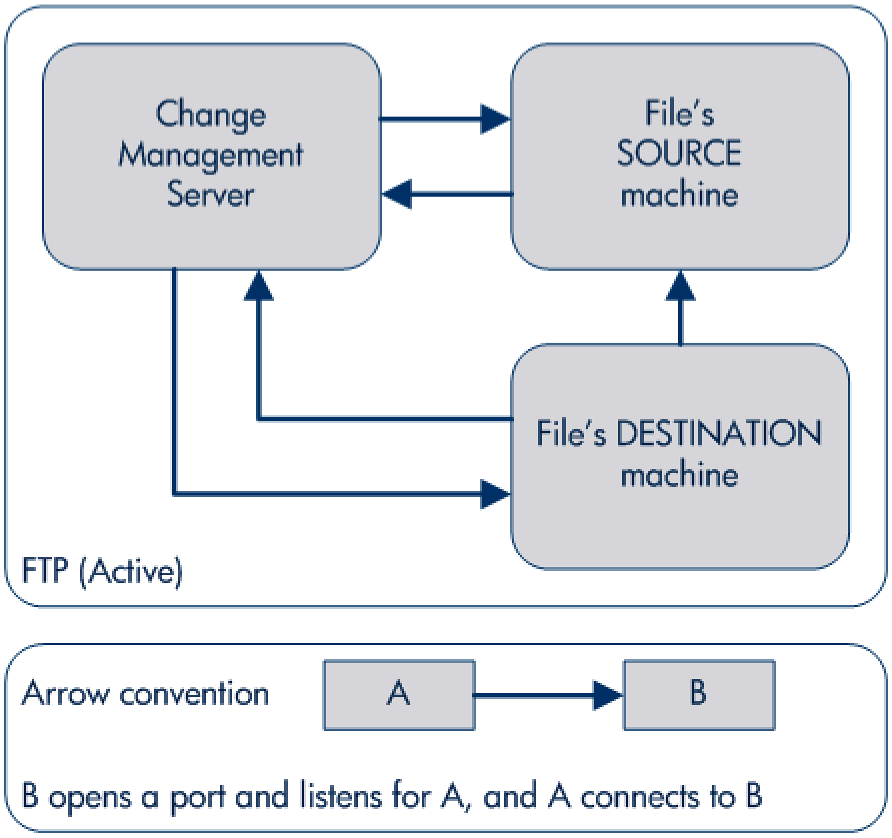
- FTP (server to server). See "Figure 7-1. FTP (server to server)" below.
  - Either the source or the destination environment needs to allow outgoing connections to a third party
  - FTP PORT command must be enabled on one of the environments
  - FTP PASV command must be enabled on the other environment

Figure 7-1. FTP (server to server)



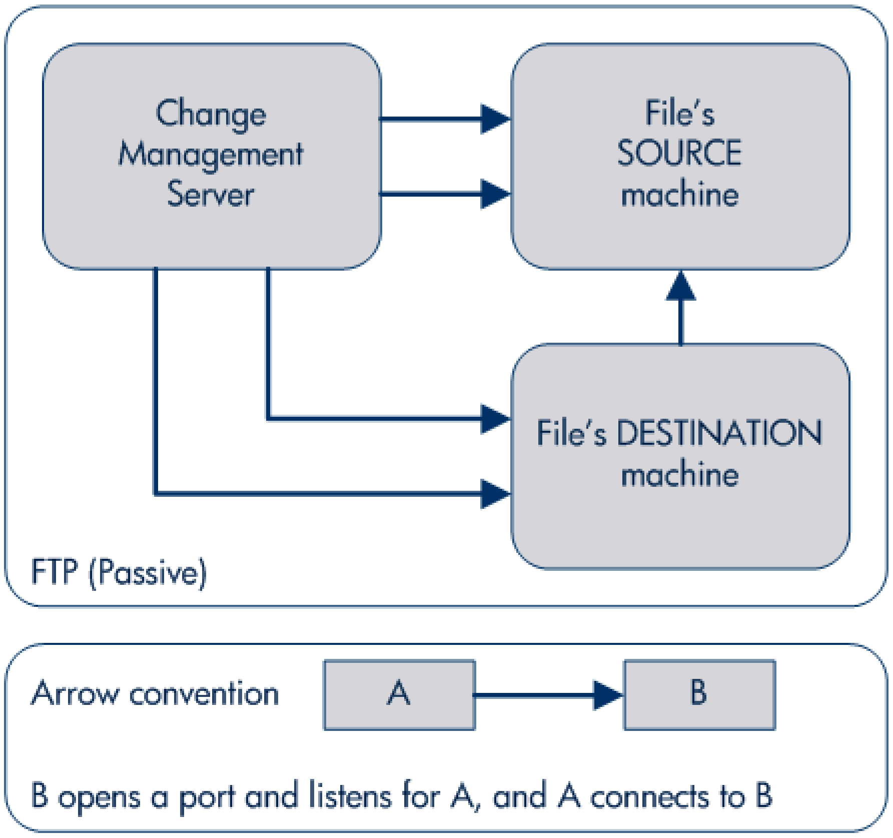
- FTP (active).
  - PORT command must be enabled on both the source and destination environments (allows outgoing back to the requestor)

Figure 7-2. FTP (active)


Change Management Server
File's SOURCE machine
File's DESTINATION machine
FTP (Active)
Arrow convention
A
B
B opens a port and listens for A, and A connects to B

- FTP (passive).
  - PASV must be enabled on both the source and destination environments. In this configuration, the HPE server sends a command to the source or destination instructing that environment to open a port. The HPE server then connects to that port.

Figure 7-3. FTP (passive)


Change Management Server
File's SOURCE machine
File's DESTINATION machine
FTP (Passive)
Arrow convention
A
B
B opens a port and listens for A, and A connects to B

Overview of Configuring Environments

Environments are configured in the Environments window. Some of the information provided on the Environments window can be gathered from the applicable worksheet in "Worksheets" on page 215.

The following lists the sections found in the Environments window:

- **General information**: Includes basic information concerning the environment, such as the environment name and description. See "Configuring General Information for Environments " on the next page.
- **Host**: Defines basic information about the client, server, and database for the environment. The fields for the client and server sections are identical. See "Creating Environments " on page 167.
- **Applications**: Every Deployment Management environment can contain its own set of applications. See "Using Application Code Environments " on page 171.

- **Extension Data**: Deployment Management Extensions require specially-configured environments. If Deployment Management Extension products are not installed, the **Extension Data** tab is disabled.
- **Ownership**: Configure who can edit the environment. See "Setting Ownership and Participants for Environments " on page 174.
- **User Access**: Configures participants of the environment. Participants can then be given specific access rights to the environment. See "Adding Participants to Environments" on page 175.
- **User Data**: Product entities, such as packages, workflows, requests, and projects, include a set of standard fields that provide information about those entities. While these fields are normally sufficient for day-to-day processing, user data fields provide the ability to capture additional information specific to each organization. User data is defined under the **User Data** tab. If there are no user data fields, the **User Data** tab is disabled.

Opening Environments Workbench

To open the Environments Workbench:

1. Log on to PPM.
2. From the menu bar, select **Open > Administration > Open Workbench**.

   The PPM Workbench opens.
3. From the shortcut bar, select **Environments > Environments**.

   The Environments Workbench opens.

Configuring General Information for Environments

To configure the general information of an environment:

1. From the Workbench shortcut bar, select **Environments > Environments**.

   The Environments Workbench opens.
2. Open an environment.

   The Environment window opens.

3. Complete the fields as specified in the following table.

Field Name	**Description**
Environment Name	The name of the environment.
Location	The location of the environment. For example, In the server room.
Description	A brief description of how the environment is being used.
Enabled	To make the environment available to the system, select **Yes**.

4. Click **OK**.

Creating Environments

To define a new environment:

1. From the Workbench shortcut bar, select **Environments > Environments.**

   The Environments Workbench opens.

2. Open an environment.

   The Environment window opens to the **Host** tab. The **Host** tab has the following three sections:

   - **Server**
   - **Client**
   - **Database**

3. Complete the fields in the **Server** section as specified in the following table.

Field Name	Description
Name	The DNS name or IP address of the computer. **Note:** If you select **SSH**, **SSH2 (Legacy)**, or **SSH2** in the **Connection Protocol** field, you can specify a port number other than the default port number of `22`. To do so, specify the **Name** field value in either of the following format: ◦ <*hostname*>:<*port_number*> ◦ <*IP_address*>:<*port_number*> Do not contain colons in the <*hostname*>, otherwise PPM Center cannot recognize the port number you specified. If you use IPv6 address, use the following format: [<*IPv6_Address*>]:<*port_number*>
Type	A list of supported server/client operating systems. Set to the operating system for the computer defined in the **Name** field.
Username	The username that Deployment Management uses to log on to the server/client to transfer files or execute commands. If that user account was generated on the local machine, this is most likely "HPE PPM."
Password	The password for the given username. The password is hidden. You can change it by clicking the button to the right of the field.
NT Domain	The domain name to use if this is a Windows Server.
Base Path	The path for applications on this computer. In many instances, this is the home directory of the defined username. When Deployment Management transfers a file or runs a command on this server, it logs in and changes directories to this base path before proceeding. Use forward slash characters (/) as directory separators, even for Windows systems.
Connection Protocol	The protocol to be used for server or client connections. SSH is a secure connections protocol, whereas Telnet is not. In order to use SSH as your connection protocol, you must first set up SSH on the destination machine. **Note:** If you select **SSH**, **SSH2 (Legacy)**, or **SSH2**, you can specify a port number other than the default port number of `22`.

Field Name	Description
	For details, see the note for the Name field above.
Transfer Protocol	The protocol to be used for moving files between various clients and servers. SCP is a secure transfer protocol, whereas FTP is not. In order to use SCP as your transfer protocol, you must first set up SCP on the destination machine.
Stream Encoding	Character encoding scheme used by the server.
Enable Server	Make the server information available to the system. Select the checkbox to make the server information available to the system.

4. Complete the fields in the **Client** section as specified in the following table.

Field Name	Description
Name	The DNS name or IP address of the computer.
Type	A list of supported server/client operating systems. Should be set to the operating system for the computer defined in the **Name** field.
Username	The username that Deployment Management uses to log onto the server/client in order to transfer files or run commands. This will usually be "HPE PPM" if that user account has been generated on this computer.
Password	The password for the given username. The password is hidden, and can be changed by clicking the button to the right of the field.
NT Domain	The domain name to use if this is a Window Server.
Base Path	The path for applications on this computer. In many instances, this is the home directory of the defined username. When Deployment Management transfers a file or runs a command on this server, it logs in and changes directories to this base path before proceeding. Note that the directory separators should utilize forward slashes (`/'), even for Windows systems.
Connection Protocol	The protocol to be used for server or client connections. SSH is a secure connections protocol, whereas Telnet is not. To use SSH as your connection protocol, you must first set up SSH on the destination machine.
Transfer Protocol	The protocol to be used for moving files between various clients and servers. SCP is a secure transfer protocol, whereas FTP is not. In order to use SCP as your transfer protocol, you must first set up SCP

Field Name	Description
	on the destination machine.
Stream Encoding	Character encoding scheme used by the client.
Enable Client	Make the client information available to the system. Select the checkbox to make the client information available to the system.

5. In the Environment window, in the **Database** section, type the database server type in the **Server Type** field.

   - If **Oracle Server** is selected, complete the fields as specified in the following table.

Field Name	Description
Host Name	The DNS name or IP Address of the system running the Oracle database instance.
Connect String	The Oracle SQL*NET connection string used to connect to this database from a remote system. This is usually the same as the Oracle SID.
User Name	The username for the schema in the database that will be used by Deployment Management to make remote database connections.
Password	The password for the given username. The password is hidden, and can be changed by clicking the button to the right of the field.
Oracle SID	The SID of the database. This is used with the Port Number for Environment Comparison reporting.
Port Number	The port number of the TNS Listener database. Used for Environment comparison reporting.
DB Link	Reference to a remote Oracle database, if applicable.
Version	The Oracle version number for this database. Used for reporting purposes.
Enable Database	Make the database information available to the system. Select the checkbox to make the database information available to the system.

   - If **SQL Server** is selected, complete the fields as specified in the following table.

Field Name	Description
Server Name	The DNS name or IP Address of the system running the SQL

Field Name	Description
	database.
DB Name	The DB name used by Deployment Management when connecting to the SQL database.
User Login	The login ID used by Deployment Management when connecting to the SQL database.
Password	The password for the given login ID.
Port Number	The port number of the database. Used for Environment comparison reporting.
Version	The SQL version number for this database. Used for reporting purposes.
Enable Database	Make the database information available to the system. Select the checkbox to make the database information available to the system.

Note: For SQL Server database, PPM Center supports ISQL utility only. Since ISQL utility is not supported in SQL Server 2005 and later versions, PPM Center does not support SQL Server 2005 and later.

6. Click **OK**.

Using Application Code Environments

Complex environments are often segmented into subsections called environment applications. The environment information consists of the default set of attributes for an environment. It is rare, however, that an actual environment could be described simply by this set of defaults. For example, files belonging to different applications may reside at different paths and may be owned by different users. SQL scripts may need to be run against a different schema than the one defined on the **Host** tab.

When adding a line to a package, you have the option to choose the application code to specify the application that the migration object belongs to. When the object is subsequently migrated, the application-specific environment items are referenced in place of the default environment items. As a general rule, any application-specific environment item that has no value is substituted by the corresponding environment value.

Environment User Data fields are inherited by each application code and are displayed on the application code's **User Data** tab. **Application Code User Data** fields behave like other application code fields (such as host name and base path), in that blank field values indicate that the application code has the same value as its parent environment.

Each environment can contain its own set of applications.

Application fields do not always have to be populated. Click **New App** to open the New Application Code window to define an application code.

To define the application codes:

1. From the Workbench shortcut bar, select **Environments > Environments**.

   The Environments Workbench opens.

2. Open an environment.

   The Environment window opens.

3. Click the **Applications** tab.

4. Click **New App**.

   The New Application Code window opens.

5. Complete the fields in the New Application Code window as specified in the following table.

Field Name	Description
App Code	Short name abbreviation for the application.
Application Name	Long name for the application.
Description	A description of the application.
Server User Name	User name used to log on when transferring files or running commands on the environment server for this application, if it is different from the default server username.
Server Password	Password for logging on to the server, if different from the default server password. This field is encrypted.
Server Base Path	Base path for the application on the server machine.
Client User Name	User name that HPE should log in as when transferring files or running commands on the environment client for this application, if different from the default client name.
Client Password	Password for logging on to the client, if different from the default client password. This field is encrypted.
Client Base Path	Base path for the application on the client machine.

Field Name	Description
DB Username	Database username for the application. It is used when running database level commands (such as SQL scripts) for this application.
DB Password	Database password for the application. It is used when running database level commands (such as SQL scripts) for this application. This field is encrypted.
DB Link	Name of the database link for this application, if different from the default environment database link.
Enabled	Indicates whether this application environment is enabled.

6. Click **OK**.
7. From the **Applications** tab, click **OK**.

Copying Application Codes from Other Environments

When generating a new environment, all or some of the applications attached to an existing environment can be copied to the new environment to speed up the setup process.

To copy the application codes:

1. From the Workbench shortcut bar, select **Environments > Environments**.

   The Environments Workbench opens.
2. Open an environment.
3. Click the **Applications** tab.
4. Click **New App**.

   The New Application Code window opens.
5. Click **Copy Apps From**.

   The Copy From Dialog window opens.
6. In the **Environment Name** field, select the environment from which to copy the applications.

   The application code names are listed.
7. To change the base path segment of all the application codes selected, type the old server and client base path segment and the new server and client base path segment in their respective fields.

8. Select all the applications to copy.
9. Click **Add**.

   These fields have, by default, the server or client base path of the environment from which the applications are being copied. For example, two applications have been selected. The server base paths for these two applications are `/u2/apps/isi` and `/u2/apps/demo_107`. To change `u2` to `u3`, type `u2` in **Old Server Base Path** and `u3` in **New Server Base Path** before you copy these applications. Every occurrence of the old server and client base path segment is changed to the new base path segment in all the selected applications. The changes will be reflected in the applications in the environment into which they were copied.
10. After copying the application codes, any necessary modifications such as adding additional applications, deleting applications, or editing any of the applications can be made.
11. Click **OK**.

Setting Ownership and Participants for Environments

Ownership groups are defined by adding security groups to the **Ownership** tab. If no ownership groups are associated with the entity, the entity is considered global and any user who can edit environments can edit, copy, or delete the environment. For more information about access grants, see the *Security Model Guide and Reference*.

If a security group is disabled or loses its ability to edit environments, that group can no longer edit the entity.

Adding Ownerships to Environments

To add an ownership:

1. From the Workbench shortcut bar, select **Environments > Environments**.
2. Open an environment.

   The Environment window opens.
3. Click the **Ownership** tab.

4. Select the ownership option.
   - **All users with the Edit Environments Access Grant**
   - **Only groups listed below that have the Edit Environments Access Grant**

   For **Only groups listed below that have the Edit Environments Access Grant**:

   a. Click **Add**.

      The Add Security Groups window opens.

   b. Select the security groups and click **OK**.

      The security groups are added to the **Ownership** tab.

5. From the **Ownership** tab, click **OK**.

Deleting Ownerships from Environments

To delete an ownership:

1. From the Workbench shortcut bar, select **Environments > Environments**.
2. Open an environment.
3. Click the **Ownership** tab.
4. Select an ownership.
5. Click **Remove**.
6. Click **OK**.

Adding Participants to Environments

You can control which users can access an environment for use in environment groups and workflows.

To add participants to the environment:

1. From the Workbench shortcut bar, select **Environments > Environments**.
2. Open an environment.
3. Click the **User Access** tab.

4. Select one of the following options:
   - **All Users**
   - **Only Users in the groups listed below**

   If you selected **Only Users in the groups listed below**:

   a. In the **User Access** tab, click **Add**.

      The Add Security Group window opens.

   b. Select the security groups, and then click **OK**.

      The security groups are added to the **User Access** tab.
5. From the **User Access** tab, click **OK**.

Deleting Participants from Environments

To delete participants from the environment:

1. From the Workbench shortcut bar, select **Environments > Environments**.

   The Environments Workbench opens.
2. Open an environment.
3. Click the **User Access** tab.
4. Select a participant to delete.
5. Click **Remove**.
6. Click **OK**.

Environment Maintenance and Utilities

Testing Environment Setups

To check the validity of the Environment:

1. From the Workbench shortcut bar, select **Environments > Environments**.
2. Open an environment.
3. Click **Check.**

   The Check Environment window opens.
4. Select the environment connections to check.

   Select a folder, such as Server, to check all connections defined for that category. Specific connections can also be tested by selecting the individual checkboxes by the connection item.
5. Click **Check**.

   The system verifies the environment definition. Results from the environment check commands are displayed in the Log File section of the Check Environment window. Use log file output to troubleshoot any connection problems identified during the environment check.

   Environment definition testing includes actions performed during regular code migration, such as opening a Telnet session to the server, opening an FTP session to the server, and connecting to the database. While the environment checker cannot guarantee that all migrations will be successful, it can help catch some of the most common setup problems.

   While the check process can take a significant amount of time, it is recommended that any new environment is checked after all the data for it is provided. Additionally, it is good practice to periodically check all environments for obvious problems, such as changed passwords or disabled accounts.
6. In the Check Environment window, click **Cancel**.
7. In the Environment window, click **Cancel**.

Mass Updates of Base Paths

It is possible to update server and client base path segments in the environment and all of its applications at the same time. This functionality is useful to relocate a particular environment and all of its applications onto a new disk or partition.

To perform a mass update of the base paths:

1. From the Workbench shortcut bar, select **Environments > Environments**.
2. Open an environment.
3. From the menu, select **Environment > Update Base Paths**.

The Update Base Path window opens.

4. Type the old server/client base path segment and the new server/client base path segment in the fields provided.

   The default value in the old server/client base path field is the environment's current server/client base path segment.

5. Click **Apply** or **OK**.

   Every occurrence of the old server/client base path segment will be replaced by the new server/client base path segment in the Environment and all of its applications.

   For example, two applications have been selected. The server base paths for these two applications are `/u2/apps/isi` and `/u2/apps/demo_107`. To change `u2` to `u3`, type `u2` in **Old Server Base Path** and `u3` in **New Server Base Path** before you copy these applications. Every occurrence of the old server/client base path segment is then changed to the new base path segment in all selected applications. The changes are reflected in the applications in the Environment into which you copy them.

6. Click **OK**.

Environment Password Management Utility

When a single user has access to multiple password-protected environments, it is often convenient to use a single username and password for all those environments that the user encounters. If the password needs to be changed or if the user's job functions are transferred to another user, the Environment Password Management Utility enables the user to update the password in all the environments located on a single host, simultaneously. This utility only updates passwords with matching parameters, such as Hostname, Username, Old Password, and Connect String. These updates can be made using the Environments interface.

To change the environment password for a user:

1. From the Workbench shortcut bar, select **Environments > Environments**.

2. Open an environment.

3. From the menu, select **Environments > Update Password**.

   The Update Environments Password window opens.

4. In the **Host Type** field, select the host type.

The required fields change to match requirements of the selected host type.

5. Complete the fields in the Update Environments Password window.

Field Name	Description
Host Name	The DNS name or IP Address of the system running the Oracle database instance.
NT Domain	The domain name to use if this is a Windows Server.
Username	The username for the schema in the database used by HPE to make remote database connections.
Old Password	The old password for the given username. The password is hidden, and can be changed by clicking the button to the right of the field.
New Password	The new password for the given username. The password is hidden, and can be changed by clicking the button to the right of the field.

6. Click **OK.**

7. Click **OK.**

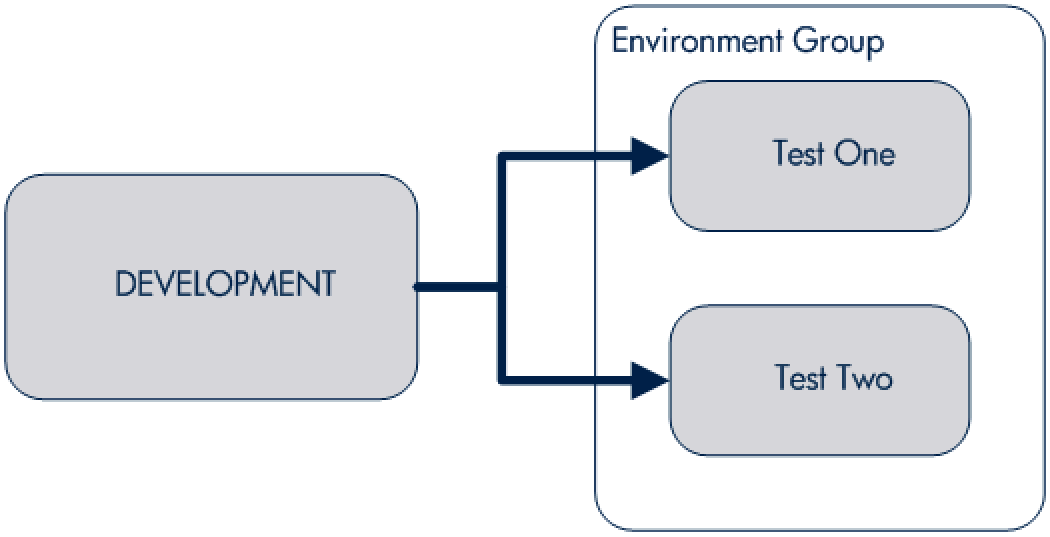
Overview of Environment Groups

Environment groups define a set of environments, which can be referenced as the source or destination for object migrations. Environment groups are defined and edited using the Environment Group Workbench.

Use environment groups where you want to run a workflow step on multiple environments. For example, it may be necessary to migrate an object to multiple testing environments for different targeted tests. These multiple environments can be referenced together as one environment group.

Note: The `ksc_connect` command works with environment groups in the "At least one" mode. If the `ksc_connect` command attempts to connect to the first server in the environment group and it fails to connect, then the `ksc_connect` command tries to connect to the second server in the environment group, and then the third, and so on. When the `ksc_connect` command succeeds in connecting to a server, the remaining commands in the object type are run before the command tries to connect to the next server in the environment group. If the `ksc_connect` command fails to connect to all servers in the environment group, the command exits with a failure.

Figure 7-6. Environment group



Overview of Configuring Environment Groups

Environment groups are configured in the Environment Group window. The following lists the main areas found in the Environment Group window:

- **General information**: Includes basic information concerning the environment group, such as the environment group name and description. For more information, see "Opening Environment Group Workbench" on the next page.
- **Environments**: Use to add existing Deployment Management environments to an environment group.
- **Host**: Lists basic information about the client, server, and database for the associated environments. For more information, see "Creating Environment Groups " on page 182.
- **Application Codes**: Lists basic information about the application codes linked to the associated environments.
- **Serial Execution Order**: Displays the order in which environments are acted on. For more information, see "Setting Order of Executions" on page 182.
- **Ownership**: Configure who can edit the environment group. For more information, see "Setting Ownership and Participants for Environment Groups" on page 183.
- **User Access**: Configures participants in the environment group. Participants can be given specific

access rights to the environment group. For more information, see "Adding Participants to Environment Groups" on page 184.

Opening Environment Group Workbench

To open the Environment Group Workbench:

1. Log on to PPM.
2. From the menu bar, select **Open > Administration > Open Workbench**.

   The PPM Workbench opens.
3. From the shortcut bar, select **Environments > Environment Groups**.

   The Environment Group Workbench opens.

Configuring General Information for Environment Groups

To configure the general information of an environment group:

1. From the Workbench shortcut bar, select **Environments > Environment Groups**.

   The Environment Group Workbench opens.
2. Open an environment group.
3. Type the information specified in the following table.

Field Name	Description
Environment Group Name	The name of the environment group.
Enabled	Makes the environment available to the system. Select **Yes** to make the environment available to the system.
Description	A brief description of how the environment group is being used.

Field Name	Description
Execution Order	Indicates whether the environments associated with an environment group are run all at once or in a specific serial order. Enables the **Serial Execution Order** tab when **Serial** is selected.
Source Environment (No App Code)	Indicates the default environment selected when the environment group is used as the source environment in an execution workflow step with no application code specified.

4. To save the changes to the environment, do one of the following:
   - Click **OK** to save the changes and close the window.
   - Click **Save** to save the changes and leave the window open.
5. Click **Cancel** to cancel the changes and close the window.

Creating Environment Groups

To define a new environment group:

1. From the Workbench shortcut bar, select **Environments > Environment Groups**.

   The Environment Group Workbench opens.
2. Open an environment group.
3. In the **Available Environments** field, select an environment name.
4. Click the right arrow to move the selected environment to the **Associated Environments** field.
5. Click **OK**.

   The changes to the environment group are saved.

Setting Order of Executions

The environments are run in sequential order until all are complete. Each environment execution waits for the previous environment execution to complete (success or fail) before beginning.

To set the environment execution order:

1. From the Workbench shortcut bar, select **Environments > Environment Groups**.

   The Environment Group Workbench opens.
2. Open an environment group.
3. Click the **Serial Execution Order** tab.
4. Select a row to move.
5. To move the selected environment to a new sequence position, use the arrow pointers under the tab.
6. When you are done, click **OK**.

Setting Ownership and Participants for Environment Groups

Ownership groups are defined by adding security groups to the **Ownership** tab. If no ownership groups are associated with an environment group, that environment group is treated as global, and any user who can edit environments for the entity can edit, copy or delete the entity. If a security group is disabled, its members can no longer edit the environment group. For more information on access grants, see the *Security Model Guide and Reference*.

Adding Ownerships to Environment Groups

Different groups of users can have exclusive control over the environment groups used by their group. These groups are referred to as ownership groups. Members of an ownership group can edit, delete or copy the environment group. Each environment group can be assigned multiple ownership groups.

Ownership groups are defined by adding security groups to the **Ownership** tab.

To set the ownership for an environment group:

1. From the Workbench shortcut bar, select **Environments > Environment Groups**.

   The Environment Group Workbench opens.
2. Open an environment group.
3. Click the **Ownership** tab.

4. Select which users will have the ability to edit this environment group.
   - **All users with the Edit Environments Access Grant**
   - **Only groups listed below that have the Edit Environments Access Grant**

   If **Only groups listed below that have the Edit Environments Access Grant** is selected:

   a. Click **Add.**

      The Add Security Groups window opens.

   b. Select the security groups and click **OK**.

   The security groups are added to the **Ownership** tab.

5. Click **OK**.

Deleting Ownerships from Environment Groups

To delete an ownership:

1. From the Workbench shortcut bar, select **Environments > Environment Groups**.

   The Environment Group Workbench opens.

2. Open an environment group.
3. Click the **Ownership** tab.
4. Select the ownership to remove.
5. Click **Remove**.
6. Click **OK**.

Adding Participants to Environment Groups

To add participants to the environment:

1. From the Workbench shortcut bar, select **Environments > Environment Groups**.

   The Environment Group Workbench opens.

2. Open an environment group.

3. Click the **User Access** tab.
4. Under **This Environment Group can be used in Workflows by** field, select one of the following.
    - To allow all users to edit this environment, select **All Users**.
    - To allow only members of a listed security group to edit the environment, select **Only Users in the groups listed below**.

    If you select **Only Users in the groups listed below**:

    a. Click **Add**. The Add Security Group window opens.

    b. Select a security group, and then click **OK**. The **User Access** tab displays the security group name.
5. Click **OK**.

Deleting Participants from Environment Groups

To delete participants from an environment group:

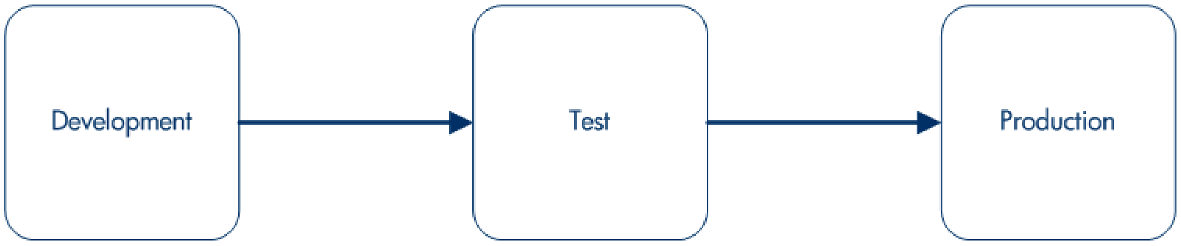
1. From the Workbench shortcut bar, select **Environments > Environment Groups**.

    The Environment Group Workbench opens.
2. Open an environment group.
3. Click the **User Access** tab.
4. Select a participant to delete, and then click **Remove**.
5. Click **OK**.

Overview of Environment Refresh

Deployment Management enables controlled migration of software and application changes to various instances. Deployment Management maintains an audit trail of activity, providing visibility into the changes for each environment. There are, however, situations when you need to refresh an environment, essentially replacing that environment with a physical copy of another environment.

Figure 7-7. Example software migration



If a standard software migration is from a Development environment to a Test environment to a Production environment, you may want to periodically copy the Production environment into both the Development and Test environments. This activity would involve copying all objects at the file system and database level from Production into the Development and Test areas, producing mirror images of the Production environment. This refresh function synchronizes the environments, and moves additional production data into development and testing environments.

After using the refresh function, Deployment Management has the ability to update its audit history and its in-process packages. This update is performed using Deployment Management's Environment Refresh Workbench functionality.

The environment refresh function in Deployment Management does not perform the actual physical refresh of one environment with another. It updates the Deployment Management data to be consistent with the refreshed physical occurrence.

Deployment Management's Environment Refresh performs the following functions:

- Identifies open package lines that have migrated through the environment and are now inaccurate due to the software changes. In the example described in "Figure 7-7. Example software migration" above, the refreshed environments would be Development and Test, while the source environment would be Production. After the physical refresh, these package lines will be inaccurate due to the software changes no longer being present in the refreshed environment.
- Updates the list of affected package lines as necessary, including adding or deleting lines from the list.
- Updates the Deployment Management internal object inventory tables to reflect the physical refresh. This copies the audit history from the source environment to the audit history of the refreshed environment. Because they are now physical matches of each other, their audit history should be the same.
- Updates the open package lines in the refresh list and resets them so they are eligible for migration into the refreshed environment again.

Overview of Configuring Environment Refresh

Environment refresh consists of an Environment Refresh window. Typically, environment refreshes are performed as part of maintaining environments between test systems.

The following lists the major sections found in the Environment Refresh window:

- **General Information.** General information stores the basic information for the refresh group, such as the names of the source and refreshed environment.
- **Refresh Lines tab.** The list of refresh lines includes basic, such as the package number and the line sequence, as well as the refresh status of the individual line. In addition to viewing refresh line information, several actions can be performed from the **Refresh Lines** tab:
    - **Add Line.** Manually adds a new refresh line to the list of refresh lines for this refresh group. This command generates the list of lines to be updated (such as a replacement of the calculate lines action), and adds additional lines to the list that might not be picked up by the calculate lines logic.

      When this action is selected, a prompt to add the refresh line information through the Add Line window appears. In this window, type a package number and select one or more package lines.

      This action can only be performed before the refresh group has been processed.
    - **Include Line and Exclude Line.** Use **Include Line** and **Exclude Line** to change the status of each individual refresh line on the list before the refresh group has been updated. All refresh lines are originally marked with a refresh status of **Pending.** When a refresh group is actually processed, using **Update Lines,** all lines in the list with a status of **Pending** are updated. Any refresh lines with a status of **Exclude** are not processed and are only used for reporting purposes.

      This action can only be performed before the refresh group has been processed.
    - **View Detail.** If a single refresh line is selected in the list, certain details regarding the refresh line can be viewed. Use this action to open up a read-only Line Detail window to view the information.
    - **Open Package.** For the complete details on the refresh line and the parent package, select a single line and click **Open Package** to navigate to the Package window and automatically open up the applicable package. Then view the complete package, as well as perform any eligible package activities.
- **User Data tab.** Each Deployment Management implementation can add custom fields that are

common to all refresh groups. These new user data fields essentially become additional refresh group header data. These fields are normally determined during initial system configuration. As part of the configuration, the layout of the fields is determined. This configuration includes:

- The look and feel of the field (whether the field is a drop-down list, or auto-complete list, or free-form text)
- The logic behind the field (how the information in the field validated)
- Graphic characteristics (such as field height, width, and placement in the tab)

Opening Environment Refresh Workbench

To open the Environment Refresh Workbench:

1. Log on to PPM.
2. From the menu bar, select **Open > Administration > Open Workbench**.

   The PPM Workbench opens.
3. From the shortcut bar, select **Deployment Mgmt > Environment Refresh**.

   The Environment Refresh Workbench opens.

Configuring General Information for Environment Refreshes

To configure the general information of an environment refresh:

1. From the Workbench shortcut bar, select **Deployment Mgmt > Environment Refresh**.

   The Environment Refresh Workbench opens.
2. Click **New Env Refresh**.

   The Environment Refresh window opens.
3. The following information is required:
4. Click **Calculate Lines**.

This can only be performed if the status of the refresh group is New or Open. If the original status of the group was New, the status is updated to Open after the package lines are added.

This action is not reversible. After performed, the status of the refresh group is changed to Completed.

This action queries all open package lines and generates a list of lines that have been migrated to the environment being refreshed (not to the source environment). After the physical refresh, these package lines will be inaccurate because the software changes are no longer present in the refreshed environment. This action only looks for packages using workflows containing both the source and refreshed environment.

All package lines matching this criteria will be added to the package lines list for the refresh group. If the refresh group already contains a list of package lines, this list will be deleted before the new calculation of eligible lines is performed.

5. Click **Update Lines**.

   This action is not reversible. After performed, the status of the refresh group is changed to Completed. After all the applicable package lines have been added to the lines list and reviewed, this action is used to perform the actual update of the Deployment Management data. This action performs:

   - The internal object inventory tables are updated to reflect the physical refresh. This copies the audit history from the source environment to the audit history of the refreshed environment. Because they are now physical matches of each other, their audit history should be the same.
   - All package lines in the refresh lines list that have a refresh status of Include are reset to be eligible for migration into the refreshed environment again. This is done using special workflow transitions from the currently eligible step to the migration step for the refreshed environment. This allows all the changes that were erased during the physical refresh to be migrated back into the environment. After each refresh line is updated, its refresh status is updated to Completed.

The refresh status is now completed.

Note: For refresh groups with a status of New and Open, it is possible to cancel the environment refresh. To cancel the environment refresh, click **Cancel Refresh.** This sets the status of the refresh group to Cancel, preventing any further action on the refresh group but allowing the details of the specific refresh group to be queried.

Configuring a Workflow with an Environment Refresh

For this example Development and Test are two PPM Servers.

To create a workflow environment refresh:

1. From the Workbench shortcut bar, select **Configuration > Workflows**.

   The Workflow Workbench window opens and the Workflow Step Resources window opens.

2. In the Workflow Workbench window, click **New Workflow**.

   A Workflow window opens.

3. In the Workflow window, fill in the required information.

4. In the Workflow window, select the **Layout** tab.

5. Go to the Workflow Step Resources window and expand the **Executions** folder.

6. Select a Deployment Management execution step, and then drag it onto the **Layout** tab of the Workflow window.

   The Workflow Step window opens.

7. In the Workflow Step window, complete the required fields:

   a. Type `Development` in the **Source Environment** field.

   b. Type `Test` in the **Dest Environment** field.

   c. Click **Apply**, and then click **OK** until you exit the Workflow Step window.

8. From the Workflow Step Resources window, drag another Deployment Management execution step onto the **Layout** tab of the Workflow window.

   The Workflow Step window opens.

9. In the Workflow Step window, complete the required fields.

   a. Type `Test` in the **Source Environment** field.

   b. Type `Development` in the **Dest Environment** field.

   c. Click **Apply**, and then click **OK** until you exit the Workflow Step window.

10. Add a transition from the first Deployment Management execution step to the second Deployment Management execution step.

   a. Right-click the first workflow step, and then select **Add Transition** on the shortcut menu. The Step Transitions window opens.

   b. Select the transition definition in the Define Transition window, and then complete the information in the Step Transitions window.

11. Add a Close step. From the Workflow Step Resources window, drag a Deployment Management Close step onto the **Layout** tab. (For information on adding a close step, see "Adding Close Step " on page 49.

    The Workflow Step window opens.

12. In the Workflow Step window, complete the required fields.

13. Add a transition from the second Deployment Management execution step to the Deployment Management Close step.

    The Step Transitions window opens.

14. Complete the information in the Step Transitions window, and then save and close the workflow.

    For information on adding transitions, see "Configuring Transitions for Workflow Steps" on page 67.

Configuring a Package with an Environment Refresh

To create a package environment refresh:

1. From the Workbench shortcut bar, select **Deployment Mgmt > Packages**.

   The Package Workbench opens.

2. Select the **Query** tab, and then click **New Package**.

   The Package window opens.

3. Type the required Package information.

4. Select the workflow configured in "Configuring a Workflow with an Environment Refresh" on the previous page.

5. In the Package window, click **New Line**.

   The Add Line window opens.

6. In the **Object Type** field, select **File Migration**.

7. In the Add Line window, complete the remaining fields, and then click **Add** and **OK**.
8. Click **Submit** to run the package through to completion.

   You may need to set the override status to Succeed in the **Status** tab.

Chapter 8: Configuring Notification Templates

- "Overview of Notification Templates " below
- "Opening Notification Template Workbench " below
- "Creating Notification Templates " on the next page
- "Configuring Notification Intervals" on page 197
- "Checking Usage of Notification Templates " on page 198

Overview of Notification Templates

Notification templates are preconfigured notifications that you can use to quickly construct the body of a message. You can use notification templates with PPM tasks, projects, requests, packages, releases, workflows, and reports.

Opening Notification Template Workbench

To create a notification template, you use the Notification Template Workbench.

To open the Notification Template Workbench:

1. Log on to PPM.
2. From the menu bar, select **Open > Administration > Open Workbench**.

   The Workbench opens.
3. From the shortcut bar, select **Configuration > Notification Templates**.

   The Notification Template Workbench opens.

Deleting Notification Templates

You cannot delete notification templates that are referenced from an existing notification. To delete such a notification template, you must remove these references. Referenced notification templates can be disabled. For information on how to determine whether a notification template is referenced, see "Checking Usage of Notification Templates " on page 198.

Creating Notification Templates

To create a notification template:

1. From the Workbench shortcut bar, select **Configuration > Notification Templates**.

   The Notification Template Workbench opens.

2. Click **New Notification Template**.

   The Notification Template window opens.

3. In the **Template Name** field, type a name for the template.

4. To indicate the range of use for this new notification, from the **Notification Scope** list, select a PPM product area.

   > **Note:** The default notification scope is **Packages.** Selecting a different scope changes the notification template format.

5. From the **Notification Format** list, select **Plain Text** or **HTML**.

6. To make this template available in PPM, for the **Enabled** option, leave **Yes** selected.

7. To make this template the default notification template for PPM, for the **Default** option, select **Yes.**

8. To provide a **From** address:

   a. In the **From** row, click **Choose.**

      The Email Header Field window opens.

   b. From the list at the top of the window, select the sender category.

The context-sensitive required field is dynamically updated to gather the necessary information for that category. For instance, if **Enter an Email Address** is selected from the list, then it is necessary to provide an Email Address. If you select **User Defined Token,** click **Tokens** to bring up a complete list of available tokens or type in a specific token.

   c. Type the applicable information in the required field.

   d. If a user-defined token has been provided, select the token type that corresponds to the evaluated token value.

   e. In the Email Header Field window, click **OK**.

9. In the Notification Template window, type a **Reply** address, as follows:

   a. Next to **Reply To,** click **Choose**.

      The Email Header Field window opens.

   b. Select the recipient category.

      The context-sensitive required field is dynamically updated to gather the necessary information for that category. For instance, if **Enter an Email Address** is selected, then it is necessary to type an Email Address. If **User Defined Token** is selected, click **Tokens** to bring up a complete list of available tokens or type in a specific token.

   c. Type the information in the required field.

   d. If **User Defined Token** is provided, select the token type that corresponds with the evaluated token value.

   e. In the Email Header Field window, click **OK**.

10. In the **Body** field, type the body of the notification text.

    Ensure the format of the notification body is the same as that specified in **Notification Format.** To incorporate linked tokens, HTML notifications for Deployment Management must include the `[NOTIF.NOTIFICATION_DETAILS]` token within the within the **Body** field.

11. In the **Body** field, add tokens to the body of the text.

    To add tokens to the body of the notification template:

    a. Click **Tokens**.

       The Token Builder window opens.

    b. Select a token.

   c. In the **Token** field, copy the name of the token and paste the name in the **Body** field.

   d. Click **Close**.

12. Configure the ownership of the notification template.

    For detailed information about how to configure the ownership of the notification template, see "Configuring Ownership of Notification Templates" below.

13. Click **OK**.

Configuring Ownership of Notification Templates

Ownership groups are defined by adding security groups to the Ownership window. If no ownership groups are associated with the entity, the entity is considered global and any user with the edit access grant for the entity can edit, copy or delete it. For more information about access grants, see the HPE.

If a security group is disabled or loses the edit access grant, members of that group can no longer edit the entity.

To configure the ownership of a notification template:

1. From the Workbench shortcut bar, select **Configuration > Notification Templates**.

   The Notification Template Workbench opens.

2. Open a notification template.

3. At the bottom of the window, click **Ownership**.

   The Ownership window opens.

4. Select one of the following ownership options:

   - **All users with the Edit Notification Templates Access Grant**
   - **Only groups listed below that have the Edit Notification Templates Access Grant**

   If **Only groups listed below that have the Edit Notification Templates Access Grant** is selected:

   a. Click **Add.**

      The Add Security Groups window opens.

   b. In the **Security Groups** field, select the security groups.

   c. Click **OK**.

      The **Ownership** tab lists the selected security groups.

5. Click **OK**.

   The changes to the notification template are saved.

Deleting Ownerships from Notification Templates

To delete an ownership:

1. From the Workbench shortcut bar, select **Configuration > Notification Templates**.

   The Notification Template Workbench opens.

2. Open a notification template.
3. Click **Ownership**.

   The Ownership window opens.

4. Select an ownership to remove.
5. Click **Remove**.
6. Click **OK**.

Configuring Notification Intervals

To create a new notification template:

1. From the Workbench shortcut bar, select **Configuration > Notification Templates**.

   The Notification Template Workbench opens.

2. From the top menu, select **Notification Templates > Intervals**.

   The Notification Intervals window opens.

3. Click **New.**

   The Notification Interval: New window opens to the **Interval** tab.

4. Type the information specified in the following table.

Field Name	Description
Interval Name	Name assigned to the interval.
Description	Optional description of the interval.
Interval Type	For internal use. This is always set to **Periodic,** unless **Immediate Interval** is used.
Start Time	Time to start sending out notifications and to start counting down the time interval until the next batch.
End Time	Time to stop sending out notifications.
Time Interval (Hours)	Number of hours to wait after the start time or the last batch is sent, before sending out the next batch of notifications.
Days	Used to select which days this interval is to run.
Enabled	If set to **Yes,** this interval is selectable. If set to **No**, this interval is unavailable.

5. Click **OK**.
6. Click **Close**.

   The new notification interval can now be used in any workflow step notification.

If notifications are sent at an hourly or daily interval, there are sometimes several notifications pending for a particular user. In this case, all notifications are grouped together in one email message. The subject of each notification is displayed in a **Summary** section at the top of the email message.

Checking Usage of Notification Templates

To check the usage of a notification template:

1. From the Workbench shortcut bar, select **Configuration > Notification Templates**.

   The Notification Template Workbench opens.
2. Open the notification template.
3. Click **Used By**.

   The Used By window opens and lists all references to the notification template.

4. Click **OK**.
5. In the Notification Template window, click **OK**.

Chapter 9: Configuring User Data

- "Overview of User Data " below
- "Opening User Data Workbench " on page 202
- " Viewing General Information for User Data Types" on page 202
- "Creating a User Data Context" on page 203
- "Creating User Data Fields" on page 204
- "Configuring User Data Layouts " on page 211

Overview of User Data

Product entities, such as packages, workflows, requests, and projects, include a set of standard fields that provide information about the entities. While these fields are normally sufficient for day-to-day processing, you can create *user data fields* to capture additional information specific to your organization. For example, if you want to include an additional field on every package, you can open the **Validation Value User Data** user data type (with global scope) and define the extra field, which is then displayed on the **User Data** tab for a validation.

You configure user data types from the User Data Context window in the User Data Workbench.

The following four columns in the User Data Workbench define the components that fully define a user data type:

- **User Data Type**: Displays the user data type names, which are predefined and uneditable in PPM.

  Although you cannot create new user data types, you can create new user data *contexts* (based on the Validation Value User Data, the Package User Data, or the Environment User Data types) and define user data fields for them.

- **Scope**: Displays the scope of the user data type field. The two possible scope values are:
  - **Global.** If the user data type field has a global scope, the **User Data** tab for every designated entity contains the defined user data field.
  - **Context.** If the user data type field has a context scope, then the defined user data field is added only to the **User Data** tab for entities that have specific **Context Field** and **Context Value** definitions.

- **Context Field**: Displays the context-sensitive fields. It applies only to user data type fields with context scope. Because each user data type only has one available context field value, the cells in this column are populated automatically.
- **Context Value**: Lists the value (context) for the context-sensitive field. It applies only to user data type fields with a context scope. You cannot create a new context value. You can only assign an existing one.

You can define up to 20 user data type fields for display on the **User Data** tab of a defined entity. You can configure the major attributes of each field, including its graphical presentation, the validation method, and whether it is required.

Referencing User Data

After you have a user data field, you can refer to it from other parts of the product (in notifications and command executions) by using its token name, preceded by the entity abbreviation and the user data (UD) qualifier. For example, Validation Value User Data might have the field "Class Name" with the token value `CLASS_NAME`, and the user data qualifier `USER_DATA1`.

Migrating User Data

For any configuration entity that has user data type fields, the data in the user data type fields is migrated with the entity.

- If two instances have identical user data configurations, then the user data is migrated correctly.
- If two instances do not have identical user data configurations, then the user data is mapped to the data model according to the storage configuration in the source instance. Check to ensure that the two instances have the same user data fields. Otherwise, you must correct the user data after migration.
- If the user data is context-sensitive, then a corresponding context-sensitive configuration must exist in the destination instance, or the migration fails.
- User data fields that have hidden and visible values can cause problems. If the hidden value of a user data field refers to a primary key (such as Security Group ID) that is different in the source and destination instances, the migrator does not correct the hidden value. In this case, you must correct the user data manually, after migration.

User Data Configuration Tasks

To configure user data, complete the tasks in "Table 9-1. User Data Configuration Tasks" below.

Table 9-1. User Data Configuration Tasks

Task	See
Open the User Data Workbench	"Opening User Data Workbench " below
Open a user data type and view general information	" Viewing General Information for User Data Types" below
Create user data fields	"Creating User Data Fields" on page 204
Configure user data field layout	"Configuring User Data Layouts " on page 211

Opening User Data Workbench

1. Log on to PPM.
2. From the menu bar, select **Open > Administration > Open Workbench**.

   The Workbench opens.
3. From the shortcut bar, select **Configuration > User Data**.

Viewing General Information for User Data Types

From the User Data Workbench, open a user data type, or create a new user data context. The User Data Context window opens.

The following table lists descriptions of the fields in the User Data Context window:

Field Name	Description
User Data Type	User data type name. For global user data types, this field is automatically populated. If you are creating a context-sensitive user data context, you select the type from the list.

Field Name	Description
Context Field	For user data types and user data contexts that have context scope, this field is automatically populated with the name of the context-sensitive field. The **Context Field** auto-complete list is only enabled for the Environment User Data and Package User Data user data types.
Context Value	For context-sensitive user data types, this field displays the value for the context field. This field is disabled for user data types with global scope. You can only define one context value for the context field. For example, you cannot have two context-sensitive user data types with the same context field and context value (such as a field labeled **Priority** with a value of "Critical").
Enable	Use this option to enable (defaults to yes) or disable the user data type in PPM.
Scope	Scope of the user data type. This field is automatically populated based on the user data type. The possible scopes for a user data type are: • **Global.** Standard user data type scope. If the scope is global, the **User Data** tab for every designated entity displays the defined fields. • **Context.** Indicates that this is a context-sensitive user data type. If the user data type has the context scope, the **User Data** tab displays the defined fields only if the designated entities have the correct **Context Field** and **Context Value** definitions.
Meta Layer View	Meta layer views relate information specific to PPM. For example, the reporting meta layer view MREQ_OPENED_CLOSED_BY_TYPE_D provides summary information for request submission and completion activity, broken down by request type and by calendar day.

Creating a User Data Context

Although you cannot create a new user data type in PPM, you can create a user data context that is based on one of the following user data types:

- **Validation Value User Data.** Create user data fields for a named drop-down list validation. Typically, you create this new user data context to associate more data with values available for users to select.

  Example: Your PPM system has a **US States** drop-down list validation that has 50 validation values. Somewhere else in the system, you need to get the capital of the state that a user has selected. So, you create a new user data context for the **US States** list and add the **Capital** field to it.

You next open the **US States** drop-down list validation, and for each validation value (state), you complete the **Capital** field. Now, the system can detect which state a user has selected and pick up the capital.

- Environment User Data
- Package User Data

To create a new user data context for a drop-down list validation:

1. From the Workbench shortcut bar, select **Configuration > User Data**.

   The User Data Workbench window opens.

2. On the **Query** tab, click **New User Data Context**.

   The User Data Context Window opens.

3. Click the **User Data Type** auto-complete list.

   The **User Data Type** field displays the value "Validation Value User Data," the **Context Field** field displays the value "Validation Name," and the **Scope** field displays "Context."

4. Use the **Context Value** auto-complete list to select a validation value for the **Validation Name** context field.

5. Create one or more user data fields.

   For information on how to create a user data field, see "Creating User Data Fields" below.

6. Click **OK**.

Creating User Data Fields

Not all user data field types have **Dependency** and **Security** tabs.

To create a user data field:

1. From the Workbench shortcut bar, select **Configuration > User Data**.

   The User Data Workbench window opens.

2. Open a user data type, or create a new user data context.

The User Data Context window opens to the **Fields** tab.

3. Click **New.**

   The Field: New window opens.

4. Type the information described in the following table.

Field Name	Description
Field Prompt	Label displayed for the user data field in the request.
Token	Uppercase text string used to identify the token. The token name must be unique to the specific user data. An example of a token name is ASSIGNED_TO_USER_ID.
Description	Type a description of the user data field in this field.
Enabled	To disable the field in PPM, select **No.** (The user data field is enabled by default.)
Validation	Use the **Validation** auto-complete list to specify the logic to use to determine the valid values for this field. This could be a list of user-defined values, a rule that the result must be a number, and so on. After you select the validation logic, the **Component Type** field displays the type of component (for example, drop-down list, text field, auto-complete list) used in the validation.
Multiselect	If the validation uses an auto-complete list component type and you want users to be able to provide multiple values, select **Yes**.

5. On the **Attributes** tab, type the information described in the following table.

Field Name	Description
User Data Col	Indicates the internal column in which the field value is to be stored. These values are then be stored in the corresponding column in the table for the given entity (such as KNTA_USERS for the users entity). User data provides the ability to store information in up to 20 columns, thus allowing for up to 20 fields. No two fields in user data can use the same column.
Display Only	Indicates whether the field is read-only. Select Use Dependency Rules to use the logic defined on the **Dependencies** tab.
Display	Indicates if the user sees this field on the **User Data** tab.
Required	Indicates whether the user must specify a value for this field. Select Use Dependency Rules to use the logic defined on the **Dependencies** tab.

6. Select the **Defaults** tab, and then type the information described in the following table.

Field Name	**Description**
Default Type	Defines if the field will have a default value. Either default the field with a constant value or default it from the value in another user data field.
Visible Value	If a default type of **Constant** is selected, the constant value can be typed here.
Depends On	To default from another field, choose the token name of that field. When using this user data, every time a value is typed or updated in the source field, it will automatically be typed or updated in this destination field.

7. Select the **Dependencies** tab, type the information specified in the following table, and then click **OK**.

Field Name	**Description**
Clear When The Following Changes	Indicates that the current field should be cleared when the specified field changes.
Display Only When	Indicates that the current field should only be editable when certain logical criteria are satisfied. The field functions with two adjacent fields, a list that contains logical qualifiers, and a text field. To use this functionality, select **Use Dependency Rules** in the **Attributes** tab.
Required When	Indicates that the current field should be required when certain logical criteria are satisfied. The field functions with two adjacent fields, a list that contains logical qualifiers, and a text field. To use this functionality, select **Use Dependency Rules** in the **Attributes** tab.

8. To specify the users who can view and edit this field, select the **Security** tab, type the information specified in the following table, and then click **OK**.

Field Name	**Description**
Visible to all users	Checking this option allows all users to see the field. If this option is not checked, you can set who can see the field. The default is for all users to be able to see a field. If this option is not checked, the Select User/Security Group that can view this field is enabled. Clearing the **Visible to all users** or **Editable by all users** checkboxes enables the Select Users/Security Groups that can view this field section of the Edit Field Security window.
Editable by all users	Checking this option allows all users to edit the field. If this option is not checked, you can set who can edit the field. The default is for all users to be able to edit a field.

Field Name	Description
	Clearing the **Visible to all users** or **Editable by all users** checkboxes enables the Select Users/Security Groups that can view this field of the Edit Field Security window.
Enter a Security Group (list)	Use to select the format for specifying users to grant visibility and edit permissions. The field displays the formats to choose users. The list dynamically updates the Security Group Validate auto-complete list. The choices are: ◦ **Enter a Username.** Select a specific user a to see and edit the field. The user must have an email address. ◦ **Enter a Security Group Name.** Select a specific security group to see and edit the field. ◦ **Enter a Standard Token.** Select a standard token to see and edit the field. ◦ **Enter a User Defined Token.** Select a user defined token to see and edit the field. Selecting this format enables the **Tokens** button. Selecting an item from this list dynamically updates the **Enter a Security Group** field.
Security Group	Provides a field for specifying the recipient. If the **Enter a Security Group** field displays: ◦ **Enter a Username,** then the Validate: Username window is returned. ◦ **Enter a Security Group,** then the Validate: Security Group window is returned. ◦ **Enter a Standard Token,** then the Validate: Standard Token window is returned. ◦ **Enter a User Defined Token,** then the Validate: User Defined Token window is returned.

9. Click **OK.**

   The Field window displays the new field.

10. Click **OK.**

Copying a Field Definition

You can streamline the process of adding fields by copying the definitions of existing fields.

To copy a field definition:

1. From the Workbench shortcut bar, select **Configuration > User Data**.

   >The User Data Workbench opens.

2. Select an existing user data type or create a new user data type.

   The User Data Context window opens to the **Fields** tab.

3. Click **New.**

   The Field: New window opens.

4. Click **Copy From**.

   The Field Selection window opens.

5. Type the required information, and then click **List**.

   > You can query fields using several criteria, including the token name or field prompt. You can also perform more complex queries. For example, you could list all fields that reference a specific validation or all fields that a specific entity uses.

   The Field Selection window refreshes with fields matching the search criteria.

6. Select a field to copy, and then click **Copy**.

7. Make any necessary changes, and then click **OK.**

Editing User Data Fields

To edit an existing field:

1. From the Workbench shortcut bar, select **Configuration > User Data**.

   The User Data Workbench opens.

2. Select an existing user data type or create a new user data type.

   The User Data Context window opens to the **Fields** tab.

3. Select the field to edit, and then click **Edit**.

   The Field window opens.

4. Make the required changes.

Be sure to include the **Attributes, Default,** and **Dependencies** tabs. For information about these tabs, see "Creating User Data Fields" on page 204.

5. Click **OK**.
6. In the User Data Context window, click **OK**.

Configuring User Data Field Dependencies

Field behavior and properties can be linked to the value of other fields defined for that entity. A **Report Type** field can become required when the value in another field in that report type is **Critical**.

You can configure a field to:

- Clear after the value in another field changes.
- Become read-only after another field meets a logical condition defined in "Table 9-2. Field dependencies" below.
- Become required after another field meets a logical condition defined in "Table 9-2. Field dependencies" below.

Table 9-2. Field dependencies

Logical Qualifier	Description
like	Looks for close matches of the value to the contents of the field chosen.
not like	Looks for contents in the selected field that are not close matches to the value field.
is equal to	Looks for an exact match of the value to the contents of the field chosen.
is not equal to	Is true when there are no results exactly matching the value of the field contents.
is null	Is true when the field selected is blank.
is not null	Is true when the field selected is not blank.
is greater than	Looks for a numerical value larger than the value typed in the value field.
is less than	Looks for a numerical value less than the value typed in the value field.

Table 9-2. Field dependencies, continued

Logical Qualifier	Description
is less than equal to	Looks for a numerical value below or the same as the value typed in the value field.
is greater than equal to	Looks for a numerical value larger than or the same as the value typed in the value field.

To configure a user data field dependency:

1. From the Workbench shortcut bar, select **Configuration > User Data**.

   The User Data Workbench opens.

2. Select an existing user data type or create a new user data type.

   The User Data Context window opens to the **Fields** tab.

3. Select the field, and then click **Edit**.

   The Field window opens.

4. Click the **Dependencies** tab.

5. Set the field dependencies, as follows:

   - In the **Clear When The Following Changes** field, select a field name to indicate that the current field is to be cleared if the selected field changes.
   - In the **Display Only When** field, select a field name to display the field only (for example, not editable) if specific logical criteria are satisfied. This field functions with a list that contains logical qualifiers and with another field that dynamically changes to a date field, list, or text field, depending on the validation for the selected field.
   - In the **Required When** field, select a field name to indicate that the field is to be required if certain logical criteria are satisfied. This field functions with a list that contains logical qualifiers and with a field that dynamically changes to a date field, list, or text field, depending on the validation for the selected.

6. Click **OK**.

7. In the Field window, click **OK**.

8. In the User Data Context window, click **OK**.

Removing Fields

To permanently remove a field from a user data type:

1. From the Workbench shortcut bar, select **Configuration > User Data**.

   The User Data Workbench opens.

2. Select an existing user data type or create a new user data type.

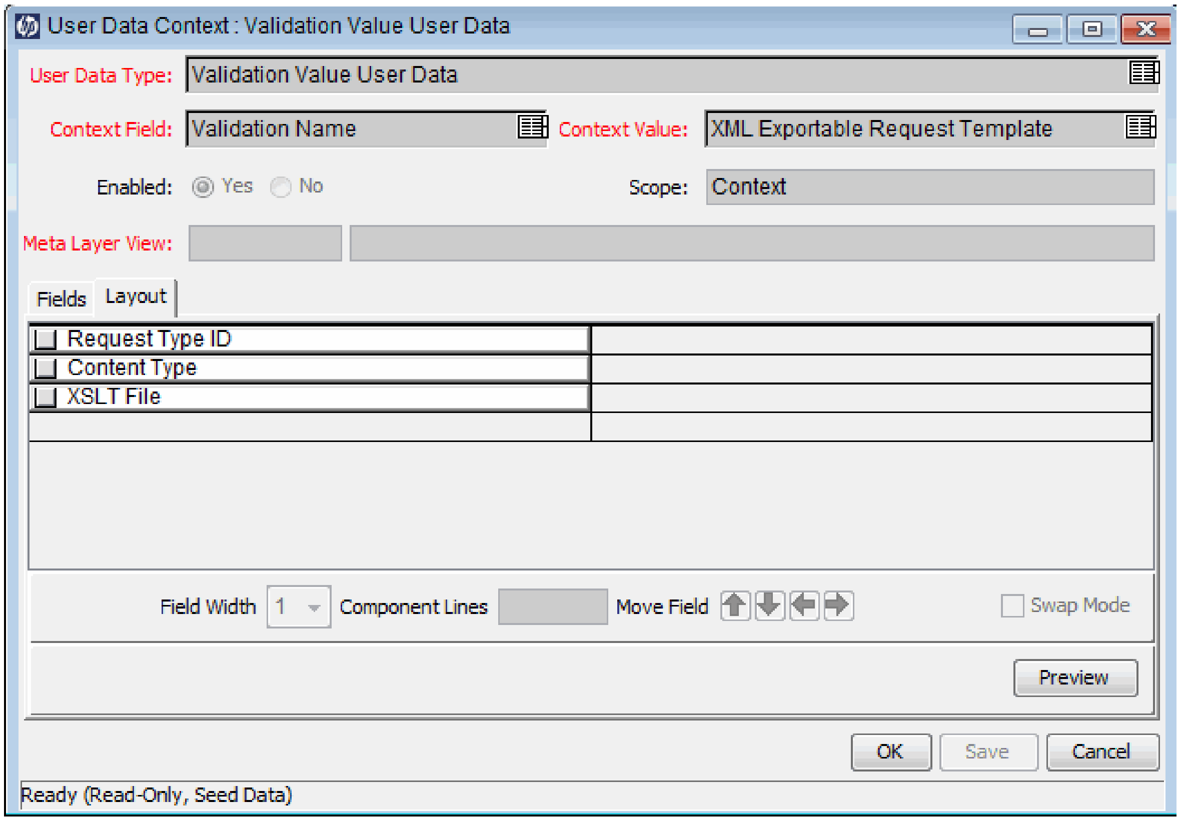
   The User Data Context window opens to the **Fields** tab.

3. Select the field to remove, and then click **Remove**.
4. Click **OK**.

Configuring User Data Layouts

The layout of user data fields can be changed in the **Layout** tab of the User Data Context window.

Figure 9-2. User Data window Layout tab


User Data Context : Validation Value User Data
User Data Type: Validation Value User Data
Context Field: Validation Name
Context Value: XML Exportable Request Template
Enabled: [✓ Yes] No
Scope: Context
Meta Layer View:
Fields
Layout
Request Type ID
Content Type
XSLT File
Field Width 1
Component Lines
Move Field
Swap Mode
Preview
OK
Save
Cancel
Ready (Read-Only, Seed Data)

Changing Column Widths

To change the column width of a field:

1. From the Workbench shortcut bar, select **Configuration > User Data**.

   The User Data Workbench opens.

2. Select an existing user data type or create a new user data type.

   The User Data Context window opens.

3. Select the **Layout** tab, and then select a field.

4. In the **Field Width** field, select either **1** or **2** inches.

   The Layout editor does not let you make changes that conflict with another field in the layout. For example, you cannot change the width of a field from **1** to **2** if another field exists in column two on the same row.

For fields of component type Text Area, you can determine the number of lines the text area will display. Select the Text Area type field and change the value in the Component Lines attribute. If the selected field is not of type Text Area, this attribute is blank and read-only.

5. Click **OK**.

Moving Fields

To move a field or a set of fields:

1. From the Workbench shortcut bar, select **Configuration > User Data**.

   The User Data Workbench opens.

2. Select an existing user data type or create a new user data type.

   The User Data Context window opens.

3. Click the **Layout** tab, and then select a field.

   To select more than one field, press `Shift` and then select the first and last fields in a set. You can only select adjacent fields.

   You cannot move a field to where another field exists.

4. Use the arrow pointers to move the fields in the layout builder.
5. Click **OK**.

Swapping Positions of Two Fields

To swap the positions of two fields:

1. From the Workbench shortcut bar, select **Configuration > User Data**.

   The User Data Workbench opens.

2. Select an existing user data type or create a new user data type.

   The User Data Context window opens.

3. Click the **Layout** tab, and then select a field.
4. Select the **Swap Mode** option.

An **S** is displayed in the option section of the selected field.

5. Double-click the field to swap positions with the selected field.
6. Click **OK**.

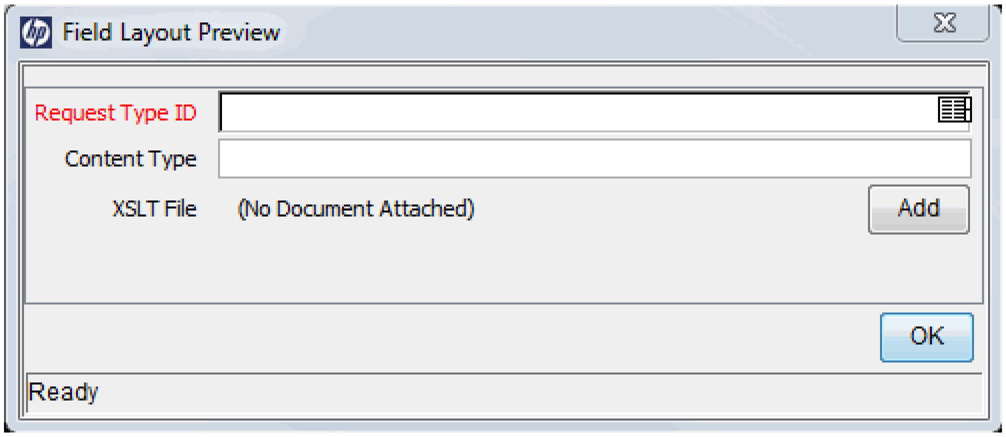
Previewing Layout

You can check to see what the layout looks like to users.

To preview field layout, select the **Layout** tab in the User Data Content window, and then click **Preview.**

A field layout window opens to show you a preview of the fields, as they are to be displayed.

Figure 9-3. Preview mode



If all fields have a width of one column, all displayed columns automatically span the entire available section when an entity of the given user data is viewed or generated.

Hidden fields do not affect the layout.

Appendix A: Worksheets

- "Configuration Workflow Worksheets" below
- "Execution Workflow Step Worksheets " on the next page
- "Decision Workflow Step Worksheets " on page 218
- "Subworkflow Workflow Step Worksheets " on page 219
- "Object Type Configuration Sheets" on page 221

Configuration Workflow Worksheets

Table A-1. Workflow skeleton

#	Step Name	Description	Type	Transition Values
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				

Table A-1. Workflow skeleton, continued

#	Step Name	Description	Type	Transition Values
16				
17				
18				
19				
20				
a. Type = Workflow Step Type: Decision (D), Execution (E), Condition (C), Subworkflow (S)				

Execution Workflow Step Worksheets

Table A-2. Workflow step [execution], step number ____

Step Field	Value
Step Name	
Goal/Result of Step	
Validation	See "Table A-3. Workflow step [execution], step number ____ validation" on the next page.
Execution Type	See "Table A-4. Workflow step [execution], step number ____ execution type" on the next page.
Processing Type	
Timeout (Days)	
Source Environment (Group)	
Dest Environment (Group)	
Security (who can act on step): • User Name • Standard Token	

Table A-2. Workflow step [execution], step number ____, continued

Step Field	Value
• User Defined Token	
Include Notification (Yes/No)	
Notification Event	
Notification Recipient: • Username • Email Address • Security Group • Standard Token • User Defined Token	
Notification Message	
Request Status at Step	
Request % Complete at Step	
Authentication Required (Y/N)	
Authentication Type (if Y)	

Table A-3. Workflow step [execution], step number ____ validation

Validation Information	Value
Existing Validation?	
New Validation?	
Validation Type: (text field, auto-complete list, drop-down list, and so on)	
Validation Definition (list of values or SQL)	

Table A-4. Workflow step [execution], step number ____ execution type

Execution Type	Value
Built-in Workflow Event: • Execute Commands • Close • Jump/Receive • Ready for Release • Return from Subworkflow	

Table A-4. Workflow step [execution], step number ____ execution type, continued

Execution Type	Value
PL/SQL Function	
Token	
SQL Statement	
Workflow step commands	

Decision Workflow Step Worksheets

Table A-5. Workflow step [decision], step number ____

Step	Value
Step Name	
Goal/Result of Step	
Validation	See "Table A-6. Workflow step [decision], step number _____ validation" on the next page.
Decisions Required (Vote on Step's outcome?)	• One • At Least One • All
Timeout (Days)	
Security (who can act on step): • Security Group • User Name • Standard Token	

Table A-5. Workflow step [decision], step number ____, continued

Step	Value
• User Defined Token	
Include Notification (Yes/No)	
Notification Event	
Notification Recipient: • Username • Email Address • Security Group • Standard Token • User Defined Token	
Notification Message	
Request Status at Step	
Request % Complete at Step	
Authentication Required (Y/N)	
Authentication Type (if Y)	

Table A-6. Workflow step [decision], step number ____ validation

Validation Information	Value
Existing Validation?	
New Validation?	
Validation Type: (text field, auto-complete list, drop-down list, and so on)	
Validation Definition (list of values or SQL)	

Subworkflow Workflow Step Worksheets

Table A-7. Workflow step [subworkflow], step number ____

Step	Definition
Step Name	
Goal/Result of Step	

Table A-7. Workflow step [subworkflow], step number ____, continued

Step	Definition
Validation*	
Vote on Step's outcome?	
Timeout (Days)	
Source Environment (Group)	
Dest Environment (Group)	
Security (who can act on step): • Security Group • User Name • Standard Token • User Defined Token	
Include Notification (Yes/No)	
Notification Event	
Notification Recipient: • Username • Email Address • Security Group • Standard Token User Defined Token	
Notification Message	
Request Status at Step	
Request % Complete at Step	
Authentication Required (Y/N)	
Authentication Type (if Y)	

Table A-8. Workflow step [subworkflow], step number ____ validation

Validation Information	Value
Existing Validation?	
New Validation?	

Table A-8. Workflow step [subworkflow], step number ____ validation, continued

Validation Information	Value
Validation Type: (text field, auto-complete list, drop-down list, and so on)	
Validation Definition (list of values or SQL)	

Object Type Configuration Sheets

Table A-9. Object type information

Field Name	Value
Object Type Name	
Description	

Table A-10. Object type fields

#	Field Name	Description
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		

Table A-11. Object type commands

Command	Definition
Goal of Commands	
Command Steps	
Conditions (When to run)	

Table A-12. Validation field information

Field Name	Definition
Field Name	
Validation	
Field Behavior	
Attributes (select one):	• Display • Editable • Display Only • Require d
Default Value	
Dependencies	
Clear field when	
Display only when	
Required when	

Table A-13. Field validation information

Validation	Definition
Existing Validation?	
New Validation?	
Validation Type: (text field, auto-complete list, drop-down list, and so on.)	
Validation Definition (list of values or SQL)	

Table A-14. Object type field information

Field Name	Definition
Field Name	
Validation	See "Table A-15. Field Validation information" below.
Field Behavior	
Attributes:	Display Editable Display Only Required
Dependencies	
Clear field when	
Display only when	
Required when	
Default Value	

Table A-15. Field Validation information

Field Validation	Definition
Existing Validation	
New Validation?	
Validation Type (text field, auto-complete list, drop-down list, and so on)	
Validation Definition (list of values or SQL)	

Example of Completed Object Type Configuration Sheets

The following tables are an example of a completed set of object type configuration sheets.

Table A-16. Example of object type information

Field Name	Value
Object Type Name	Dev to Test File Migration
Description	Deployment of a file from Dev server to Test server.

Table A-17. Example of object type fields

#	Field Name	Description
1	File Location	The name of the field. See"Table A-18. Example of validation for File Location" below and "Table A-19. Field validation for File Location" below.
2	Sub Path	The name of the field. User this field to select the directory containing the file to be deployed. See "Table A-20. Example of validation for Sub Path" on the next page and "Table A-21. Field validation for Sub Path" on the next page.
3	File Name	The name of the file. See "Table A-22. Example of validation for File Name" on the next page and "Table A-23. Field validation for File Name" on page 226.

Table A-18. Example of validation for File Location

Field Name	Definition
Field Name	File Location
Validation	See "Table A-19. Field validation for File Location" below.
Field Behavior	
Attributes:	• Display: Yes, the field is visible on the package line. • Editable: Yes, the field can be edited after the package is submitted. • Display Only: Never. This field can be edited. If set to Always, you can never edit this field. • Required: Always. This field must contain a value.
Default Value	None
Dependencies	
Clear field when	None
Display only when	None
Required when	None

Table A-19. Field validation for File Location

Validation	Definition
Existing Validation?	DLV - File Location
New Validation?	Not Applicable
Validation Type (text field, auto-complete list, drop-down list, and so on)	Drop-down list.
Validation Definition (list of values or SQL)	SQL

Table A-20. Example of validation for Sub Path

Field Name	Definition
Field Name	Sub Path
Validation	See "Table A-21. Field validation for Sub Path" below.
Field Behavior	
Attributes:	• Display: Yes, the field is visible on the package line. • Editable: Yes, the field can be edited after the package is submitted. • Display Only: Never. This field can be edited. If set to Always, you can never edit this field. • Required: Always. This field must contain a value.
Default Value	None
Dependencies	
Clear field when	None
Display only when	None
Required when	None

Table A-21. Field validation for Sub Path

Validation	Definition
Existing Validation?	Directory Chooser
New Validation?	Not Applicable
Validation Type (text field, auto-complete list, drop-down list, and so on)	Directory Chooser
Validation Definition (list of values or SQL)	Default behavior

Table A-22. Example of validation for File Name

Field Name	Definition
Field Name	File Name
Validation	See "Table A-23. Field validation for File Name" on the next page.
Field Behavior	

Table A-22. Example of validation for File Name, continued

Field Name	Definition
Attributes:	• Display: Yes, the field is visible on the package line. • Editable: Yes, the field can be edited after the package is submitted. • Display Only: Never. This field can be edited. If set to Always, you can never edit this field. • Required: Always. This field must contain a value.
Default Value	None
Dependencies	
Clear field when	None
Display only when	None
Required when	None

Table A-23. Field validation for File Name

Validation	Definition
Existing Validation?	File Chooser - Full File Name
New Validation?	Not Applicable
Validation Type (text field, auto-complete list, drop-down list, and so on)	File Chooser
Validation Definition (list of values or SQL)	Default Behavior

Send documentation feedback

If you have comments about this document, you can contact the documentation team by email. If an email client is configured on this system, click the link above and an email window opens with the following information in the subject line:

Feedback on Deployment Management Configuration Guide (Project and Portfolio Management Center 9.40)

Just add your feedback to the email and click send.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to your_IE_team_PDL@hpe.com.

We appreciate your feedback!