



Mercury IT Governance Center™
Mercury Change Management™:
Configuring a Deployment System

Version: 6.0



MERCURY™

This manual, and the accompanying software and other documentation, is protected by U.S. and international copyright laws, and may be used only in accordance with the accompanying license agreement. Features of the software, and of other products and services of Mercury Interactive Corporation, may be covered by one or more of the following patents: United States: 5,511,185; 5,657,438; 5,701,139; 5,870,559; 5,958,008; 5,974,572; 6,137,782; 6,138,157; 6,144,962; 6,205,122; 6,237,006; 6,341,310; 6,360,332, 6,449,739; 6,470,383; 6,477,483; 6,549,944; 6,560,564; 6,564,342; 6,587,969; 6,631,408; 6,631,411; 6,633,912; 6,694,288; 6,738,813; 6,738,933; 6,754,701; 6,792,460 and 6,810,494. Australia: 763468 and 762554. Other patents pending. All rights reserved.

Mercury, Mercury Interactive, the Mercury logo, the Mercury Interactive logo, LoadRunner, WinRunner, SiteScope and TestDirector are trademarks of Mercury Interactive Corporation and may be registered in certain jurisdictions. The absence of a trademark from this list does not constitute a waiver of Mercury's intellectual property rights concerning that trademark.

All other company, brand and product names may be trademarks or registered trademarks of their respective holders. Mercury disclaims any responsibility for specifying which marks are owned by which companies or which organizations.

Mercury
379 North Whisman Road
Mountain View, CA 94043
Tel: (650) 603-5200
Toll Free: (800) TEST-911
Customer Support: (877) TEST-HLP
Fax: (650) 603-5300

© 1997–2005 Mercury Interactive Corporation. All rights reserved.

If you have any comments or suggestions regarding this document, please send email to documentation@mercury.com.

List of Figures

Figure 1-1	Mercury IT Governance Center components.....	24
Figure 2-1	Deployment process, high level.....	32
Figure 2-2	Deployment process, detailed level.....	34
Figure 2-3	Process change one	35
Figure 2-4	Migrate DEV to TEST	37
Figure 2-5	Using a subworkflow in your deployment process (example).....	40
Figure 2-6	Using a Ready for Release step in the deployment process.....	42
Figure 2-7	Workflow steps requiring environment specification (red).....	49
Figure 3-1	Workflow components.....	59
Figure 3-2	Step 1. Create a block diagram.....	61
Figure 3-3	Step 2. Create the workflow	62
Figure 3-4	Drag and drop	66
Figure 3-5	Workflow step source	67
Figure 3-6	Workflow step source validation	67
Figure 3-7	AND example	68
Figure 3-8	OR example.....	69
Figure 3-9	SYNC example.....	69
Figure 3-10	FIRST LINE and LAST LINE example.....	70
Figure 3-11	Close workflow step.....	71
Figure 3-12	Step sequence tab	72
Figure 3-13	Workflow tab.....	73
Figure 3-14	Workflow step properties.....	76
Figure 3-15	Transitions using other results	104
Figure 3-16	Transitioning back to the same step.....	105

Figure 3-17	Add a transition based on a previous workflow step	109
Figure 3-18	Transitioning to and from subworkflows	111
Figure 3-19	Workflow step sources and validations	112
Figure 3-20	Workflow step window for environments	115
Figure 3-21	Jump/Receive workflow steps.....	118
Figure 4-1	Information used to create the decision step source.....	135
Figure 4-2	Information used to create the execution step source.....	139
Figure 4-3	Transitioning based on a token.....	151
Figure 4-4	Redirecting the workflow, step 1.....	165
Figure 4-5	Redirecting the workflow, step 2.....	165
Figure 5-1	Example of an object type.....	168
Figure 5-2	Object Type window	169
Figure 5-3	Field window	173
Figure 5-4	Example of an object type and Layout tab.....	184
Figure 6-1	Ready for release step in workflow	201
Figure 6-2	Role of the distribution workflow	202
Figure 6-3	Distribution workflow	203
Figure 6-4	Dependencies and run groups.....	205
Figure 6-5	Release window	210
Figure 6-6	Package added to a release through the package reference.....	216
Figure 6-7	Distribution Status tab	226
Figure 7-1	FTP (server to server).....	232
Figure 7-2	FTP (active).....	232
Figure 7-3	FTP (passive).....	233
Figure 7-4	Worksheet and Environments window.....	234
Figure 7-5	Applications tab.....	243
Figure 8-1	Environment group	260
Figure 9-1	Notifications Template window	272
Figure 10-1	User data types	288
Figure 10-2	User Data window Layout tab.....	306
Figure 10-3	Preview mode.....	309
Figure 10-4	Project and task example	311

Table of Contents

List of Figures	iii
List of Tables	xiii
Chapter 1: Introduction.....	15
About This Document.....	16
Who Should Read This Document	18
Prerequisite Documents	18
Related Documents.....	19
Overview of Mercury Change Management.....	19
Mercury Change Management Concepts.....	20
Overview of Deployment Systems.....	22
Accessing Mercury IT Governance Center.....	24
Creating Deployment Systems	26
Chapter 2: Gathering Process Requirements.....	27
Overview of Gathering Process Requirements	28
Defining Deployment Processes	28
Process (Workflow) Considerations.....	28
Decision Steps.....	29
Execution Steps.....	29
Immediate Versus Manual Executions.....	29
Timeouts	29
Defining Business Flows.....	30
Example: Defining the Business Flow.....	30
Defining Technical Flows	33
Example: Defining the Technical Flow	33

Gathering Information on Steps in the Process	36
Example: Gathering Workflow Step Information	37
Considering Subworkflows.....	39
Example: Using Subworkflows	39
Consider Using Release Management.....	41
Example: ACME Uses Release Management in their Deployment Process.	41
Determine Information to Describe Objects.....	42
Example: ACME collects information on their objects.....	43
Determine Commands Needed for Objects	46
Example: High level Command design	47
Gather Information on Environments.....	47
Example: ACME specifies the Environments.....	48
Identify Participants and Security	50
Example: ACME determines participants and security	51
Establish Communication Points and Visibility	55
Example: ACME configures Notifications.....	56
Chapter 3: Configuring Workflows	57
Overview of Workflows.....	58
Mapping Workflows.....	60
Opening the Workflow Workbench	63
Creating Workflows	64
Configuring General Information for Workflows.....	64
Dragging and Dropping Workflow Steps.....	65
Choosing Workflow Steps.....	66
Overview of Decisions Workflow Steps	68
Overview of Condition Workflow Steps	68
Overview of Execution Workflow Steps.....	70
Overview of Subworkflow Workflow Steps	71
Adding Close Workflow Steps.....	71
Adjusting Workflow Step Sequences	72
Specifying the First Step.....	72
Verifying and Enabling Workflows	73
Configuring Workflow Steps.....	75
Configuring General Information for Workflow Steps	77
Configuring Security for Workflow Steps.....	79
Configuring Dynamic Security for Workflow Steps.....	82
Configuring Notifications for Workflow Steps	84
Configuring Setup Tabs	86
Configuring Message Tabs.....	95
Configuring Timeouts for Workflow Steps.....	98
Configuring Transitions for Workflow Steps	100
Adding Transitions Based on Specific Results	101

Adding Transitions not Based on Specific Results.....	103
Adding Transitions Back to the Same Step	105
Adding Transitions Based on Previous Workflow Step Results.....	108
Adding Transitions To and From Subworkflows.....	110
Configuring Validations for Workflow Steps.....	111
Validations and Execution Type Relationships.....	113
Integrating Object Types and Workflows	114
Integrating Object Type Commands and Workflows	114
Integrating Environments and Workflows.....	115
Choosing Source Environments Based on Application Code	116
Integrating Request and Package Workflows	117
Setting Up WF - Jump/Receive Step Label Validations	119
Generating Jump Step Sources	121
Generating Receive Step Sources.....	123
Including Jump and Receive Workflow Steps in Workflows.....	124
Chapter 4: Configuring Workflow Components.....	127
Overview of Workflow Step Sources	128
Configuring and Using Workflow Step Source Restrictions.....	129
Opening the Workflow Workbench.....	130
Overview of Creating Workflow Step Sources	131
Configuring Ownership of Workflow Step Sources	133
Creating Decision Workflow Step Sources	135
Creating Execution Workflow Step Sources.....	139
Setting Up Execution Steps.....	143
Defining Execution Types	143
Executing Object Type Commands.....	144
Closing Packages as Success.....	145
Closing Packages as Failed	146
Marking Packages Ready for Release	147
Executing PL/SQL Functions and Creating Transitions Based on the Results	147
Executing SQL Statements and Creating Transitions Based on the Results.....	148
Evaluating Tokens and Creating Transitions Based on the Results.....	150
Executing Multiple System Level Commands.....	152
Creating Subworkflow Workflow Step Sources	153
Subworkflows Returning to Change Management Workflows	153
Using Workflow Parameters	155
Creating Workflow Parameters	155
Example: Building a Loop Counter Using Workflow Parameters	157
Modifying Workflows Already In Use.....	161
Performance Considerations when Modifying Security	162
Performance Considerations when Migrating Workflows	162
Copying and Testing Trial Versions of Workflows	163

Modifying Production Workflows.....	164
Disabling Workflow Steps.....	164
Redirecting Workflows.....	164
Moving Requests or Packages Out of Steps.....	165
Chapter 5: Configuring Object Types.....	167
Overview of Object Types.....	168
Opening the Object Type Workbench.....	170
Configuring General Information for Object Types.....	171
Creating Object Type Fields.....	172
Overview of Object Type Field Validations.....	173
Selecting Validations.....	174
Creating Object Type Fields.....	174
Configuring Field Dependencies.....	178
Copying Object Type Fields.....	180
Editing Object Type Fields.....	182
Removing Fields.....	183
Configuring Layouts for Object Types.....	183
Changing Field Widths.....	184
Moving Fields.....	185
Setting Object Names.....	187
Setting Object Revisions.....	188
Configuring Commands for Object Types.....	189
Adding Commands to Object Types.....	189
Editing Commands of Object Types.....	191
Copying Commands in Object Types.....	192
Deleting Commands in Object Types.....	193
Command Conditions.....	194
Configuring Ownership for Object Types.....	195
Adding Ownerships to Object Types.....	195
Deleting Ownerships from Object Types.....	197
Using Commands to Change Field Values.....	198
Chapter 6: Configuring Releases and Distributions.....	199
Overview of Releases and Distributions.....	200
Workflow Scope.....	200
Release Management and Package Workflows.....	200
Release Distribution Workflows.....	201
Package Level Subworkflows.....	202
Dependencies and Run Groups.....	204
Opening Releases.....	206
Submitting Releases.....	206
Overview of Using Release Management - Process.....	206

Release Management Pre-Configuration	206
Creating Releases	207
Processing Releases.....	208
Distributions.....	208
Overview of Configuring Releases	209
Opening the Release Workbench.....	211
Creating Releases	212
Adding Packages to Releases	214
Adding Packages Through the Release Window	214
Adding Packages Through the Package Window	216
Adding Packages by the Ready for Release Workflow Step.....	217
Adding Packages from Requests	219
Adding Requests to Releases.....	219
Adding Requests Through the Release Window.....	219
Adding Requests Through the Requests Window	221
Verifying Releases.....	222
Creating Distributions.....	223
Enabling/Disabling Package Lines in a Distribution.....	225
Running Distributions through a Workflow.....	226
Processing Distribution Steps	226
Processing Package Lines.....	227
Completing Distributions.....	228
Chapter 7: Configuring Environments.....	229
Overview of Environments.....	230
Environment Connection Protocols	230
Environment Transfer Protocols.....	231
Transfer Protocol Configuration Notes.....	231
Selecting the FTP Protocol	231
Overview of Configuring Environments	234
Opening the Environments Workbench	236
Configuring General Information for Environments	237
Creating Environments.....	238
Using Application Codes Environments	243
Copying Application Codes from Other Environments.....	246
Setting Ownership for Environments.....	248
Adding Ownerships to Environments.....	248
Deleting Ownerships from Environments.....	250
Adding Participants to Environments.....	251
Deleting Participants from Environments.....	252

Environment Maintenance and Utilities	253
Testing Environment Setups	253
Mass Updates of Base Paths	255
Environment Password Management Utility	256
Chapter 8: Configuring Environment Groups	259
Overview of Environment Groups	260
Overview of Configuring Environment Groups	261
Opening the Environment Group Workbench	262
Configuring General Information for Environment Groups	263
Creating Environment Groups	264
Setting the Order of Executions	265
Setting Ownership for Environment Groups	266
Adding Ownerships to Environment Groups	266
Deleting Ownerships from Environment Groups	268
Setting Participants for Environment Groups	269
Deleting Participants from Environment Groups	270
Chapter 9: Configuring Notification Templates	271
Overview of Notification Templates	272
Opening the Notification Templates Workbench	273
Deleting Notification Templates	273
Creating Notification Templates	274
Configuring Ownership of Notification Templates	279
Deleting Ownerships from Notification Templates	281
Configuring Notification Intervals	282
Checking the Usage of Notification Templates	285
Chapter 10: Configuring User Data	287
Overview of User Data	288
Referring to User Data	289
Migrating User Data	290
Overview of Configuring User Data	290
Opening the User Data Workbench	291
Configuring General Information for User Data Types	292
Creating User Data Fields	295
Copying a Field's Definition	299
Editing User Data Fields	300
Configuring User Data Field Dependencies	302
Removing Fields	305

Configuring User Data Layouts.....	306
Changing Column Widths.....	306
Moving Fields.....	307
Swapping Positions of Two Fields.....	308
Previewing the Layout.....	309
Configuring Project and Task User Data Roll-Ups.....	310
Example Using Project and Task User Data Roll-Up.....	311
Overview of Configuring User Data Roll-Ups.....	312
Configuring Task User Data for User Data Roll-Ups.....	312
Configuring Project User Data for User Data Roll-Ups.....	314
Configuring User Data Roll-Ups.....	316
Editing User Data Roll-Ups.....	319
Deleting User Data Roll-Ups.....	320
Chapter 11: Rolling Out Your Deployment Process.....	321
General Deployment System Configuration Checklist.....	322
Workflow Checklist.....	323
Object Type Checklist.....	326
Environments Checklist.....	327
Security and User Access Checklist.....	328
Dashboard and Portlet Checklist.....	329
Cross-Entity Checklist.....	330
Migrating Configuration Data into Production.....	331
Enabling Entities and User Access.....	332
Educating Your Users.....	332
Appendix A: Worksheets.....	333
Configuration Workflow Worksheets.....	334
Execution Workflow Step Worksheets.....	335
Decision Workflow Step Worksheets.....	337
Subworkflow Workflow Step Worksheets.....	339
Object Type Configuration Sheets.....	341
Index.....	345

List of Tables

Table 2-1	ACME process workflow step information.....	38
Table 2-2	Java file - object type.....	44
Table 2-3	Environment settings.....	48
Table 2-4	ACME's security groups.....	52
Table 2-5	ACME package creation security.....	53
Table 2-6	ACME package processing security, deployment workflow.....	54
Table 2-7	ACME - security around managing the process.....	54
Table 2-8	Information to gather for workflow step notifications.....	55
Table 2-9	Information to gather for workflow step notifications.....	56
Table 2-10	ACME - Workflow steps with notifications.....	56
Table 3-1	Specific errors for workflow steps.....	89
Table 3-2	Smart URL tokens.....	98
Table 3-3	Workflow transition errors.....	104
Table 3-4	Relationship between validation and execution types.....	113
Table 4-1	Execution window values to execute object type commands.....	144
Table 4-2	Execution window values for closing packages as success.....	145
Table 4-3	Execution window values for closing packages as failed.....	146
Table 4-4	Execution window values for marking packages as Ready for Release.....	147
Table 4-5	Execution window values for executing PL/SQL functions.....	148
Table 4-6	Execution window values for executing SQL statements.....	148
Table 4-7	Execution window values for evaluating tokens.....	150
Table 4-8	Example of execution window values for evaluating tokens.....	151

List of Tables

Table 4-9	Execution window values for returning from subworkflows	154
Table 4-10	Rules for modifying production workflows	161
Table 5-1	Field dependencies	178
Table 5-2	Example conditions	194
Table 10-1	Field dependencies	302
Table 11-1	General configuration checklist.....	322
Table 11-2	Workflow configuration checklist	323
Table 11-3	Workflow logical guidelines	324
Table 11-4	Object type configuration checklist	326
Table 11-5	Environment definition checklist.....	327
Table 11-6	Security/user access configuration checklist.....	328
Table 11-7	Dashboard/portlet configuration checklist	329
Table 11-8	Cross entity configuration checklist.....	330
Table A-1	Workflow skeleton	334
Table A-2	Workflow step [execution], step number ____	335
Table A-3	Workflow step [execution], step number ____ validation	336
Table A-4	Workflow step [execution], step number ____ execution type.....	336
Table A-5	Workflow step [decision], step number ____	337
Table A-6	Workflow step [decision], step number ____ validation	338
Table A-7	Workflow step [subworkflow], step number ____	339
Table A-8	Workflow step [subworkflow], step number ____ validation	340
Table A-9	Object type information.....	341
Table A-10	Object type fields	341
Table A-11	Object type commands.....	342
Table A-12	Object type field information	343
Table A-13	Field validation information	343
Table A-14	Object type field information	344
Table A-15	Field validation information	344

Chapter 1 Introduction

In This Chapter:

- *About This Document*
 - *Who Should Read This Document*
 - *Prerequisite Documents*
 - *Related Documents*
 - *Overview of Mercury Change Management*
 - *Mercury Change Management Concepts*
 - *Overview of Deployment Systems*
 - *Accessing Mercury IT Governance Center*
 - *Creating Deployment Systems*
-

About This Document

Mercury Change Management™ is a Mercury IT Governance Center™ product that automates the migrations and deployment of software code, configurations, and content. These objects are grouped into packages and are routed along business processes modeled in Mercury's configurable workflow. Creating workflows to follow your business processes requires a variety of Mercury IT Governance entities configured to work together. This document details those entities and how they can be configured to support your business processes.

This document contains the following chapters:

- [Introduction on page 15](#)

This chapter describes the document and provides an overview of the configuration process.

- [Gathering Process Requirements on page 27](#)

This chapter discusses the information that needs to be collected before developing a deployment system.

- [Configuring Workflows on page 57](#)

This chapter discusses how to create workflows. Workflows represent business processes. Workflows allow you to map business rules and processes to your organization. Information discussed in this chapter includes the following:

- Demand Management workflows
- Change Management workflows
- Release Management workflows

- [Configuring Workflow Components on page 127](#)

This chapter discusses how to build workflow components. Included in this discussion is how to create execution workflow steps, decision workflow steps and subworkflow workflow steps. Information discussed in this chapter includes the following:

- Demand Management workflows
- Change Management workflows
- Release Management workflows

- [Configuring Object Types on page 167](#)

This chapter discusses how to build object types. Object types are used to define the technical steps required to deploy a particular object, such as a package moving from one instance to another instance.

- [Configuring Releases and Distributions on page 199](#)

This chapter discusses how to build releases and distributions. A release is a group of packages and related requests that need to be deployed together. Distributions are the deployment of releases. In a distribution, the release manager specifies which workflow will control the release process and which of the release's packages will be included.

- [Configuring Environments on page 229](#)

This chapter discusses how to build an environment. Where object types are used to define the technical steps required to deploy a particular object, such as a package moving from one instance to another instance, environments define the instances available for a deployment.

- [Configuring Environment Groups on page 259](#)

This chapter discusses how to build environment groups. Environment groups define a set of environments which can be referenced as the source or destination for deployments.

- [Configuring Notification Templates on page 271](#)

This chapter discusses how to build notification templates. Notification templates are pre-configured notifications that can be used to quickly construct the body of a message.

- [Configuring User Data on page 287](#)

This chapter discusses how to build user data fields. Product entities such as packages, workflows, requests and projects include a set of standard fields that provide information. User data fields are additional fields you can configure to accompany those product entities.

- [Rolling Out Your Deployment Process on page 321](#)

This chapter provides things to consider and checklist when you are ready to rollout a deployment system.

- [Worksheets on page 333](#)

This appendix provides a series of worksheets to help gather information required to build a workflow.

Who Should Read This Document

This document is for the following audience types:

- Application developers and configurators

For More Information

For information about audience types, see *Guide to Documentation*.

Prerequisite Documents

Prerequisite documents for this document are:

- *Guide to Documentation*
- *Key Concepts*
- *Getting Started*

For More Information

For information about these documents and how to access them, see *Guide to Documentation*.

Related Documents

Related documents for this document are:

- *Commands, Tokens, and Validations Guide and Reference*
- *Open Interface Guide and Reference*
- *Reports Guide and Reference*
- *Security Model Guide and Reference*

For More Information

For information about these documents and how to access them, see *Guide to Documentation*.

Overview of Mercury Change Management

Mercury Change Management enables you to plan, package, release, and deploy changes to your applications portfolio. It digitizes best practice software change management processes across platforms and environments (such as mainframe, UNIX, NT, and Linux), types of change (code, configurations, content), or applications (such as Oracle, PeopleSoft, SAP, and Siebel). By automating formerly manual tasks, Change Management lets you accomplish more with less and dramatically lessen the risk of “broken” deployments.

With Mercury Change Management, you can:

- Automate migrations and deployments of software changes across your system landscape, from development to test, staging, and production.
- Digitize your change management best practices and methodologies to uniformly plan, deploy, and manage changes.
- Hide the complexity of point tools such as version control and testing, while leveraging their functionality.
- Pinpoint problems quickly and roll back changes if necessary.

Mercury Change Management Concepts

The following lists and defines the major concepts of Mercury Change Management.

- **Deployment.** Mercury Change Management can be configured to automate deployment processes. Deployment is the act of moving an object (such as a file, script, code, or full application) 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 from one machine to another, moving files, and running required scripts or compilers.
- **Environments.** To automate the migration of file system objects, Mercury Change Management must have knowledge of the sources and destinations for the various objects. This data is stored in environments, which are then referenced through workflows and object types.

An environment consists of a server, a single database instance, and an associated remote client machine. Not all of these components need be present in a single environment. For example, it is possible to have an environment which does not contain a database.

When migrating objects, Mercury Change Management connects to remote computers in the same way as any other user (using FTP, SCP, SSH, or Telnet). Change Management can logon using any existing username and password. However, it is recommended that a new user be generated on each computer that Mercury Change Management will access. This will help clarify the setup and relieve some administrative burden. The Mercury ITG user should have full access to the *ITG_Home* directory as well as the correct read and write permissions on other required directories. In addition, on Windows NT computers, the Administrators group must have read access to Change Management's home directory.

- **Environment Groups.** Situations may arise where it is desirable to execute 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 in one environment group.

Environment groups define a set of Mercury Change Management environments which can be referenced as the source or destinations for object migrations and executions.

- **Object Types.** Mercury Change Management automates complex software deployment processes. While Mercury IT Governance 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 a SQL Script object type might address the migration and execution of database scripts.

Object types are used by users who create and process packages. Each package line in a package consists of one object of a specific object type. When defining a package line, the user will select an object type in the Add Line window in the package screen. Fields dynamically appear that are required to process that type of object.

- **Package.** Mercury Change Management gathers all information required for a successful deployment (such as information on environments and objects to be migrated) into a single logical unit called the package. The package, consisting of the migrating objects, is then processed through a business workflow. This results in a successful, easy-to-track software or application change.

Each object in a package is defined in a separate package line. While each line can be acted upon separately, the group of package lines (objects) represent a logical unit that should be moved and tracked together. The processing of a package and package lines can vary greatly depending upon the workflow specified for that package.

- **Package Lines.** Packages can be used to deploy multiple objects. Each object is specified on a separate package line. It is possible to configure your workflow to process different types of objects along different processes. For example, you might want your Java code to undergo more approvals than a basic readme file. The processing of a package and package lines can vary greatly depending on the workflow specified for that package.
- **Release and Release Distribution.** Release Management introduces repeatable, reliable processes surrounding software and application releases. Mercury Change 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.

Using a release, you can:

- Group related packages and requests in a single window
- Provide visibility into related package statuses
- Set dependencies between packages
- Define how a release is distributed to different environments
- **Workflow.** A workflow consists of a logical series of steps that define the path followed by objects (package lines) in a package. It is possible to create custom workflows to model virtually any business process. This allows a department to generate workflows to automate existing processes, rather than forcing them to adopt a new set of processes to perform their work.

Workflow steps can range in usage from functional approvals to actual migrations. For example, you can create a migration step to automatically move specified objects from the source environment to the destination environments.

Overview of Deployment Systems

This document provides the building blocks required to support Mercury Change Management. The integrated workflow engine enables you to digitize both simple and complex process requirements at all levels of your IT operations. This support ranges from high-level collaboration requirements, such as an executive review of your IT portfolio, to detailed automation, such as deploying code to an application server. Consider the case of the process illustrated in [Figure 1-1 on page 24](#). This simplified workflow process includes the following workflow steps:

- **Release Patch.** In this decision workflow step, a Mercury IT Governance Center user will approve or reject the patch release. The configuration elements required for this step include the following:
 - A decision workflow step is configured with security groups, notifications, timeouts, and transitions. Only members of the specified security groups can approve or reject this patch. A notification is sent out to the members of the specified security groups. If no one approves or rejects the request within an allotted time, another notification is sent.
 - A notification template configured to be used by the decision workflow step.

- **Test to PreProduction.** In this execution workflow step, a patch has been submitted for distribution. The patch consists of a configured package. A Mercury IT Governance Center user will migrate the patch (package) from the Test environment to the PreProduction environment. The configuration elements required for this step include the following:
 - An execution workflow step is configured with security groups, notifications, environments, timeouts, and transitions. Only members of the specified security groups can migrate the package from one environment to another environment. A notification is sent out to the members of the specified security groups. If no one migrates the patch within an allotted time, another notification is sent.
 - An object type is configured to migrate the package from one environment to another environment. The package includes a user data field to track user satisfaction.
 - A notification template configured to be used by the execution workflow step.
 - A user data field is created to track user satisfaction. This user data field is configured as part of all packages.
- **Verify Patch.** In this decision workflow step, a Mercury IT Governance Center user will verify the patch migration was successful. The configuration elements required for this step include the following:
 - A decision workflow step is configured with security groups, notifications, timeouts, and transitions. Only members of the specified security groups can verify the patch. A notification is sent out to the members of the specified security groups. If no one approves or rejects the request within an allotted time, another notification is sent.
 - A notification template configured to be used by the decision workflow step.
- **Close (Success).** In this execution workflow step, the package is automatically closed.
 - An execution workflow step configured with security groups and transitions. The workflow step is configured to automatically close the package.

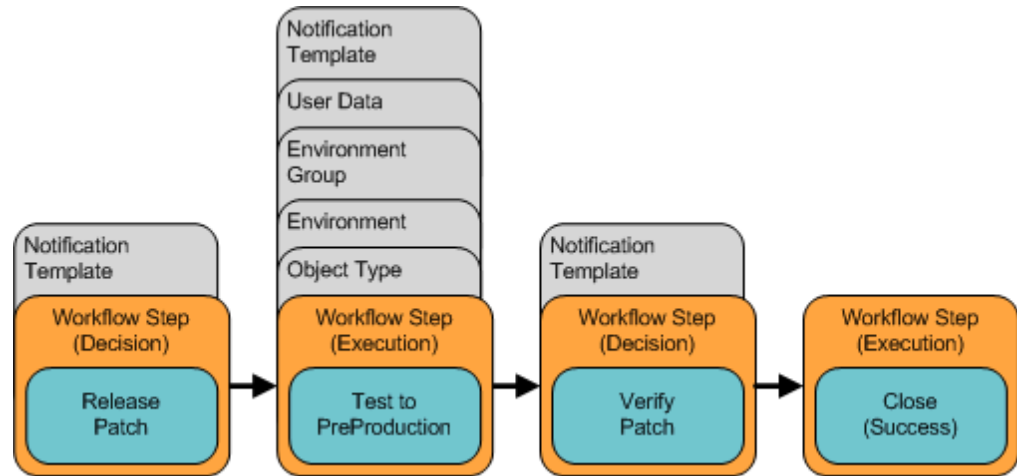


Figure 1-1. Mercury IT Governance Center components

Accessing Mercury IT Governance Center

Businesses often need to control access to certain information and business processes. This can be done to protect sensitive information, such as employee salaries, or to simplify business processes by hiding data that is irrelevant to the user. Mercury IT Governance Center includes a set of features to help control data and process security on the following levels:

- Limiting who can access certain windows or pages
- Limiting who can view or edit certain fields
- Limiting the data displayed in sensitive fields or screens
- Limiting which users can view, create, edit or process Mercury IT Governance Center entities, such as requests, packages, projects, portfolios, and programs
- Limiting which users can view, create or edit Mercury IT Governance Center configuration entities, such as workflows, request types, object types, and security groups
- Limiting which users can alter the security settings

The following features control the data and process security in Mercury IT Governance Center. These features can be combined in a number of ways to provide a secure system:

- **Licenses.** Each user is assigned a license that provides the user with the potential to access to a set of Mercury IT Governance Center product-related screens and functions. Licenses dictate available behavior but need to be used in conjunction with access grants to enable specific fields and functions.
- **Access Grants.** Linked to users through security groups, access grants define which windows and functions users can view, edit, or perform actions in. Access grants also provide varying levels of control over certain entities and fields.
- **Entity-level restrictions.** Settings on the entity that specify who can create, edit, process, and delete Mercury IT Governance Center entities, such as requests, packages, and projects. You can also control which request types and object types can be used with certain workflows. These restrictions are often configured in the configuration entities, such as workflows, request types, and object types.
- **Field-level restrictions.** For each custom field that you define in Mercury IT Governance Center, you can configure when it is visible or editable. For some fields, you can additionally specify which users can view or edit the field.
- **Configuration-level restrictions.** You can specify, using ownership groups settings, which users can modify configuration entities in the system. For example, you can control who is allowed to edit an existing workflow. This allows you to guarantee that only appropriate users are altering your Mercury IT Governance Center-controlled processes.

For More Information

For more information concerning accessing the Mercury IT Governance Center, security groups, and access grants, see *Security Model Guide and Reference*.

Creating Deployment Systems

To configure a deployment system or process:

1. *Gathering Process Requirements*

Before configuring Mercury Change Management to manage deployment processes, collect specific related information. This includes gathering information on the business process, technical process, the types of objects that will be deployed, source and destination environments, participants who will create and process packages, and the communication devices surrounding the process.

2. *Configuring Workflows*

Using the information gathered in the *Gathering Process Requirements* chapter, build the workflow. This includes setting up required workflow step sources, adding notifications, adding security groups, and adding steps and transitions to your workflow. Also see, *Configuring Workflow Components* and *Configuring Releases and Distributions*.

3. *Configuring Object Types*

Using the information gathered in the *Gathering Process Requirements* chapter, build the object type(s). This includes creating and configuring object type fields and adding commands to the object type.

4. *Configuring Environments*

Define all environments involved in the deployment and distribution process. This includes setting up environments and environment groups and then adding them to the workflow definition. Also see, *Configuring Environment Groups*.

5. *Rolling Out Your Deployment Process*

It is recommended that a formal change management process be followed when configuring Mercury Change Management. This includes testing the configurations, migrating them into the production instance, enabling the processes, and training the user base.

Gathering Process Requirements

In This Chapter:

- *Overview of Gathering Process Requirements*
 - *Defining Deployment Processes*
 - *Process (Workflow) Considerations*
 - *Defining Business Flows*
 - *Defining Technical Flows*
 - *Gathering Information on Steps in the Process*
 - *Considering Subworkflows*
 - *Consider Using Release Management*
 - *Determine Information to Describe Objects*
 - *Determine Commands Needed for Objects*
 - *Gather Information on Environments*
 - *Identify Participants and Security*
 - *Establish Communication Points and Visibility*
-

Overview of Gathering Process Requirements

This chapter discusses the information that needs to be collected before developing a deployment system. This includes the following business and technical information:

- **Business process.** What are the steps in the process; which steps need to be reviewed and approved?
- **Technical process.** Which steps require objects to be deployed and scripts to be run?
- **Types of objects that will be deployed.** Will the same process be used to deploy different type of objects (such as files, data, and scripts)?
- **Source and destination environments.**
- **Participants who will create and process packages.** Determine the level of security to place on this system.
- **Communication devices surrounding the process.** Do you want to communicate using notifications, the Mercury IT Governance Dashboard, or reports.

Defining Deployment Processes

The first step to configuring a deployment process is to define the process—the actual steps required to deploy an object. This includes process information such as when to obtain reviews and approvals on the object to be deployed, when to deploy objects, and the path (transitions) between steps in the process.

Process (Workflow) Considerations

The Mercury Change Management workflow includes a number of features to consider when defining a process. The following section describes a few of the notable features. For a more comprehensive discussion of workflow features and configuration techniques, see [Configuring Workflows on page 57](#).

Decision Steps

Decisions are workflow steps that require an external process to decide their outcome. Typically, this is an individual logged into the system (such as a QA Manager approving a bug fix). Decisions are used for a wide variety of purposes within a workflow. They may be used to gather approvals before code is migrated, or they may be used to request additional information on a request (within Mercury Demand Management).

Execution Steps

Executions are workflow steps that perform actual work. This work can consist of activities like object migrations, the creation of a new package or the completion of a request. Executions can be immediate, manual or scheduled. Technical processing of the object in an environment occurs at an execution step.

Immediate Versus Manual Executions

Execution timing is often very important in a deployment process. Mercury Change Management includes functionality to control when certain execution steps are run. You can configure your process with an **Immediate** execution step that will execute the step at the instant that the step becomes eligible (package enters the step). You can also configure your process so that you need to manually execute a step.

Timeouts

If a process step is in a specific state for a predetermined period of time, the process can “timeout.” The process can then process based on the timeout event to avoid potential bottlenecks.

Defining Business Flows

Map the business process. This consists of identifying all steps (decisions, conditions, and executions) and transitions needed to deploy changes. It is helpful to graphically map these processes by:

- Identifying all decision points in the process
- Determining a flow between steps (transitions). You should consider all possible exit values from each step (such as approved, not approved, rework, and error)
- Identifying process closure points (success or failure)

The following example provides an illustration of the design issues that should be consider.

Example: Defining the Business Flow

ACME Company needs to configure a deployment process for changes to their financial applications system. This system consists of over ten modules including billing, accounts payable, accounts receivable, fixed asset management, inventory, reporting, payroll, and cash management. Deployment to the different modules can require unique processing items (such as destinations, environment management, and post deployment processing steps). ACME's IT group needs to create a process that can address the complications related to deploying changes to this system.

Additionally, they need to address the following requirements:

- The TESTING environment must use the code and data that is housed in the version control system.
- Only certain users can approve and migrate changes.
- The PRODUCTION update must occur between 1:00 and 2:00 a.m. on Friday. Any additional system downtime could result in financial loss.

Example: ACME defines a high-level process flow

ACME first creates a high-level business process. The process begins after the software engineers and IT staff develop changes to the Financial Applications module, and check their code into their version control system. ACME's deployment process begins with the Migrate DEV to TEST Step.

1. **Migrate DEV to TEST.** Migrate the changes from the development area (DEV) to the testing area (TEST). This includes checking the code out from the version control system into the DEV area, transferring the files, and then compiling the code on the TEST instance.
2. **Validate changes in TEST.** Validate the changes in TEST using standard Quality Assurance processes. Any required rework is routed back to the appropriate engineering staff.
3. **Migrate TEST to PROD.** Migrate the changes from TEST to the Production area (PROD). This, again, involves checking out files from the version control system, transferring the files and compiling the code.

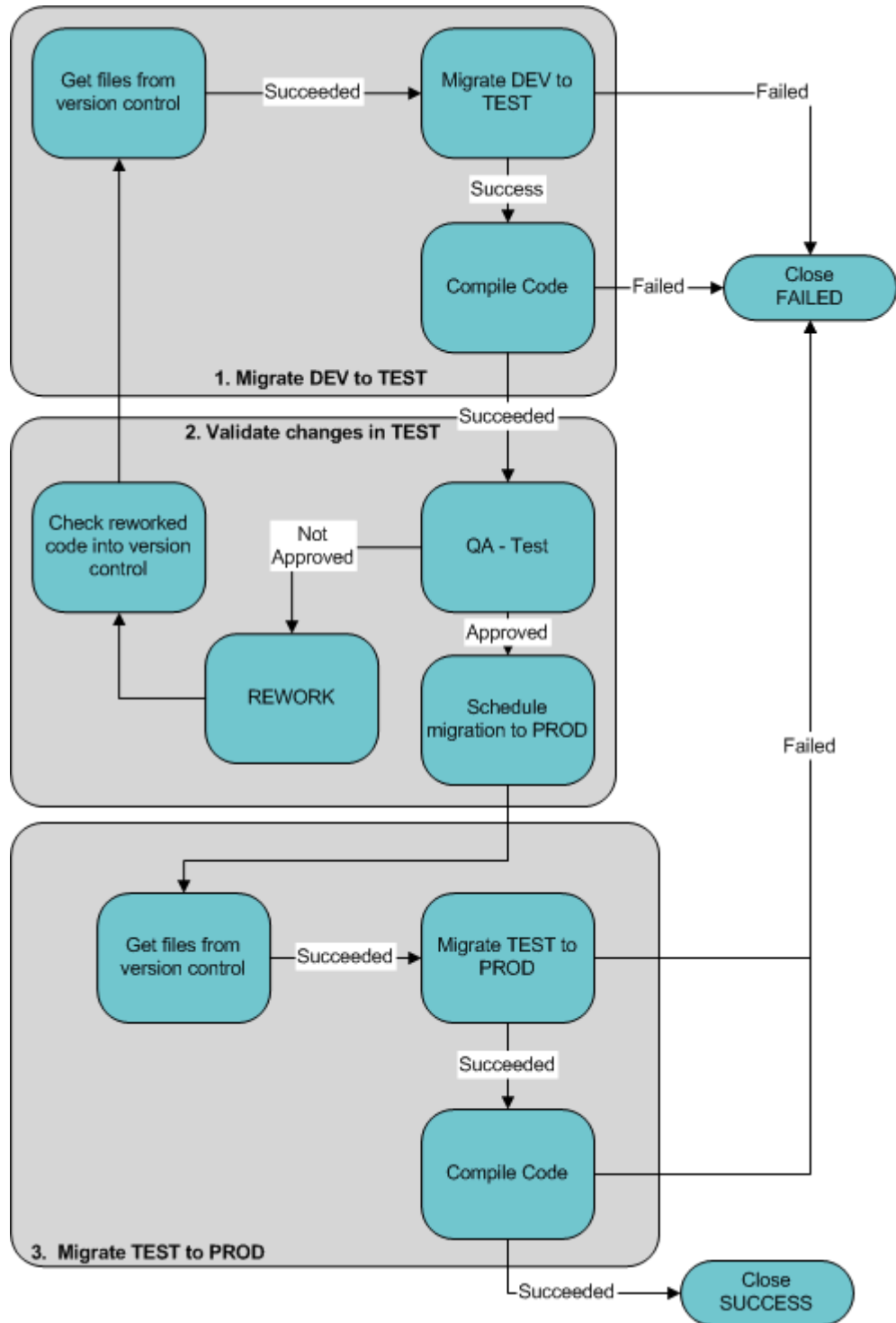


Figure 2-1. Deployment process, high level

Defining Technical Flows

Once the high-level business process is defined, overlay a technical process by identifying:

- Types of objects that you will be deploying
- Dependencies between objects, environments, and workflow steps
- Any required system level commands
- Where to use condition steps (AND, SYNC, OR, FIRST LINE, LAST LINE)

Example: Defining the Technical Flow

After investigating some more technical requirements of their system, ACME identified the following information:

Types of objects to be deployed:

- HTML files
- Java files
- Database changes: changes to the schema as well as additional system data for existing tables

Object-related dependencies:

- The server and database are located on different machines. Therefore, database-related objects need to be deployed to a different machine than the other objects (files).

Additional process requirements:

- The server must be stopped before the migration.
- The code can not be compiled until all objects have been migrated.
- Following a FAILED execution step, it should be able to reset the execution.

Example: Detailed Process

ACME updated their process to address the additional information, resulting in the following process.

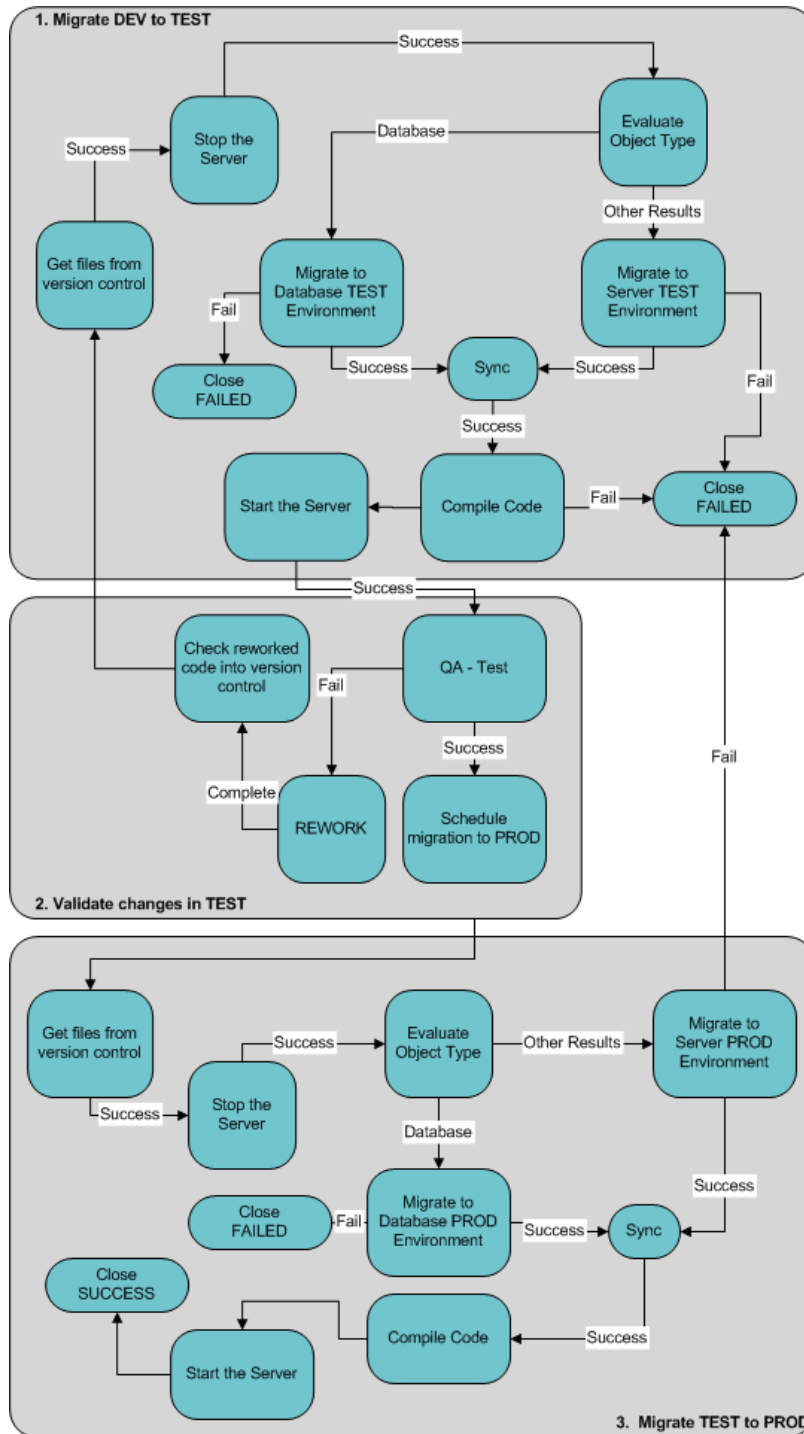


Figure 2-2. Deployment process, detailed level

To address the technical requirements of their deployment process, ACME introduced the following changes to their workflow:

Process change one:

- **Requirement.** The server and database are located on different machines. Therefore, database-related objects need to be deployed to a different machine than the other objects (files).
- **Result.** To address this requirement, ACME added the evaluate object type step. If the object being routed through the deployment process is a database-related object, then it is deployed to the database environment. All other objects are routed to the server environment.

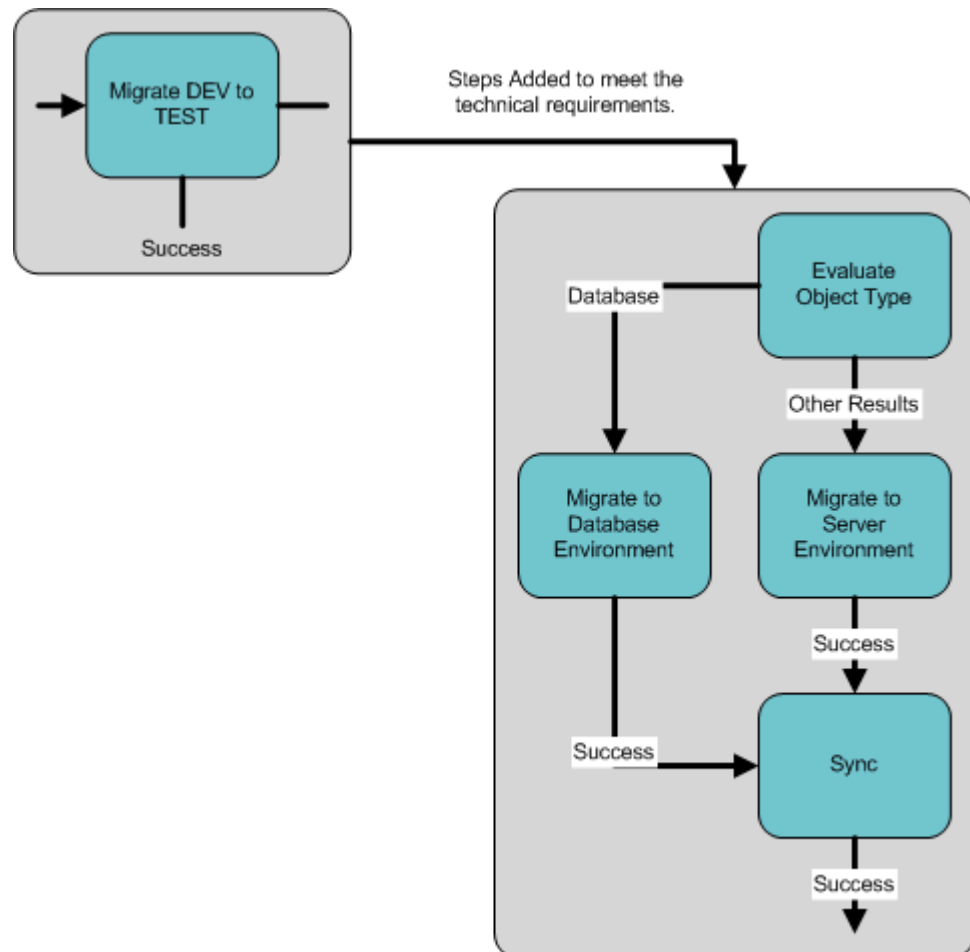


Figure 2-3. Process change one

Process change two:

- **Requirement.** The server must be stopped before the migration
- **Result.** To address this requirement, ACME added a Stop the Server step before the files are transferred. This step appears during both DEV to TEST and TEST to PROD migrations.

Process change three:

- **Requirement.** The code can not be compiled until all objects have been migrated.
- **Result.** To address this requirement, ACME introduced a Sync step following the migration steps. This ensures that all objects (Package Lines) reach a specific point before the Package can continue along its process. It also ensures that the branched objects (database versus others) are reunited in the process.

Gathering Information on Steps in the Process

After gathering the business and technical process steps of the deployment process into a single Workflow, gather detailed information on each step and transition in the process. This section discusses the information that needs to be collected. [Worksheets on page 333](#) includes a worksheet to help you collect the required information.

For each step in the process, collect the following information:

- **Step name.**
- **Description.** Describe the goal of the step. This is especially helpful for Execution Steps.
- **Step Type.** Decision, execution, condition, or subworkflow.
 - **Decision Step Specific Information.** Such as number of approvals required and timeouts.
 - **Execution Step Specific Information.**
 - **Desired Results of the Execution.** This will help to choose the execution type and build any required commands.
 - **Execution Timing.** Determine whether the execution should occur immediately or be processed manually.
 - **Subworkflow Step Specific Information.**

- Transition Values and Validation.** Transition values are the possible results for the step. Depending on the result, the process will proceed in different directions. Use one of Mercury Change Management's system validations or create a custom validation.

Example: Gathering Workflow Step Information

ACME begins capturing the step information for the migrate DEV to TEST portion of their process.

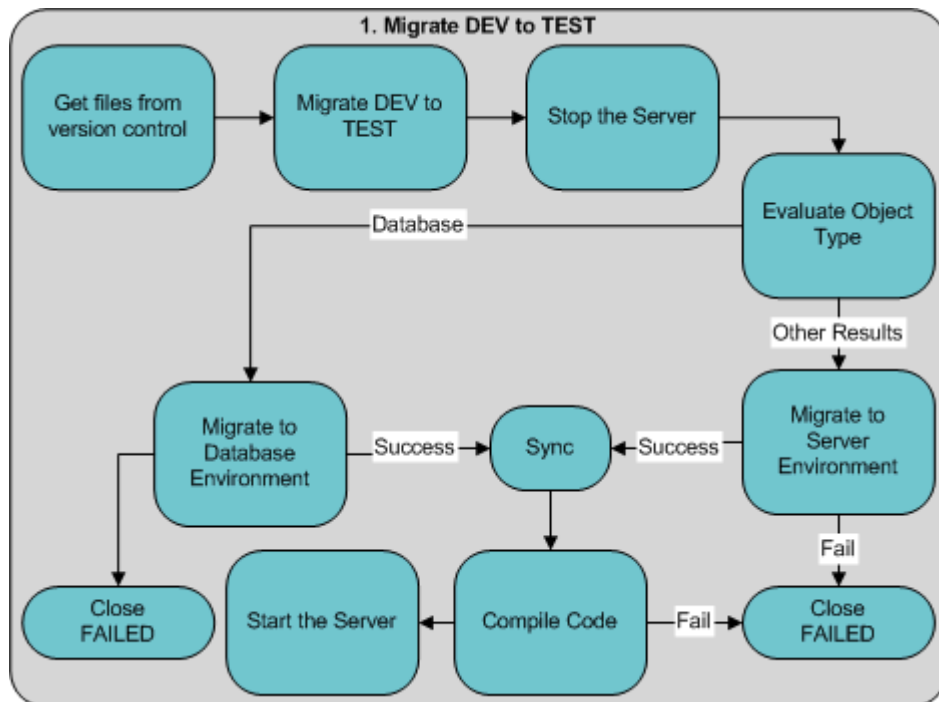


Figure 2-4. Migrate DEV to TEST

Table 2-1. ACME process workflow step information

Step Name	Type	Transition Values	Validation	Description
Get files from version control	Execution	Succeeded Failed	WF - Standard Execution Results	Connect to the version control system and check out files into the DEV instance.
Migrate DEV to TEST	Decision	Approved Not Approved	WF - Approval Step	Decide whether to begin the migration process.
Stop the Server	Execution	Succeeded Failed	WF - Standard Execution Results	Connect to the server and stop the processes running on it.
Evaluate Object Type	Execution	Database File SQL Script	Custom Validation defined at site.	Evaluate the object type for each package line. Resolve the object type token.
Migrate to Database Environment	Execution	Succeeded Failed	WF - Standard Execution Results	Migrate the database changes to the TEST database environment. To do this, execute commands located in the object type.
Migrate to Server Environment	Execution	Succeeded Failed	WF - Standard Execution Results	Migrate the changes to the TEST server environment. To do this, execute commands located in the object type.
Sync	Condition	Success	WF - Standard Condition Results	Have all package lines enter this step before any continue to the next process step.
Compile Code	Execution	Succeeded Failed	WF - Standard Execution Results	Connect to the server and compile the code located on it.
Start the Server	Execution	Succeeded Failed	WF - Standard Execution Results	Connect to the server and start the processes on it.
Close FAILED	Execution	Succeeded	WF - Standard Execution Results	When a package line (object) enters this step, close the package.

Considering Subworkflows

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

Workflows can be used as subworkflows within a parent workflow. An entire subworkflow is represented by a single icon in the parent workflow window's **Layout** tab. This simplifies the potentially complex graphical layout and enables the easy reuse of common workflow configurations.



Note

Subworkflows will appear to the user processing the package as expandable/collapsible sections in the Package Status panel. See *Processing Packages (Change Management)* for examples.

Example: Using Subworkflows

ACME decides to use a subworkflow for the object migration portion of their process. This subworkflow can be referenced in two parts of the process.

[Figure 2-5 on page 40](#) shows where ACME could implement a subworkflow.

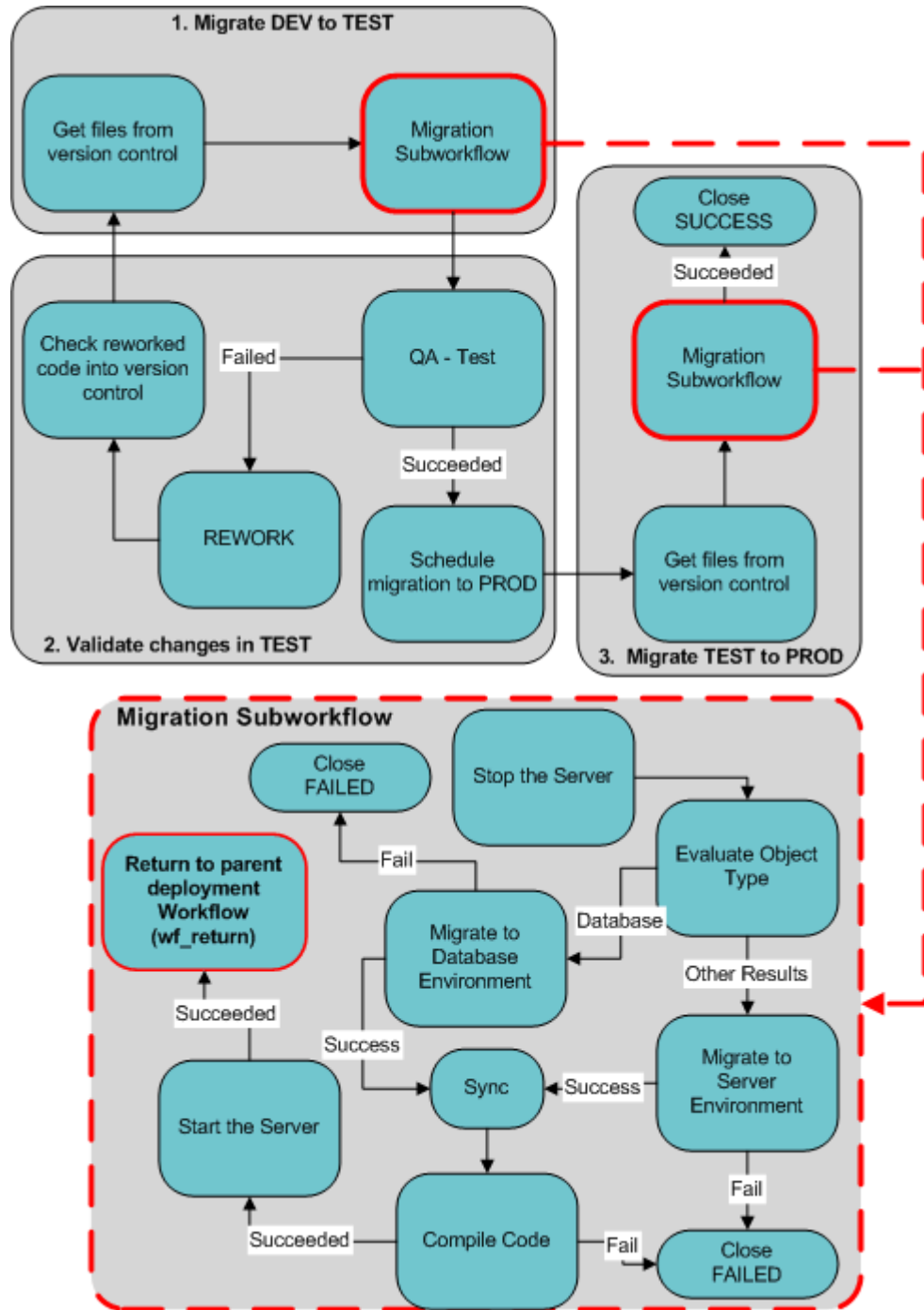


Figure 2-5. Using a subworkflow in your deployment process (example)

Consider Using Release Management

Release Management introduces repeatable, reliable processes surrounding software and application releases. Mercury Change 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.

Using Release Management functionality, Release Managers can:

- Group related packages and requests in a single window
- Provide visibility into related package statuses
- Set dependencies between packages
- Define how a release is distributed to different environments

This consolidation of common Release Management activities provides a powerful tool for creating repeatable and reliable releases.

It is possible to configure a deployment process to feed packages into a release. A Ready for Release step can be included in the workflow. When a package line enters the Ready for Release step, the developer (or other Mercury IT Governance 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 of the package lines are confirmed in the Ready for Release step, the package is ready to be used in the release.

Example: ACME Uses Release Management in their Deployment Process.

ACME decides to create a process that feeds the tested packages into a release for final distribution. Their resulting process then substitutes the final migration to PROD with a Ready for Release step. The distribution workflow defined for the Release Management process then handles this final migration.

When the final distribution to PROD occurs, the Release Management process communicates with this deployment process and the package will close.

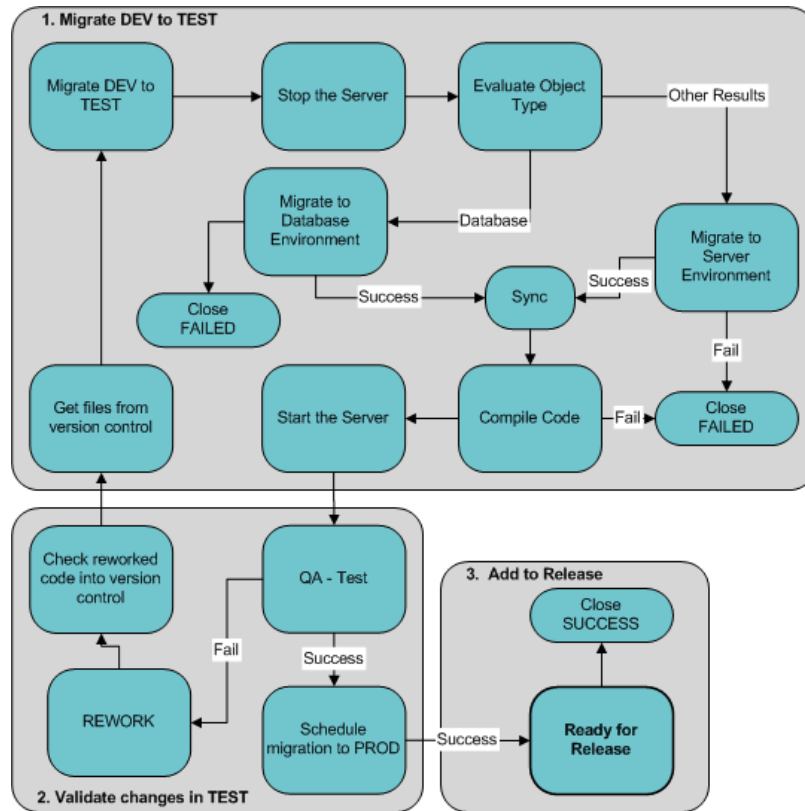


Figure 2-6. Using a Ready for Release step in the deployment process

Determine Information to Describe Objects

Many types of objects are deployed through a workflow such as files, SQL scripts, and data. Each object requires different information to process it. These are defined in the object type fields. For example, to migrate a file, the name, file type, and location of the file must be known. Additional information such as whether or not the file will be compiled after migration can also be specified on the field level (as well as in the commands).

Different object types can be processed through a single Workflow. Information contained on the package line (which is defined in the object type) works in conjunction with the workflow process to ensure that the object is correctly processed. For example, deploying an HTML file requires different processing than deploying a Java file. Therefore, it is likely that at least one

field on the object type will specify the type of object being processed. When the workflow deploys an HTML file, it will consider this field when routing or processing this object.

For each type of object that will be deployed through the process, collect the following information:

- The name of the object
- Object category (optional, used for reporting purposes)
- Parameters that describe the object: what it is, where it is, its name, what needs to be done to it, and so on. This information will translate into object type fields. For each parameter (field), define the following:
 - Field name
 - Validation and component type (dictated by the validation)
 - Field behavior: whether it is displayed, required, any defaulting behavior, and so on.
- Commands contained in the object. Object type commands often reference information stored in the parameters. These commands are executed at specific points (execution steps) in the workflow. For additional information, see [Determine Commands Needed for Objects on page 46](#).

For assistance in collecting the correct data, use the [Worksheets on page 333](#).



Successfully defining object parameters and commands is essential to an effective deployment process. To provide additional guidance on creating object types for specific types of objects, a number of examples are provided in this document. For some sample object types, fields and commands, see [Configuring Object Types on page 167](#).

Example: ACME collects information on their objects

ACME needs to be able to deploy the following types of objects:

- HTML Files
- Java Files
- Database changes, changes to the schema as well as additional system data for existing tables

The following worksheet includes data for the File object type. To describe the file object, ACME decided the following fields need to be defined:

- File Location: whether the file located on the environment’s client or server.
- Sub path: the directory where the file is located.
- File Name: the name of the specific file.
- File Type: the type of file (Java or HTML)

ACME decided on the appropriate validations and field behavior and completed the worksheet shown in the following table (*Table 2-2*).

Table 2-2. Java file - object type

	Value	Description/Notes
Object Type Name	File	
Description		
Field 1		
Field Prompt	File Location	The name of the field.
Validation		
Existing Validation?	DLV - File Location	Specifies if the file is located on the environment's client or server.
New Validation?	Not Applicable	Not Applicable
Validation type	Drop-down List	This field is a drop-down list.
Validation definition	SQL	This validation is defined by SQL.
Field Behavior		
Attributes		
Display	Yes	This field is visible on the package line.
Editable	Yes	This field can be edited even 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	No defaults are set for this field.
Dependencies		

Table 2-2. Java file - object type [continued]

	Value	Description/Notes
Clear When	None	No dependencies are set for this field.
Display Only When	None	No dependencies are set for this field.
Required When	None	No dependencies are set for this field.
Field 2		
Field Prompt	Sub Path	The name of the field. Use this field to select the directory containing the file to be migrated.
Validation		
Existing Validation?	Directory Chooser	Specifies if the file is located on the environment's client or server.
New Validation?	Not Applicable	Not Applicable
Validation type	Directory Chooser	
Validation definition	Default behavior	
Field Behavior		
Attributes		
Display	Yes	This field is visible on the package line.
Editable	Yes	This field can be edited even 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	No defaults are set for this field.
Dependencies	None	No dependencies are set for this field.
Field 3		
Field Prompt	File Name	The name of the field. In this field, users can select a file to deploy.
Validation		
Existing Validation?	File Chooser - Full File Name	Validation that allows you to select a file on the client or server.
New Validation?	Not Applicable	Not Applicable
Validation type	File Chooser	
Validation definition	Default Behavior	

Table 2-2. Java file - object type [continued]

	Value	Description/Notes
Field Behavior		
Attributes		
Display	Yes	This field is visible on the package line.
Editable	Yes	This field can be edited even 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	No defaults are set for this field.
Dependencies	None	No dependencies are set for this field.

Determine Commands Needed for Objects

When defining a deployment process, consider what commands need to be executed to achieve the desired results. Commands control which steps must be executed for each workflow step to complete successfully. This can involve such things as migrating a file, executing a script, or compiling some 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 executed. It is then possible to use this information to build your commands adhering to Mercury Change Management's command standards, or to deliver these as design specifications for an engineer in your group.

Collect the following information for each object type that are designed:

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

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

Example: High level Command design

To deploy the Java and HTML files, ACME must include commands in the File object type. As part of their design, they determine that the following command steps must be executed:

Goal of command:

To copy the file from the source environment to the destination environment.

Steps to achieve goal:

1. Check to see if the source file is located on the client or the server.
2. Connect the destination environment to the source (client or server).
3. Check to see if the directory exists in the destination. If one does not exist, create the directory.
4. Copy the file from the source to the destination.

When to run the commands:

These commands should be run during the Migrate to Server Environment workflow step.

Gather Information on Environments

Some execution type workflow steps require environment information to complete their executions. For a deployment process, collect the following Environment requirements for each workflow execution step:

- Execution step name
- Source environment (or environment group)
- Destination environment (or environment group)

This information can be collected using the workflow step worksheet in [Worksheets on page 333](#).

Each environment must be carefully configured to ensure that passwords, communication protocols, and transfer protocols are configured properly. For instructions on configuring your environments, see [Configuring Environments on page 229](#). These newly configured environments can then be used in the deployment process.



If there are multiple applications on a single environment, you can use the application codes feature in the environment definition. For additional details, see [Configuring Environments on page 229](#).

Example: ACME specifies the Environments

ACME determines that the following workflow steps require the environment settings specified in the following table ([Table 2-3](#)).

Table 2-3. Environment settings

Workflow Execution Step	Source Environment	Destination Environment
Get files from Version Control	VERSION CONTROL	DEV Server
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
Stop the Server	TEST Server	PROD Server
Migrate to Server Environment	TEST Server	PROD Server
Compile Code	TEST Server	PROD Server
Start the Server	TEST Server	PROD Server
Migrate to Database Environment	TEST Database	PROD Database

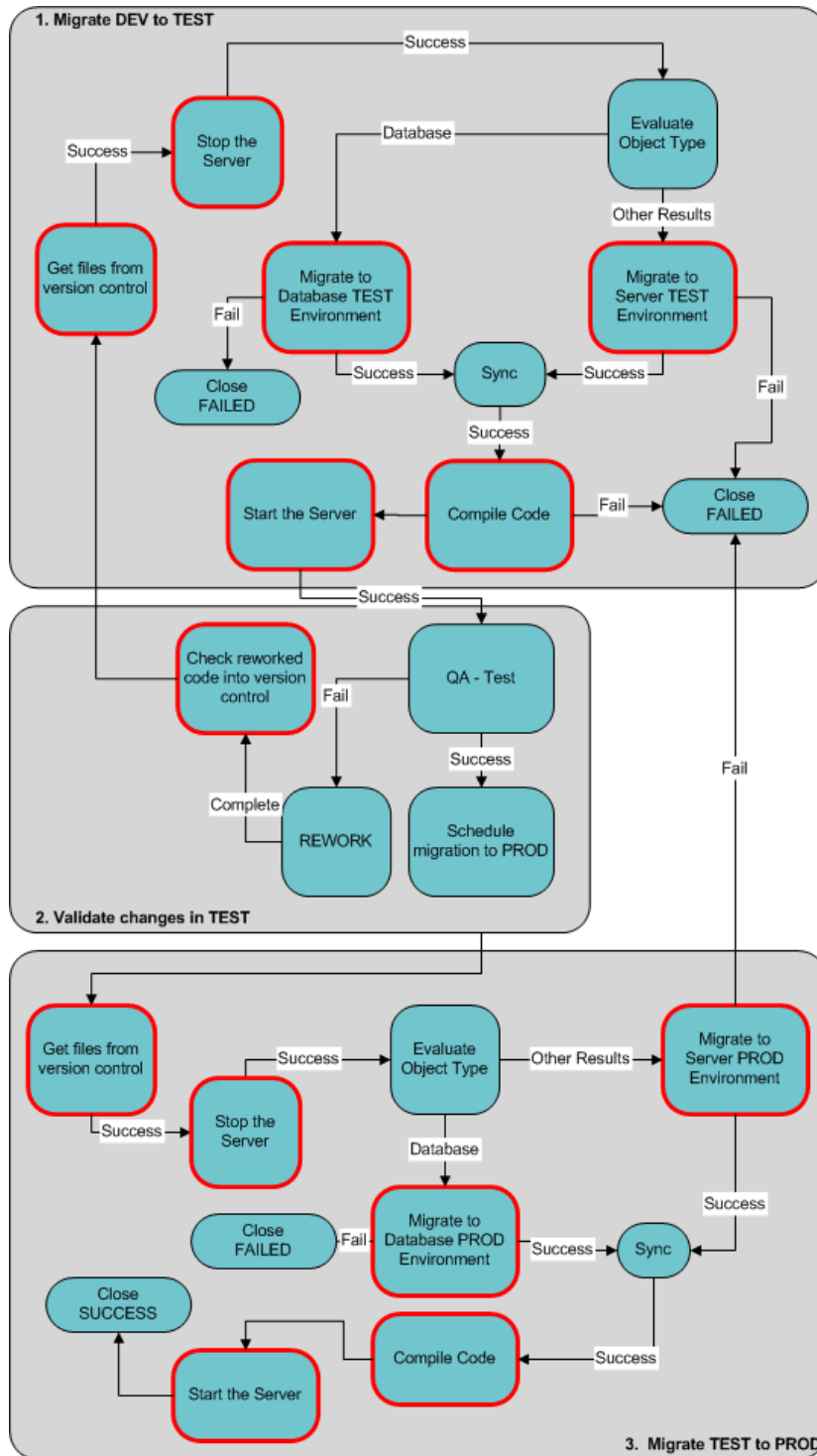


Figure 2-7. Workflow steps requiring environment specification (red)

Identify Participants and Security

Mercury Change Management allows a great deal of control over who can participate in the deployment process. Users' actions can be restricted around:

Package creation:

- Who can create packages
- Who can use a specific workflow
- Who can use specific object types

Package processing:

- Who can approve/process each step in the workflow. For this restriction, enable access by specifying specific users or security groups. Access can also be provided dynamically by having a token resolve to provide access. For more information, see *Security Model Guide and Reference*.
- Whether only "Participants" can process the packages. Participants are defined as the assigned user, the creator of the package, members of the assigned group, or any users who have access to the workflow step(s).

Managing your deployment process:

- Who can change the workflow
- Who can change each object type
- Who can modify security groups
- Who can modify environments

Whenever possible, use security groups or dynamic access (tokens). Avoid specifying a list of users to control an action; for example, specifying a list of users who can act on a workflow step. If the list of users changes (due to a departmental reorganization), you would have to update that list in many places on the workflow. 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 steps.

For the deployment process, collect the above information using the [Worksheets on page 333](#). This activity includes identifying users, grouping them into security groups, and restricting access to certain functions in Mercury Change Management.



This information is gathered in multiple worksheets. For example, workflow security is captured in the workflow step worksheet and security group membership is captured in the security groups worksheet.

For a comprehensive discussion of Mercury IT Governance screen, entity and user security, see *Security Model Guide and Reference*.

Example: ACME determines participants and security

ACME’s IT department has a single team responsible for deployments to the Financial Applications system. The team’s name is “Dev - Financial Apps.” The team consists of the following members:

- John Smith - manager of Dev - Financial Apps team
- Wendy Jones - functional designer and engineer
- Pat Lee - engineer
- Joe Franklin - QA manager
- Matt Davis - QA engineer and tester
- Raj Satish - Database expert and engineer

Within this group of users, there are some logical divisions of labor. Using this division, ACME constructs the following security groups.

Table 2-4. ACME's security groups

Security Group	Members	Responsibilities
Financial Apps - Manage Deployment	John Smith Joe Franklin	Responsible for deployment process. These people have the ability to modify the deployment process (workflow, object types, and security groups). They can also act on any step in the process.
Financial Apps - Database	Raj Satish	Responsible for deployments to the database environment. Also responsible for the correct setup of the database environment and its definition in Mercury Change Management.
Financial Apps - Engineer	Wendy Jones Pat Lee	Responsible for designing changes and deploying them to the server. Also responsible for the commands used in the object type to deploy objects to the server.
Financial Apps - QA	Joe Franklin Matt Davis	Responsible for testing the changes and reporting on the outcome.

Using these security groups and user definitions, ACME collects specific information related to their deployment process. This information will be considered later when defining your security groups and workflows.

Package Creation Security:

Table 2-5. ACME package creation security

Action	Users allowed to perform action	Controlled by: (Users, Security Group, Token)
Create a package	Wendy Jones; Pat Lee; Raj Satish	Financial Apps - Engineer Financial Apps - Database Financial Apps - Manage Deployment
Use the deployment workflow	Wendy Jones; Pat Lee; Raj Satish	Financial Apps - Engineer Financial Apps - Manage Deployment
Use the file object type	Wendy Jones; Pat Lee; Raj Satish	Financial Apps - Engineer Financial Apps - Database Financial Apps - Manage Deployment
Use the database object type	Raj Satish	Financial Apps - Database Financial Apps - Manage Deployment

Notice that the Financial Apps - Manage Deployments group was added to each action. This provides a single group with override privileges to keep the process moving.

Package Processing Security:

ACME decides not to use Mercury Change Management's participant restriction functionality in their deployment process. For additional details on this configuration option, see *Security Model Guide and Reference*. [Table 2-6 on page 54](#) documents which users can act on a specific step in the Workflow. ACME also indicates how they would like to control which users can act on each step. They select to exclusively use security groups and tokens. Notice that multiple criteria can be specified to enable access to a single step: for example, specify two security groups and a TOKEN [PKG.CREATED_BY] to enable access. Users who meet any of the requirements (members of at least one security group or the contextual value of the token) can act on the step.

Only a subset of the workflow steps are included in the below table. To see the process referenced in this table, see [Figure 2-2 on page 34](#).

Table 2-6. ACME package processing security, deployment workflow

Workflow Step Name	Users allowed to act on	Controlled by: (Users, Security Group, Token)
Stop the server	Wendy Jones; Pat Lee; Raj Satish	Financial Apps - Engineers; Financial Apps - Manage Deployment
Migrate to database environment	Raj Satish	Financial Apps - Database
Migrate to server environment	Wendy Jones; Pat Lee	TOKEN (PKG. CREATED_BY); Financial Apps - Manage Deployment
QA - Test	Joe Franklin; Matt Davis	Financial Apps - QA Financial Apps - Manage Deployment
Rework	Wendy Jones; Pat Lee; Raj Satish	TOKEN (PKG.ASSIGNED_TO_ USERNAME);
Schedule migration to PROD	John Smith; Joe Franklin	Financial Apps - Manage Deployment

ACME must also specify who can modify the existing process. This level of security is configured using ownership settings and security group access grants. Work with the instance administrator to configure user and security group definitions. For more information on these topics, see *Security Model Guide and Reference*.

Table 2-7. ACME - security around managing the process

Action	Users allowed to perform action	Controlled by: (Users, Security Group, Token)
Modify the Workflow	John Smith; Joe Franklin	Financial Apps - Manage Deployment
Modify the File Object Type	Wendy Jones; Pat Lee	Financial Apps - Engineering
Modify the Database Object Type	Raj Satish	Financial Apps - Database
Modify the Environment definitions in the Mercury ITG Center	Raj Satish	Financial Apps - Database

Establish Communication Points and Visibility

Determine the communication points and methods for providing visibility into the process and Package statuses. The following features help increase visibility:

- Notifications on the workflow step
- Portlets on the Dashboard
- Reports

It is possible to send a notification when a workflow step becomes eligible, has a specific outcome, or has a specific error. For each workflow step in the process, collect the following information using the [Worksheets on page 333](#):

Table 2-8. Information to gather for workflow step notifications

Workflow Step name	Include notification for step? (Yes/No)
Step 1 - Name	Yes
Step 2 - Name	No
Step 3 - Name	No

For each step that requires a notification, gather the following information:

Table 2-9. Information to gather for workflow step notifications

Parameter	Description
Workflow Step Name	The name of the step that requires a workflow.
Notification Event (All, Eligible, Specific Result, Specific Error)	Specifies the event that triggers the notification. The possible values are All , Eligible , Specific Result , or Specific Error .
Value (for Specific Result)	Specifies that a notification is sent for the selected result.
Error (for Specific Error)	Specifies that a notification is sent for the selected error.
Recipient	Determine who should receive the message. you can choose to send the notification to users based on: Username , Email Address , Security Group , Standard Token , or User Defined Token .
Message	Determine what the message will say. Also determine if it will contain a link to the package.

Example: ACME configures Notifications

ACME determines that they would like to add a notification to the following steps:

Table 2-10. ACME - Workflow steps with notifications

Workflow Step name	When to send notification	Recipients
Migrate to Database Environment	Failed (specific result)	Financial Apps - Database
Migrate to Server Environment	Failed (specific result)	Financial Apps - Engineer
Compile Code	Failed (specific result)	Financial Apps - Engineer
QA - Test	Eligible	Financial Apps - QA
Rework	Eligible	Financial Apps - Engineer Financial Apps - Database
Schedule Migration to PROD	Eligible	Financial Apps - Manage Deployment

Chapter 3 Configuring Workflows

In This Chapter:

- *Overview of Workflows*
- *Mapping Workflows*
- *Opening the Workflow Workbench*
- *Creating Workflows*
 - *Configuring General Information for Workflows*
 - *Dragging and Dropping Workflow Steps*
 - *Choosing Workflow Steps*
 - *Adding Close Workflow Steps*
 - *Adjusting Workflow Step Sequences*
 - *Specifying the First Step*
 - *Verifying and Enabling Workflows*
- *Configuring Workflow Steps*
 - *Configuring General Information for Workflow Steps*
 - *Configuring Security for Workflow Steps*
 - *Configuring Notifications for Workflow Steps*
 - *Configuring Timeouts for Workflow Steps*
 - *Configuring Transitions for Workflow Steps*
 - *Configuring Validations for Workflow Steps*
- *Integrating Object Types and Workflows*
 - *Integrating Object Type Commands and Workflows*
- *Integrating Environments and Workflows*
 - *Choosing Source Environments Based on Application Code*
- *Integrating Request and Package Workflows*
 - *Setting Up WF - Jump/Receive Step Label Validations*
 - *Generating Jump Step Sources*

- *Generating Receive Step Sources*
 - *Including Jump and Receive Workflow Steps in Workflows*
-

Overview of Workflows



Note

This chapter covers information concerning Demand Management workflows, Change Management workflows, and Release Management workflows.

A workflow represents a business process and is used to map business rules and processes to your organization.

The following is a list of the basic components of a workflow:

- **Begin.** For each workflow, you must explicitly define the first eligible workflow step.
- **Workflow step.** Workflow steps are events that are linked together to form a complete workflow. The following lists the basic workflow steps:
 - **Decision step.** Decision steps represent manual activities performed outside of Mercury IT Governance Center. For example, a decision step is where a user or group of users approves a request.
 - **Execution step.** Execution steps represent actions that are automated through Mercury IT Governance Center. For example, updating a web page with the results of a test.
 - **Condition step.** Condition steps are logic steps used for complex workflow processing, such as allowing the workflow to proceed only when each of the workflow steps are completed.
 - **Subworkflows step.** A subworkflow step represents 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 workflow step that must be communicated to another workflow step. For example, the results of a decision step is **Approved** and **Not Approved**.

- **Workflow step security.** Workflow step security determines who has permission to execute or choose a result for a workflow step. For example, for a Approve Request decision step, only the IT project manager can **Approve** or **Not Approved** the request.
- **Notification.** Notifications are emails alerts sent out at specific workflow steps. For example, for a Approve Request decision step, an email alert is sent to the product manager.
- **Close step.** Close steps indicate the end of the workflow. The close step is an execution step that marks the request as completed.

Figure 3-1 illustrates the basic workflow components in a workflow.

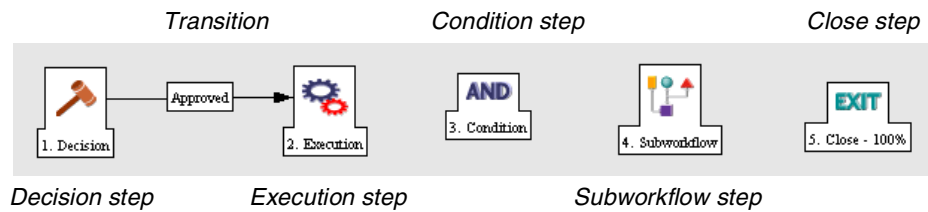


Figure 3-1. Workflow components

Mapping Workflows

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

Step 1. Create a block diagram. Map each Workflow Step Worksheet as a one block in the diagram. On the block diagram include transitions, workflow step security, and notifications.

Step 2. Map the block diagram to the workflow. Open the Workflow Workbench window and start a new workflow. Map each component from the block diagram to the new workflow.

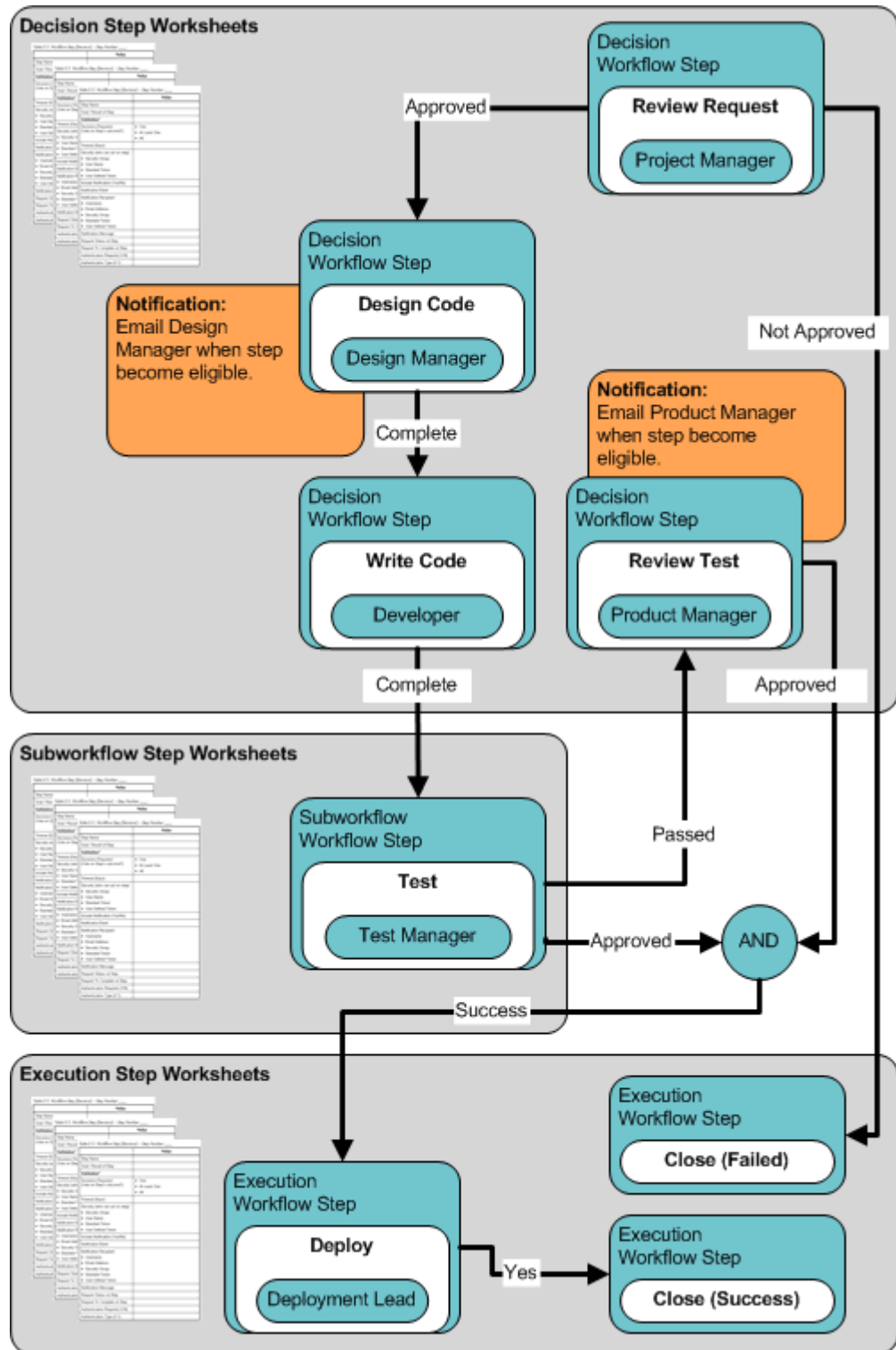


Figure 3-2. Step 1. Create a block diagram

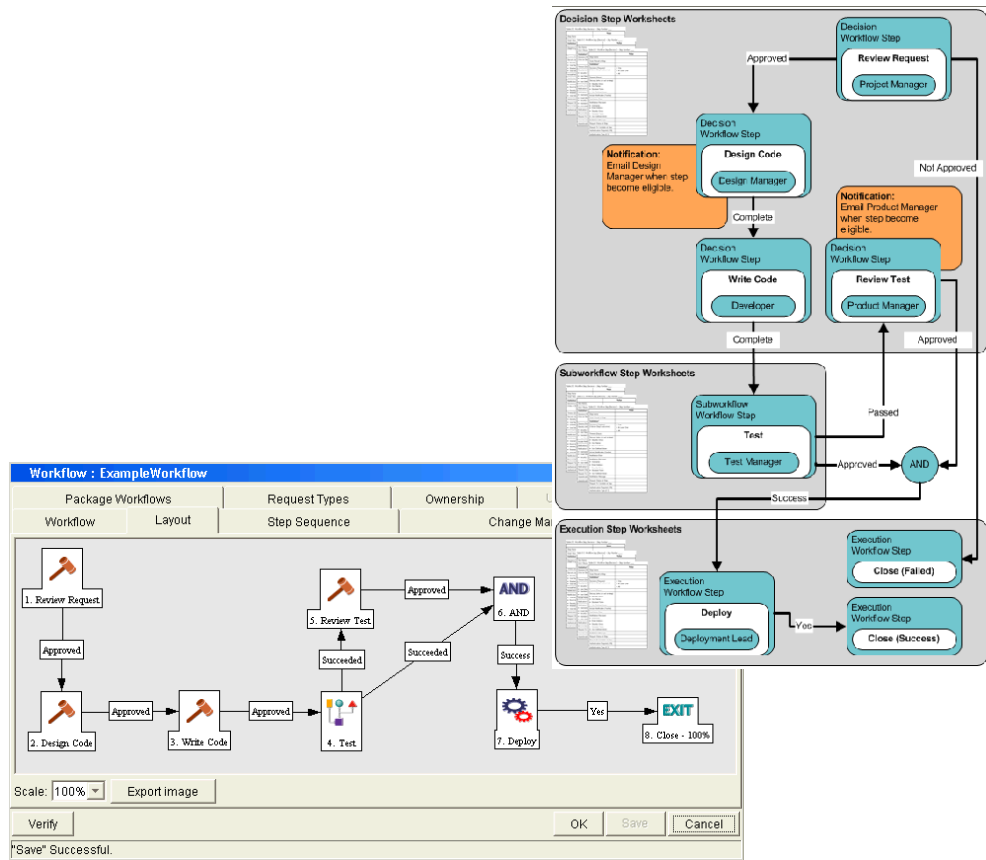


Figure 3-3. Step 2. Create the workflow

Opening the Workflow Workbench

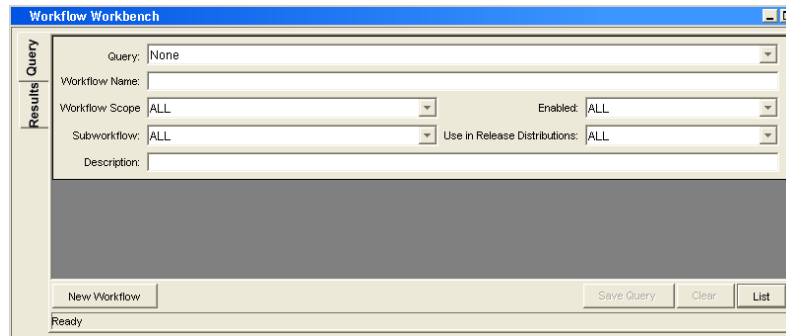
To open the Workflow Workbench:

1. Log on to the Mercury IT Governance Center.
2. From the menu bar, select **Administration > Open Workbench**.
3. A Workbench status window opens. A few minutes later, a Warning - Security window opens.
4. In the Warning - Security window, select **Yes**.

The Workbench opens.

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

The Workflow Workbench window opens.



For More Information

For information on how to search and select an existing workflow, copy a workflow, and delete a workflow, see *Getting Started*.

Creating Workflows

Starting a new workflow requires knowing how to use the Workflow Workbench. This section covers the basics on how to create a workflow.

Configuring General Information for Workflows

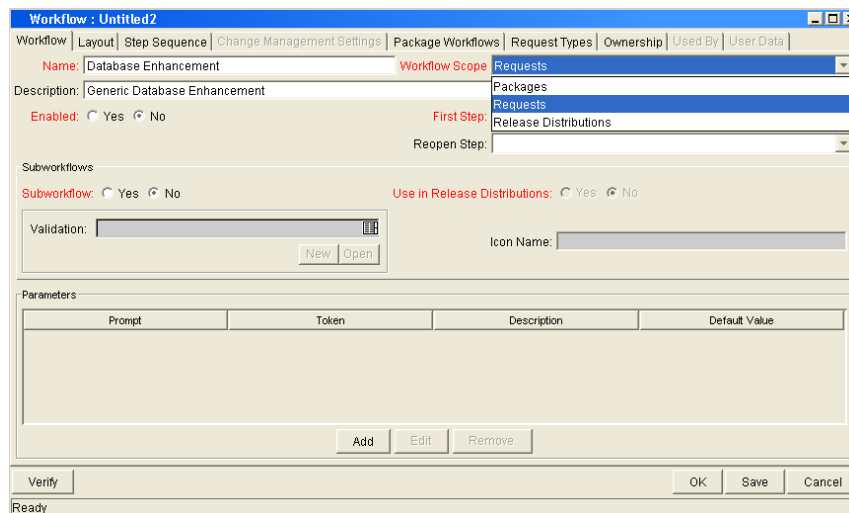
To enter basic workflow information:

1. Open the Workflow Workbench.

To open the Workflow Workbench, see [Opening the Workflow Workbench on page 63](#). The Workflow Workbench window opens.

2. In the Workflow Workbench window, click **New Workflow**.

The Workflow window opens.



The screenshot shows the 'Workflow : Untitled2' window with the following configuration:

- Name:** Database Enhancement
- Workflow Scope:** Requests (selected from a dropdown menu)
- Description:** Generic Database Enhancement
- Enabled:** Yes No
- First Step:** Release Distributions (selected from a dropdown menu)
- Reopen Step:** (empty dropdown menu)
- Subworkflows:**
 - Subworkflow:** Yes No
 - Use in Release Distributions:** Yes No
- Validation:** (text field with 'New' and 'Open' buttons)
- Icon Name:** (text field)
- Parameters:** A table with columns: Prompt, Token, Description, Default Value. Below the table are 'Add', 'Edit', and 'Remove' buttons.
- Buttons:** 'Verify', 'OK', 'Save', 'Cancel'.
- Status:** Ready

3. In Name, enter the name of the new workflow.

4. In Workflow Scope, select one of the following from the drop-down list:

- For Mercury Change Management packages, select **Packages**.
- For Mercury Demand Management requests, select **Requests**.
- For Mercury Change Management releases and distributions, select **Release Distributions**.

5. In the Workflow window, click **Save**.

Click **OK** to save the changes and close the Workflow window. Click **Save** to save the changes and leave the Workflow window open. Click **Cancel** to drop the changes and close the Workflow window.

Dragging and Dropping Workflow Steps

A library of existing workflow steps resides in the Workflow Step Source window. The Workflow Step Source window includes a Filter by field, allowing you to see only the workflow steps available for you to use.

Workflow steps are assembled into workflows in the **Layout** tab of the Workflow window. Select a workflow step from the Workflow Step Sources window and drag and drop the workflow step onto the **Layout** tab. As part of the drag and drop process, a Workflow Step window opens. The Workflow Step window is used to configure the following:

- General information concerning the workflow step
- Security for the workflow step
- Notifications for the workflow step
- Timeouts for the workflow step

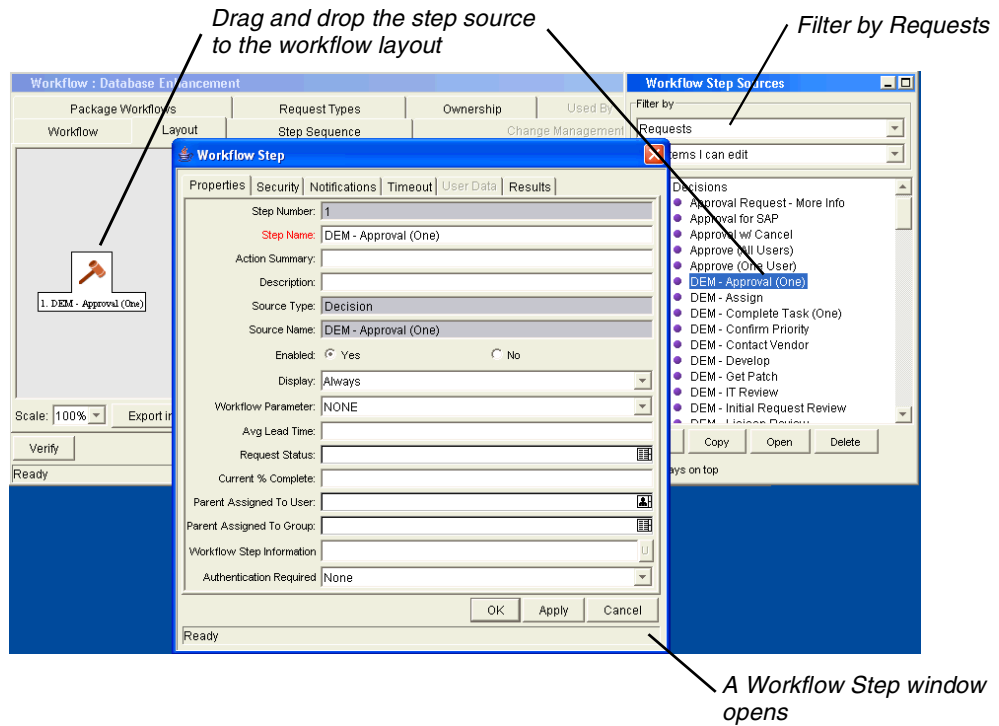


Figure 3-4. Drag and drop

Choosing Workflow Steps

Mercury IT Governance Center comes with many pre-defined workflow steps. These workflow steps are located in the Workflow Step Source window. Workflow steps in the Workflow Step Source window are filtered using the Filter by field. Select an entry from the Filter by drop-down list filter the workflow steps. The following lists the folders found in the Workflow Step Source window:

- Decision
- Conditions
- Executions
- Subworkflows

To evaluate a workflow step, determine which of the four workflow folders is required for your workflow step. Open the Workflow Step Source folder and open those workflow steps that seem to best meet your needs (see [Figure 3-5 on page 67](#)).

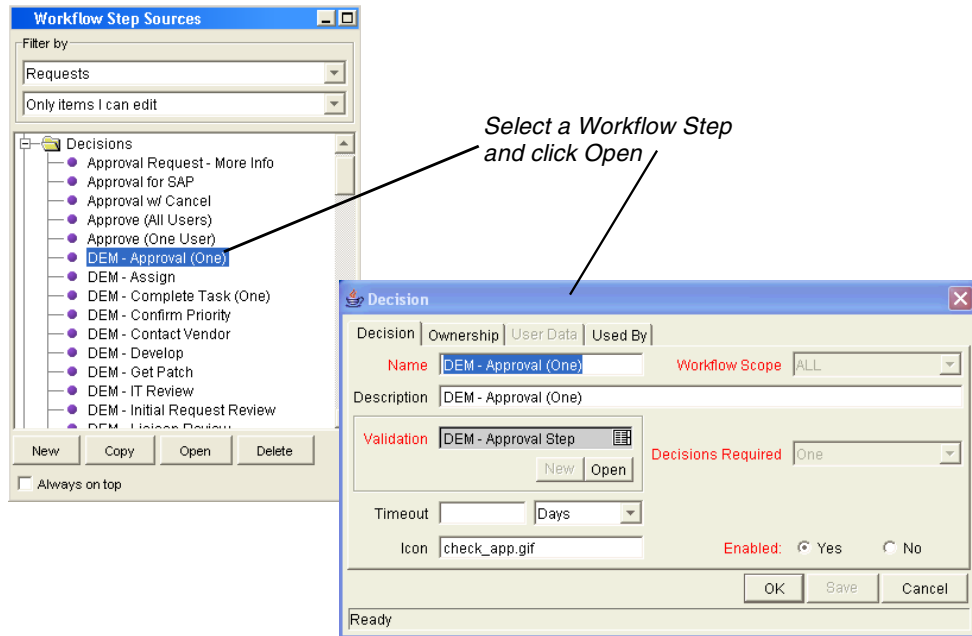


Figure 3-5. Workflow step source

Check the validation to see if the validation values meet your transition requirements. The validation values are the acceptable values a workflow step can have (see [Configuring Validations for Workflow Steps](#) on page 111).

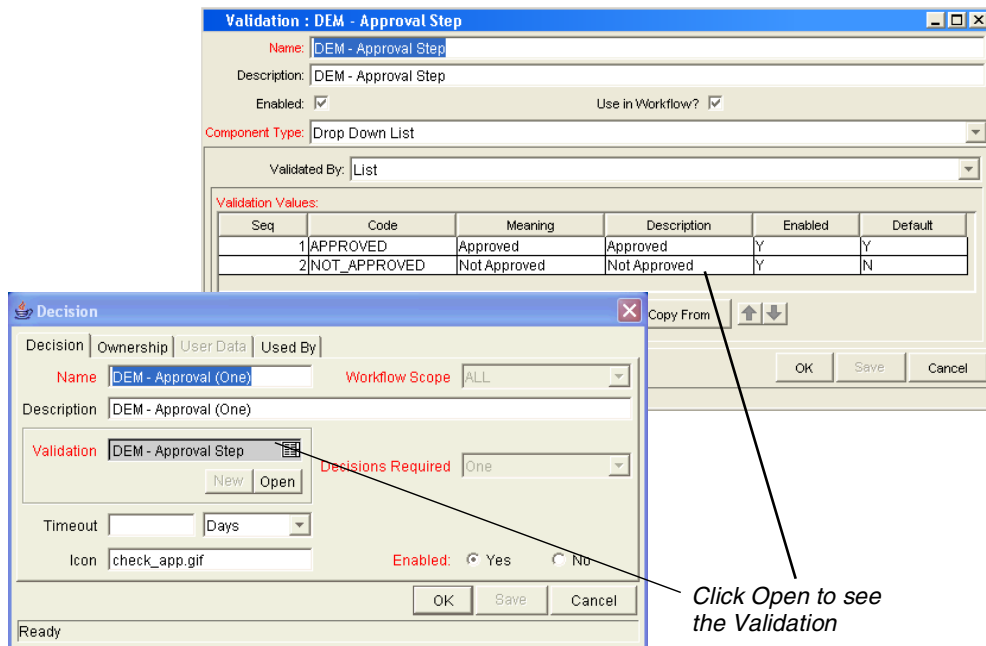


Figure 3-6. Workflow step source validation

Overview of Decisions Workflow Steps

Decision workflow steps represent manual activities performed outside of Mercury IT Governance Center. 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 when each of the workflow steps are completed. The following is a list of the conditions workflow steps.

- **AND.** An AND condition is satisfied only if all workflow steps leading to it reach the status they are supposed to attain.

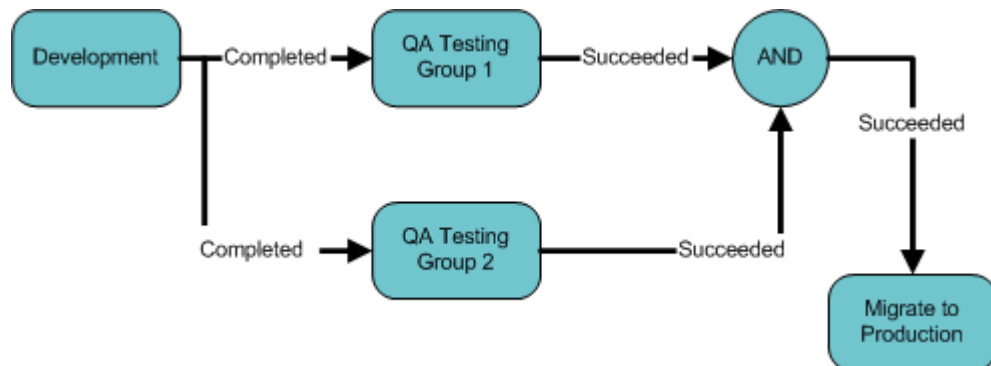


Figure 3-7. AND example

- **OR.** An OR condition is successful when at least one of the workflow steps leading to it reaches the status it is supposed to attain.

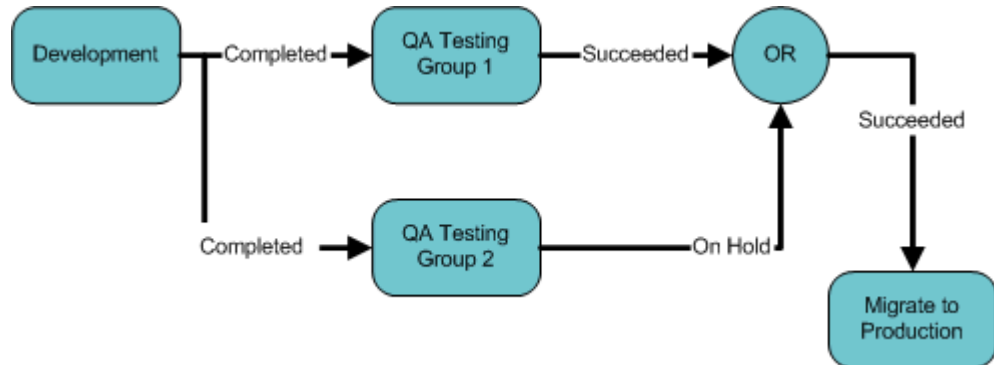


Figure 3-8. OR example

- **SYNC.** A SYNC workflow step is successful only if all the package lines of that package reach the status expected for the workflow step right before the SYNC step.

Consider the business process outlined in [Figure 3-9](#). According to the flow chart, when **QA Testing Group 1** is successful for all package lines, SYNC becomes successful and the next step, **Migrate to Prod** becomes eligible.

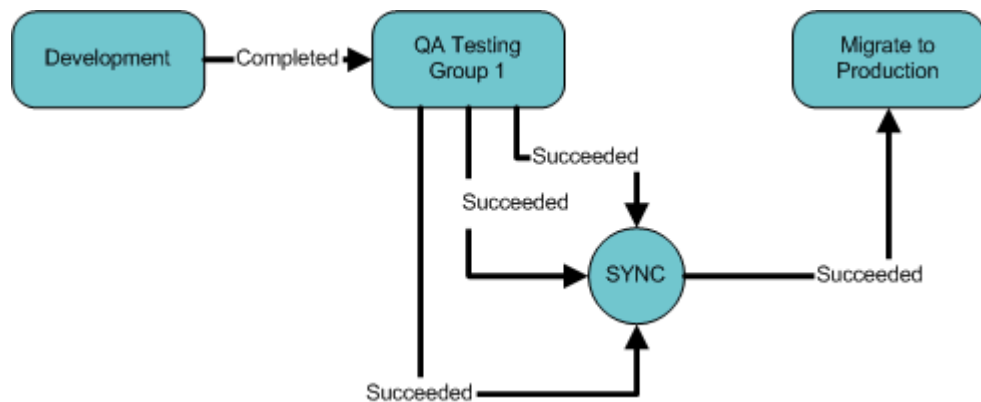


Figure 3-9. SYNC example

- **FIRST LINE and LAST LINE.** For FIRST LINE, only the first line to reach the condition workflow step takes the **True** transition. All successive 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.

Consider the business process outlined in *Figure 3-10*. This business process could be part of a website maintenance life cycle. As part of this life cycle, three HTML files are being processed on three respective package lines in a single package. The website updates are large enough to warrant shutting down the web server while migrating the changes.

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 steps, the server remains down until the last active line reaches the condition step. The last active line takes the **True** transition and the web server starts up and the maintenance is complete.

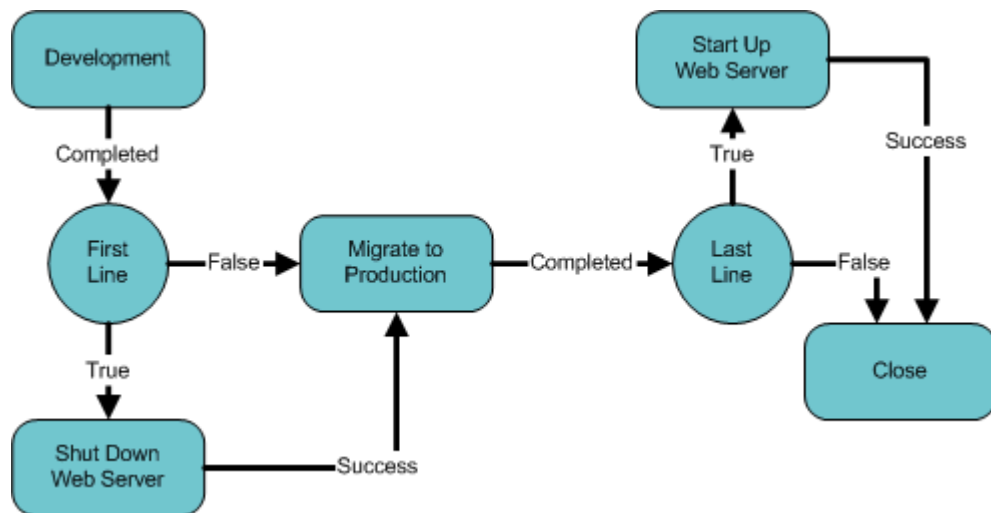


Figure 3-10. *FIRST LINE and LAST LINE example*

Overview of Execution Workflow Steps

Execution workflow steps represent actions that are automated through the Mercury IT Governance Center. Execution workflow steps include such activities as:

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

Overview of Subworkflow Workflow Steps

A subworkflow step represent multiple workflow steps (the subworkflow) in a workflow. When the workflow process reaches the subworkflow step, it follows the path defined in that subworkflow. Subworkflows can either close within that workflow or return to the parent workflow.

Adding Close Workflow Steps

Every workflow, no matter how long or short, must include a close workflow step (see [Figure 3-11](#)). A close workflow step is a specific kind of execution workflow step and can be found in the Executions folder of the Workflow Step Sources window.

There are three close workflow steps:

- **Close (Immediate Success).** Close (Immediate Success) immediately completes a request or package with a status of **Success**.
- **Close (Manual Success).** Close (Manual Success) requires manual intervention to complete a request or package. The status of the request or package is set to **Success**.
- **Close (Immediate Failure).** Close (Immediate Failure) immediately completes a request or package with a status of **Failure**.

Add a close workflow step to a workflow as you would any other workflow step. See [Figure 3-11](#).

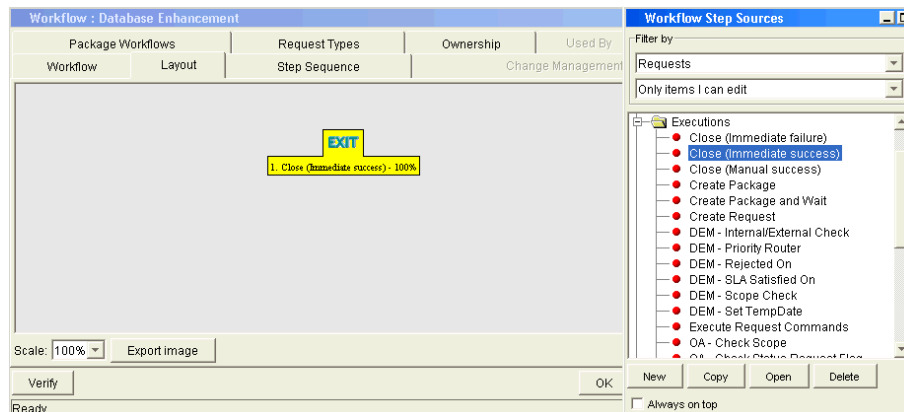


Figure 3-11. Close workflow step

Adjusting Workflow Step Sequences

Once all of the workflow steps are assembled in the **Layout** tab, you can adjust the sequence of the steps. In the Workflow window, select the **Step Sequence** tab. The **Step Sequence** tab lists all of the workflow steps. Select a workflow step and click the **Arrow** icons at the bottom of the tab to move the selected workflow step up or down.

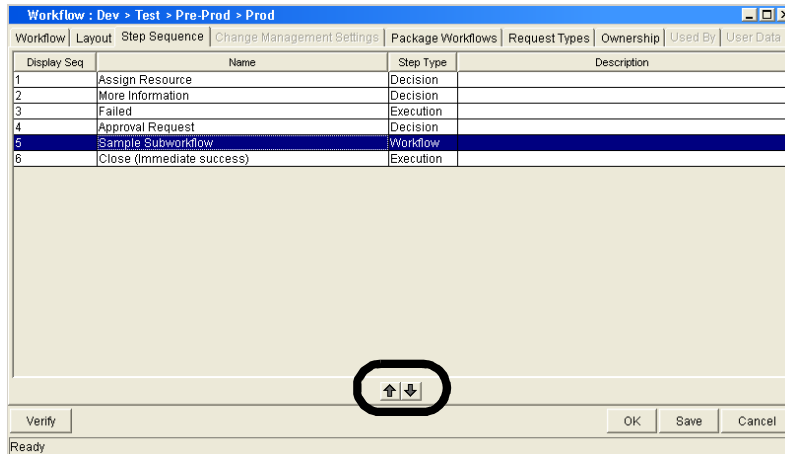


Figure 3-12. Step sequence tab

Specifying the First Step

Once all of the workflow steps are assembled and properly sequenced, you must specify the first step in the workflow process. To specify the first step, open the **Workflow** tab in the Workflow Workbench window (see [Figure 3-13](#)). Open the drop-down list in the First Step field and select the first step.

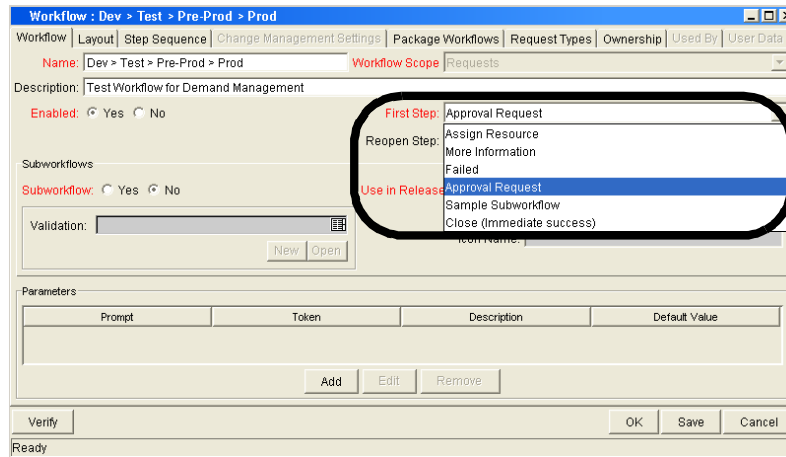


Figure 3-13. Workflow tab

Verifying and Enabling Workflows

Verifying and enabling a workflow are the last steps required to make a workflow available. Verify a workflow checks to make sure the logic of the workflow is correct. Enabling a workflow makes the workflow available for use.

To verify a workflow:

1. Open the Workflow Workbench.

To open the Workflow Workbench, see [Opening the Workflow Workbench on page 63](#). The Workflow Workbench window opens.

2. Open a workflow.

The Workflow window opens. The **Workflow** tab is displayed.

3. At the bottom left-hand corner of the **Workflow** tab, click **Verify**.

The logic of the workflow is checked and a status window is returned.

To enable a workflow:

1. Open the Workflow Workbench.

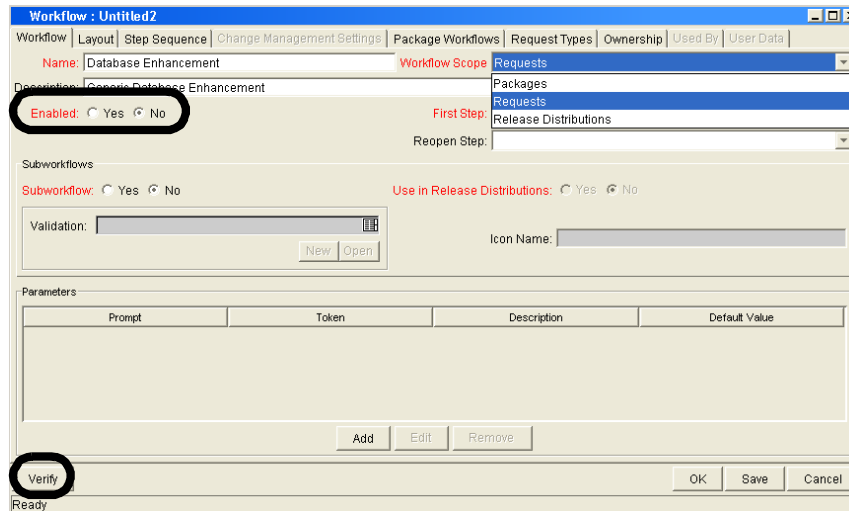
To open the Workflow Workbench, see [Opening the Workflow Workbench on page 63](#). The Workflow Workbench window opens.

2. Open a workflow.

The Workflow window opens. The **Workflow** tab is displayed.

3. In the **Workflow** tab, in the Enable field, select **Yes**.

The workflow is enabled.



4. In the **Workflow** tab, click **Save**.

The changes to the workflow are saved.

Configuring Workflow Steps

Every time a workflow step is dragged and dropped from the Workflow Step Source window to the **Layout** tab of the Workflow window, a Workflow Step window opens. You can enter none, some, or all of the known information at the initial window opening, or you can open the Workflow Step window later in the workflow design process.

Information entered in the Workflow Step window can be gathered from the appropriate Workflow Step Worksheets. Each Workflow Step window includes the following tabs:

- **Properties.** General information concerning the workflow step is defined under the **Properties** tab.
- **Security.** Permission settings for specific individuals or groups authorized to act on a workflow step are defined under the **Security** tab.
- **Notifications.** Emails can be sent when a workflow step becomes eligible or after a workflow step is complete. 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. Notifications are defined under the **Notifications** tab.
- **Timeout.** Timeouts determine how long a workflow step can remain inactive before generating an error. Timeouts are defined under the **Timeout** tab.
- **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.** Each workflow step includes a validation. The **Results** tab lists the validation, the component type and the results.

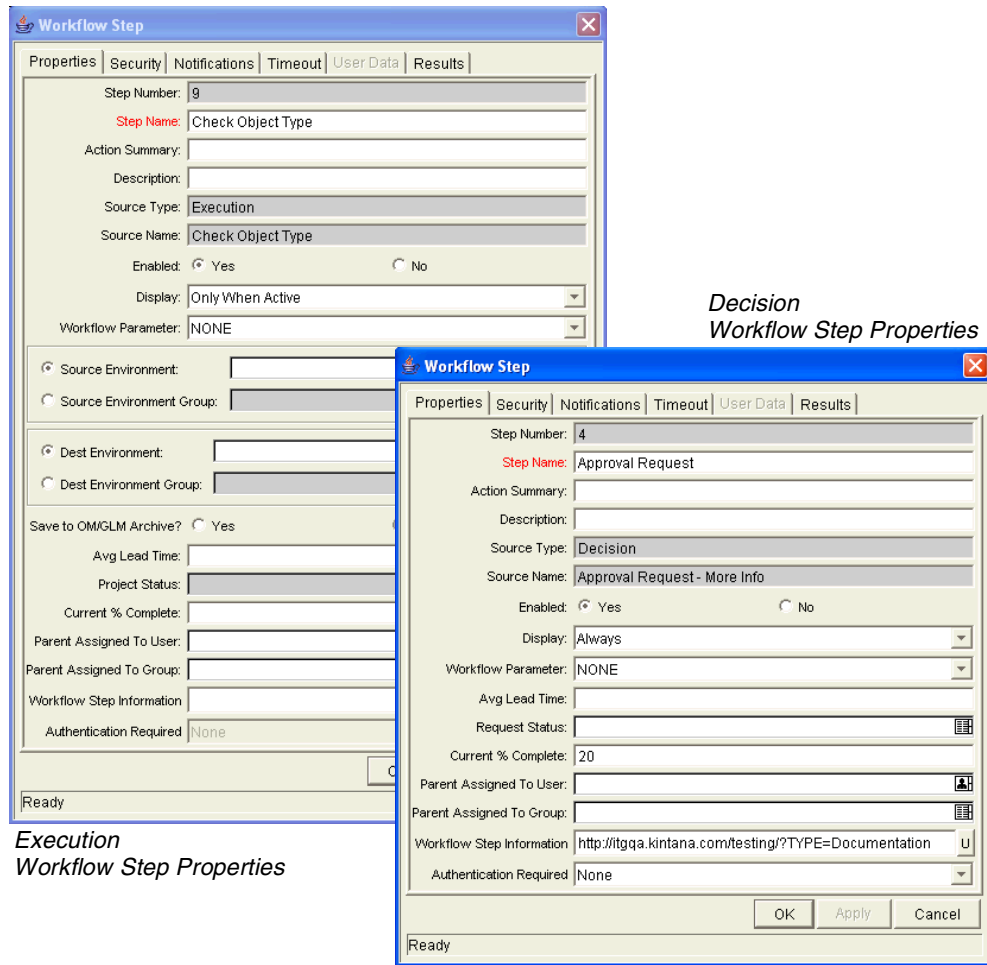


Figure 3-14. Workflow step properties

Configuring General Information for Workflow Steps

General information concerning the workflow step is defined in the Workflow Step window under the **Properties** tab.

To add general information to a workflow step:

1. Open the Workflow Workbench.

To open the Workflow Workbench, see [Opening the Workflow Workbench on page 63](#). The Workflow Workbench window opens.

2. Open a workflow.

The Workflow window opens.

3. Open the **Layout** tab of the Workflow window.

4. Right-click a workflow step.

The workflow step is highlighted. A menu window opens.

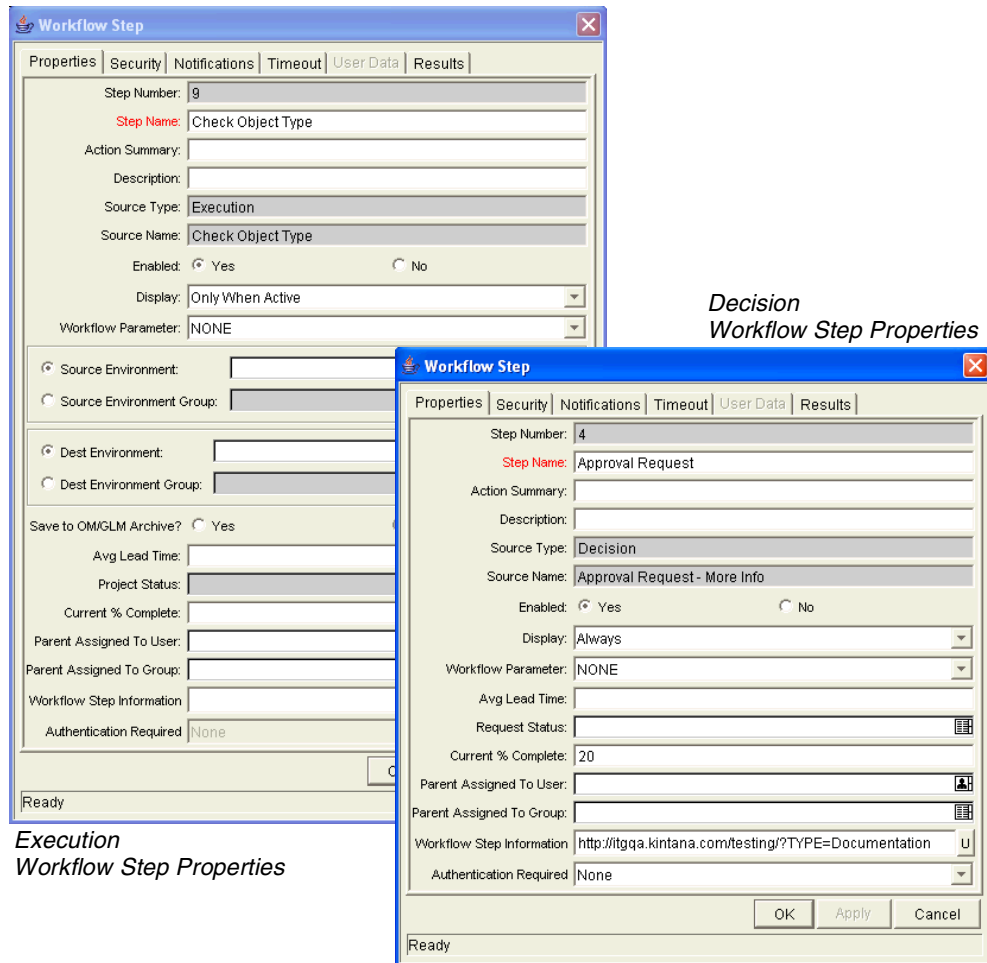
5. In the menu window, select **Edit**.

The Workflow Step window opens.

6. Make sure you are in the **Properties** tab.

The **Properties** tab is the default tab for the Workflow Step window.

7. Complete the fields in the **Properties** tab.



*Execution
Workflow Step Properties*

*Decision
Workflow Step Properties*

8. In the **Properties** tab of the Workflow Step window, click **Save**.

The changes to the workflow are saved.

Configuring Security for Workflow Steps

Workflow steps need to have permission settings to define the specific individuals or groups who are authorized to act on each workflow step.

To add workflow step security to a workflow step:

1. Open the Workflow Workbench.

To open the Workflow Workbench, see [Opening the Workflow Workbench on page 63](#). The Workflow Workbench window opens.

2. Open a workflow.

The Workflow window opens.

3. Open the **Layout** tab of the Workflow window.

4. Right-click a workflow step.

The workflow step is highlighted. A menu window opens.

5. In the menu window, select **Edit**.

The Workflow Step window opens.

6. In the Workflow Step window, select the **Security** tab.

The **Security** tab opens.

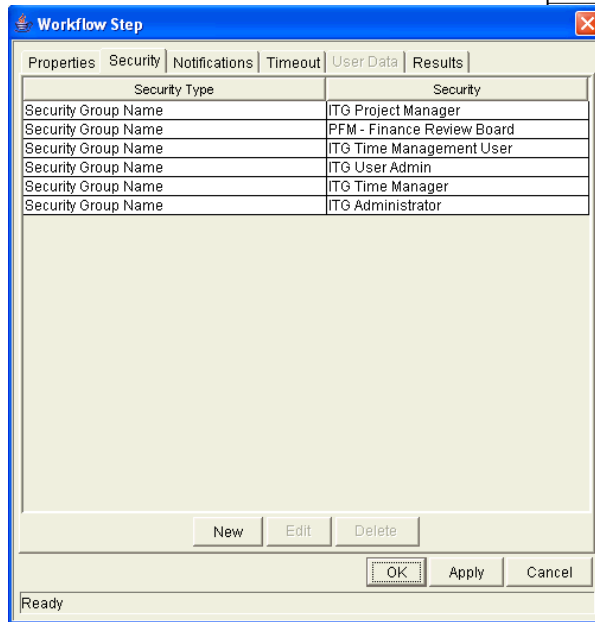
7. In the **Security** tab, click **New**.

The Workflow Step Security window opens.

Decision Workflow Step Worksheets

Table A-5. Workflow step [decision], step number ____.

	Value
Step Name	
Goal / Result of Step	
Validation*	
Decisions Required (Vote on Step's outcome?)	<input type="checkbox"/> One <input type="checkbox"/> At Least One <input type="checkbox"/> All
Security (who can act on step): <input type="checkbox"/> Security Group <input type="checkbox"/> User Name <input type="checkbox"/> Standard Token <input type="checkbox"/> User Defined Token	
Include Notification (Yes/No)	
Event	
Recipient	
Address	
Group	
Defined Token	
Message	
Status at Step	
Complete at Step	
Notification Required (Y/N)	
Notification Type (if Y)	



8. In the Workflow Step Security window, select the security type from the drop-down list.

The security type options are:

- **Enter a Security Group Name.** Select a security group to act upon the workflow step. Selecting a security group changes the name of the autocomplete field to Security Group. The security type is dynamically changed to **Security Group**.
- **Enter a Username.** Select a user to act upon the workflow step. Selecting a user changes the name of the autocomplete field to Username. The security type is dynamically changed to **Username**.

- **Enter a Standard Token.** Select a standard token to act upon the workflow step. Selecting a standard token changes the name of the autocomplete field to Standard Token. The security type is left undefined. Select a standard token from the autocomplete field. The Security Type field is defined based on the standard token chosen.
 - **Enter a User Defined Token.** Select a user defined token to act upon the workflow step. Selecting a user defined token changes the name of the autocomplete field to User Defined Token. The security type is dynamically changed to a drop-down list. The **Tokens** button is enabled. Click **Tokens** to open the Token Builder window and select a token. Select one of the following from the drop-down list:
 - **Username.** The selected token resolves to a username.
 - **User ID.** The selected token resolves to a user ID.
 - **Security Group Name.** The selected token resolves to a security group.
 - **Security Group ID.** The selected token resolves to a security group ID.
9. In the Workflow Step Security window, click **OK**.
- The Workflow Step Security window closes.
10. In the Workflow Step window, click **OK**.
- The Workflow Step window closes.
11. In the **Security** tab of the Workflow Step window, click **OK**.
- The changes are added to the workflow.

Configuring Dynamic Security for Workflow Steps

Workflow steps can also be configured so that its security is determined at runtime based on information entered in the request or package.

To configure a workflow step with dynamic security:

1. Open the Workflow Workbench.

To open the Workflow Workbench, see [Opening the Workflow Workbench on page 63](#). The Workflow Workbench window opens.

2. Open a workflow.

The Workflow window opens.

3. Open the **Layout** tab of the Workflow window.

4. Right-click a workflow step.

The workflow step is highlighted. A menu window opens.

5. In the menu window, select **Edit**.

The Workflow Step window opens.

6. In the Workflow Step window, select **Security** tab.

The **Security** tab opens.

7. In the **Security** tab, click **New**.

The Workflow Step Security window opens.

8. In the Workflow Step Security window, select the security type from the drop-down list.

The security type options are:

- **Enter a Security Group name.** Select a security group to act upon the workflow step. Selecting a security group changes the name of the autocomplete field to Security Group. The security type is dynamically changed to **Security Group**.

- **Enter a Username.** Select a user to act upon the workflow step. Selecting a user changes the name of the autocomplete field to Username. The security type is dynamically changed to **Username**.
 - **Enter a Standard Token.** Select a standard token to act upon the workflow step. Selecting a standard token changes the name of the autocomplete field to Standard Token. The security type is left undefined. Select a standard token from the autocomplete field. The Security Type field is defined based on the standard token chosen.
 - **Enter a User Defined Token.** Select a user defined token to act upon the workflow step. Selecting a user defined token changes the name of the autocomplete field to User Defined Token. The security type is dynamically changed to a drop-down list. The **Tokens** button is enabled. Click **Tokens** to open the Token Builder window and select a token. Select one of the following from the drop-down list:
 - **Username.** The selected token resolves to a username.
 - **User ID.** The selected token resolves to a user ID.
 - **Security Group Name.** The selected token resolves to a security group.
 - **Security Group ID.** The selected token resolves to a security group ID.
9. In the Workflow Step Security window, click **OK**.
- The Workflow Step Security window closes.
10. In the Workflow Step window, click **OK**.
- The Workflow Step window closes.
11. In the **Security** tab of the Workflow Step window, click **OK**.
- The changes are added to the workflow.

Configuring Notifications for Workflow Steps

Notifications can be sent when a workflow step becomes eligible or after a workflow step is complete. 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). Notifications are defined in the **Notifications** tab of the Workflow Step window.

When configuring a notification for a workflow step, consider the following:

- When to send the notification
- Who should receive the notification
- What the notification should say

Review the Workflow Step Worksheet for notification information.

To add a notification to a workflow step:

1. Open the Workflow Workbench.

To open the Workflow Workbench, see [Opening the Workflow Workbench on page 63](#). The Workflow Workbench window opens.

2. Open a workflow.

The Workflow window opens.

3. Open the **Layout** tab of the Workflow window.

4. Right-click a workflow step.

The workflow step is highlighted. A menu window opens.

5. In the menu window, select **Edit**.

The Workflow Step window opens.

6. In the Workflow Step window, select the **Notifications** tab.

The **Notifications** tab opens.

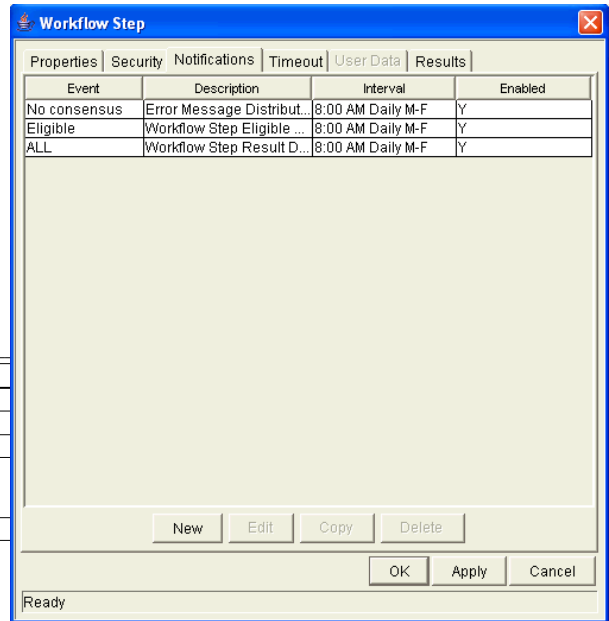
7. In the **Notifications** tab, click **New**.

The Add Notification for Step window opens. The Add Notification for Step window has two tabs: **Setup** and **Message**.

8. In the **Setup** tab of the Add Notification for Step window, configure:

- When the notification should be sent (Event and Interval)
- Who receives the notification (Recipients)

9. In the **Message** tab of the Add Notification for Step window, configure the body of the notification.



Decision Workflow Step Worksheets

Table A-5. *Workflow step [decision], step number*

Step Name	
Goal / Result of Step	
Validation*	
Decisions Required (Vote on Step's outcome?)	<input type="checkbox"/> One <input type="checkbox"/> At Least One <input type="checkbox"/> All
Timeout (Days)	
Security (who can act on step):	
<input type="checkbox"/> Security Group <input type="checkbox"/> User Name <input type="checkbox"/> Standard Token <input type="checkbox"/> User Defined Token	
include Notification (Yes/No)	
Notification Event	
Notification Recipient:	
<input type="checkbox"/> Username <input type="checkbox"/> Email Address <input type="checkbox"/> Security Group <input type="checkbox"/> Standard Token <input type="checkbox"/> User Defined Token	
Notification Message	
Request Status at Step	
Request % Complete at Step	
Authentication Required (Y/N)	
Authentication Type (if Y)	

10. In the Add Notification for 1Step window, click **OK**.

The notification is added to the **Notifications** tab. You can send different notifications to different recipients for different events by clicking **New** and repeating the previous process. The following lists some of the reasons you might want to send different notifications for a single workflow step:

- Send different notifications depending on the result of the step
- Send different notifications depending on the type error

- Send the notifications to a different set of users depending on the step's result or error
 - Specifying different intervals or reminders based on the type of step error
11. In the **Notifications** tab of the Workflow Step window, click **OK**.
The changes are added to the workflow.

Configuring Setup Tabs

You can configure a workflow step to send notifications at different times, different intervals, 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, open the **Notifications** tab.

See [Configuring Notifications for Workflow Steps on page 84](#). The **Notifications** tab opens.

2. In the **Notifications** tab, click **New**.

The Add Notification for Step window opens. The Add Notification for Step window has two tabs: **Setup** and **Message**.

3. In the Add Notification for Step window, select the **Setup** tab.

The **Setup** tab is the default tab.

4. Configure the **Setup** tab as follows:

Field	Value	Notes
Event	Eligible	
Interval	Immediate	A notification can be sent at different intervals. For example, you might choose to send a notification of a final approval step at midnight so that it is ready for approval in the morning. Note also that multiple notifications to a single recipient can be brought together in a batch and sent together. Selecting an interval other than Immediate will allow this batching to occur.
Send Reminder	Yes/No	This field is optional. 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 .
Enabled	Yes	

5. In the **Setup** tab, click **OK**.

The **Setup** tab closes. The Workflow Step window opens.

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

The changes are added to the workflow.

Sending Notifications when Workflow Steps have Specific Results

It is possible to configure a notification to be sent when a workflow step has a specific decision or execution result. The value for these results is determined by the workflow step source's validation.

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

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

See [Configuring Notifications for Workflow Steps on page 84](#). The **Notifications** tab opens.

2. In the **Notifications** tab, click **New**.

The Add Notification for Step window opens. The Add Notification for Step window has two tabs: **Setup** and **Message**.

3. In the Add Notification for Step window, select the **Setup** tab.

The **Setup** tab is the default tab.

4. Configure the **Setup** tab as follows:

Field	Value	Notes
Event	Specific Result	
Value	Select the value to trigger the Notification.	The list of values is determined by the workflow step source's validation. Therefore, this selection will always be limited to the possible results of the step.
Interval	Immediate	A notification can be sent at different intervals. For example, you might choose to send a notification of a final approval step at midnight so that it's ready for approval in the morning. Note also that multiple notifications to a single recipient can be brought together in a batch and sent together. Selecting an interval other than Immediate will allow this batching to occur.
Send Reminder	Yes/No	This field is optional. 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 .
Enabled	Yes	

5. In the **Setup** tab, click **OK**.

The **Setup** tab closes. The Workflow Step window opens.

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

The changes are added to the workflow.

Sending Notifications When Workflow Steps Have Specific Errors

It is possible to configure the notification to be sent when a workflow step has a specific error. [Table 3-1](#) lists the workflow step errors.

Table 3-1. Specific errors for workflow steps

Specific Error	Meaning
No consensus	When all users of all security groups, or users linked to the workflow step need to vote, and there is no consensus.
No recipients	When none of the security groups linked to the workflow step has users linked to it, no user can act on the workflow step.
Timeout	When 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 request is expecting a message from a package line that is cancelled or closed.
Child closed	For wf_receive or wf_jump steps, a package line is expecting a message from a request that is cancelled or closed.
No parent	For wf_receive or wf_jump steps, a request is expecting a message from a package line that has been deleted.
No child	For wf_receive or wf_jump steps, a package line is expecting 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	When the package level subworkflow receives multiple results from package lines that traversed through it.

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

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

See [Configuring Notifications for Workflow Steps on page 84](#). The **Notifications** tab opens.

2. In the **Notifications** tab, click **New**.

The Add Notification for Step window opens. The Add Notification for Step window has two tabs: **Setup** and **Message**.

3. In the Add Notification for Step window, select the **Setup** tab.

The **Setup** tab is the default tab.

4. Configure the **Setup** tab as follows:

Field	Value	Notes
Event	Specific Error	
Error	Select the value to trigger the Notification.	This is a standard set of errors. See Sending Notifications When Workflow Steps Have Specific Errors on page 89 .
Interval	Immediate	A notification can be sent at different intervals. For example, you might choose to send a notification of a final approval step at midnight so that it's ready for approval in the morning. Note also that multiple notifications to a single recipient can be brought together in a batch and sent together. Selecting an interval other than Immediate will allow this batching to occur.
Send Reminder	Yes/No	This field is optional. 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 .
Enabled	Yes	

5. In the **Setup** tab, click **OK**.

The **Setup** tab closes. The Workflow Step window opens.

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

The changes are added to the workflow.

Specifying the Time Notifications are Sent

Use the Interval field in the workflow step to specify when the notification will be sent. The interval determines how frequently the notification will be sent.

To send the time notification are sent:

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

See [Configuring Notifications for Workflow Steps on page 84](#). The **Notifications** tab opens.

2. In the **Notifications** tab, click **New**.

The Add Notification for Step window opens. The Add Notification for Step window has two tabs: **Setup** and **Message**.

3. In the Add Notification for Step window, select the **Setup** tab.

The **Setup** tab is the default tab.

4. Configure the **Setup** tab, configure the Interval field as follows:

- **8:00 AM Daily M-F:** This notification is sent every 8:00 a.m. on the next available work day after the notification event occurs.
- **Hourly M-F:** This notification is sent every hour, starting on the next available work day after the notification event occurs.
- **Immediate:** This notification is sent immediately.

5. In the **Setup** tab, click **OK**.

The **Setup** tab closes. The Workflow Step window opens.

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

The changes are added to the workflow.

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, open the **Notifications** tab.

See [Configuring Notifications for Workflow Steps on page 84](#). The **Notifications** tab opens.

2. In the **Notifications** tab, click **New**.

The Add Notification for Step window opens. The Add Notification for Step window has two tabs: **Setup** and **Message**.

3. In the Add Notification for Step window, select the **Setup** tab.

The **Setup** tab is the default tab.

4. In the **Setup** tab, configure the Interval field as follows:

Field	Value	Notes
Event		Selects any event except for All .
Send Reminder	Yes	Selecting Yes enables the Reminder Days field.
Reminder Days	Enter the number of days.	The number of days to wait before sending a reminder notification.

5. In the **Setup** tab, click **OK**.

The **Setup** tab closes. The Workflow Step window opens.

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

The changes are added to the workflow.

Configuring Notification Recipients

When creating a notification, at least one recipient must be added for the message. 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, open the **Notifications** tab.

See [Configuring Notifications for Workflow Steps on page 84](#). The **Notifications** tab opens.

2. In the **Notifications** tab, click **New**.

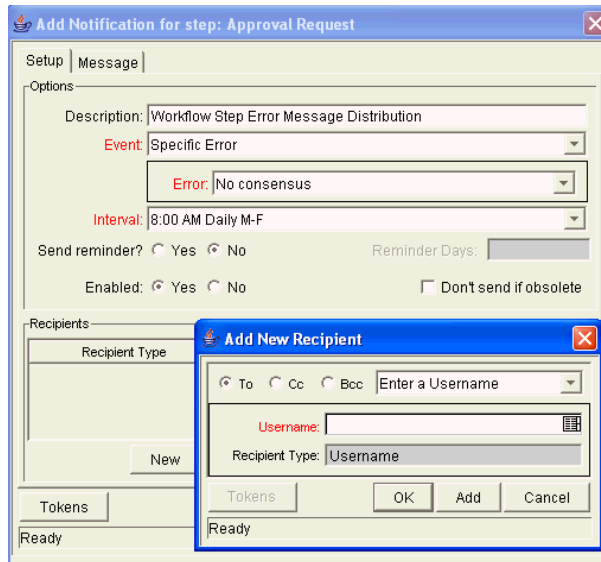
The Add Notification for Step window opens. The Add Notification for Step window has two tabs: **Setup** and **Message**.

3. In the Add Notification for Step window, select the **Setup** tab.

The **Setup** tab is the default tab.

4. In the **Setup** tab, click **New**.

The Add New Recipient window opens.



5. Select how to specify the recipient from the drop-down list:

- **Enter a Security Group.** Select a specific security group, and all enabled users in the group with email addresses will receive the notification.
- **Enter a Username.** Select a specific user to receive the notification. The user must have an email address.
- **Enter an Email Address.** Enter any email address of the notification.
- **Enter a Standard Token.** Select from a list of system tokens that corresponds to a user, security group, or email address.
- **Enter a User Defined Token.** Enter any field token that corresponds to a user, security group, or email address.

Selecting a value will automatically update the Recipient Type field. For example, selecting **Enter a Security Group** will change the Recipient Type field to **Security Group**.

6. Enter the specific value that corresponds to the recipient type selected above.

This can be a username, email address, security group, or a token.

Use security groups or dynamic access (distributions) to define the notification recipients whenever possible. Avoid specifying a list of users or an individual user's email address. If the list of users changes (due to a departmental or company reorganization), that list would have to be updated manually. By using a security group instead of a list of users, the security group can be updated once, and the changes will be propagated throughout the workflow steps.

Use distributions when sending a notification to an undetermined party. For example, the notification can be configured to be sent to the Assigned to User by specifying **[REQ.ASSIGNED_TO_USERID]** in the Add New Recipient window.

7. In the Add New Recipient window, click **OK**.

8. In the **Setup** tab, click **OK**.

The **Setup** tab closes. The Workflow Step window opens.

9. In the Workflow Step window, click **OK**.

The changes are added to the workflow.

Configuring Message Tabs

It is possible to construct the notification's message to ensure that it contains the correct information or instructions for the recipient. For example, if a notification is sent to instruct you that a request requires your approval, the message should instruct you to log onto Mercury IT Governance Center and update the request's status. Additionally, the notification should include a link (URL) to the referenced request.

Notifications include the following features to make them easier to configure and use:

- Select from a number of pre-configured notification templates to more quickly construct the body of your message.
- The body of the notification can be plain text or HTML.
- Multiple tokens can be included in the notification. These tokens will 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, open the **Notifications** tab.

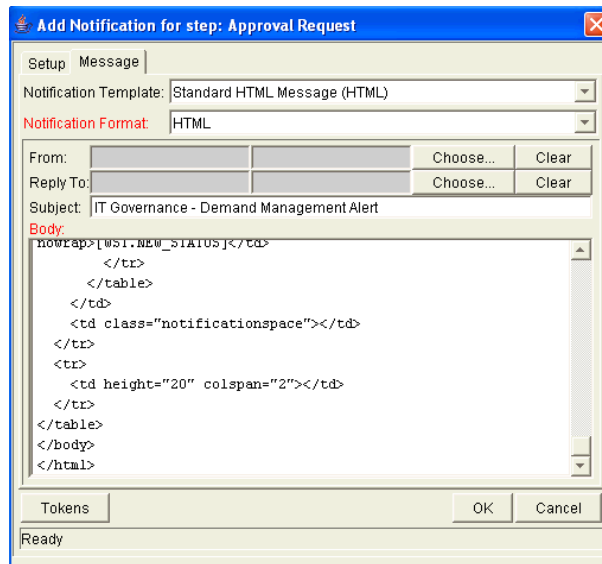
See [Configuring Notifications for Workflow Steps on page 84](#). The **Notifications** tab opens.

2. In the **Notifications** tab, click **New**.

The Add Notification for Step window opens. The Add Notification for Step window has two tabs: **Setup** and **Message**.

3. In the Add Notification for Step window, select the **Message** tab.

The **Message** tab opens.



4. Select a Notification Template from the drop-down list.

This updates the contents in the Body section with the information defined for the selected template.

5. In the Notification Format field, select **HTML** or **Plain Text** from the drop-down list.

Selecting **HTML** allows more flexibility when formatting the look and feel of the notification. The HTML code can be written and tested in any HTML editor and then pasted into the Body window.

6. Select values for the From and Reply to fields.

7. Construct the body of the message.

When constructing the body, consider utilizing the following:

- Token for the URL to the Request Detail page. See [Table 3-2 on page 98](#) for a list of these tokens.
- Token for the URL to the package (Workbench or standard interface). See [Table 3-2 on page 98](#) for a list of these distributions.
- **Tokens in the body of the message.** Click the **Tokens** button to access the Token Builder window where tokens can be added to the message body.

- **Tokens related to specific package lines or request detail fields.** Add tokens to the Linked Token list to include tokens that resolve information related to the individual package line or request detail field.

8. In the **Message** tab, click **OK**.

The Add Notification for Step window closes. The **Notifications** tab is enabled.

9. In the **Notifications** tab, click **OK**.

The changes to the workflow are saved.

Using Tokens in the Message Body

It is possible to select any of the available tokens accessed through 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 needing 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 or Netscape Messenger), 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.

The most commonly used smart URL tokens for packages and requests are described in [Table 3-2](#).

Table 3-2. 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.

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 long to set the timeout.

To set timeouts for a workflow step:

1. Open the Workflow Workbench.

To open the Workflow Workbench, see [Opening the Workflow Workbench on page 63](#). The Workflow Workbench window opens.

2. Open a workflow.

The Workflow window opens.

3. Open the **Layout** tab of the Workflow window.

4. Right-click a workflow step.

The workflow step is highlighted. A menu window opens.

5. In the menu window, select **Edit**.

The Workflow Step window opens.

6. In the Workflow Step window, select the **Timeout** tab.

The **Timeout** tab opens.

Decision Workflow Step Worksheets

Table A-5. Workflow step [decision], step number ____.

	Value
Step Name	
Goal / Result of Step	
Validation*	
Decisions Required (Vote on Step's outcome?)	= One = At Least One
Timeout (Days)	
Security (Who can act on step):	
Security Group	
Time	
Id Token	
Signed Token	
Authentication (Yes/No)	
Event	
Recipient:	
Name	
Address	
Group	
Id Token	
Signed Token	
Message	
Status at Step	
Complete at Step	
Validation Required (Y/N)	
Validation Type (if Y)	

7. Configure the timeout as follows:

- **Use Workflow Step Source.** This is the default setting. The Workflow Step Source field determines the workflow step's timeout. The Timeout and Interval fields are disabled.
- **Specific Value.** You can enter a value for the workflow step's timeout according to the Timeout Type entry.

8. In the **Timeout** tab of the Workflow Step window, click **Apply**.

The changes are applied to the workflow.

Configuring Transitions for Workflow Steps

Transitions are the rules that logically connect workflow steps. They are added to a workflow to establish what direction a process should take based on the results of a workflow step. For example, a request is entered into a request resolution system. The first step in the workflow is to 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 had been dragged and dropped from the Workflow Step Source window to the **Layout** tab of the Workflow window. You can choose a transition between workflow steps based on the following workflow step results:

- **Specific result.** The specific result follows this transition. The specific results is the default workflow step results. Specific results are based on the workflow step's validation.
- **Other results.** All other results that do not have transitions set follow this transition.
- **All results.** All results follow this transition.
- **Specific Event. (Demand Management only.)** The specific event follows this transition. Specific events are based on the workflow step's validation. Used only for Demand Management IT solution.
- **Specific Error.** The specific error follows this transition.
- **Other Errors.** All other errors that do not have transitions set follow this transition.
- **All Errors.** All errors follow this transition.

Adding Transitions Based on Specific Results

To add a Specific Result transition:

1. Open the Workflow Workbench.

To open the Workflow Workbench, see [Opening the Workflow Workbench on page 63](#). The Workflow Workbench window opens.

2. Open a workflow.

The Workflow window opens.

3. Open the **Layout** tab of the Workflow window.

4. Right-click a workflow step.

The workflow step is highlighted. A menu window opens.

5. In the menu window, select **Add Transition**.

The menu window closes. The workflow step remains highlighted.

6. Select the destination workflow step for the transition.

A line with an arrowhead appears between the workflow steps. The Define Transition and Step Transitions windows opens. The Define Transition window is enabled and has many options on how to define the transition. The most common transition is Specific Results. For information on other transitions definitions, see [Adding Transitions not Based on Specific Results on page 103](#).

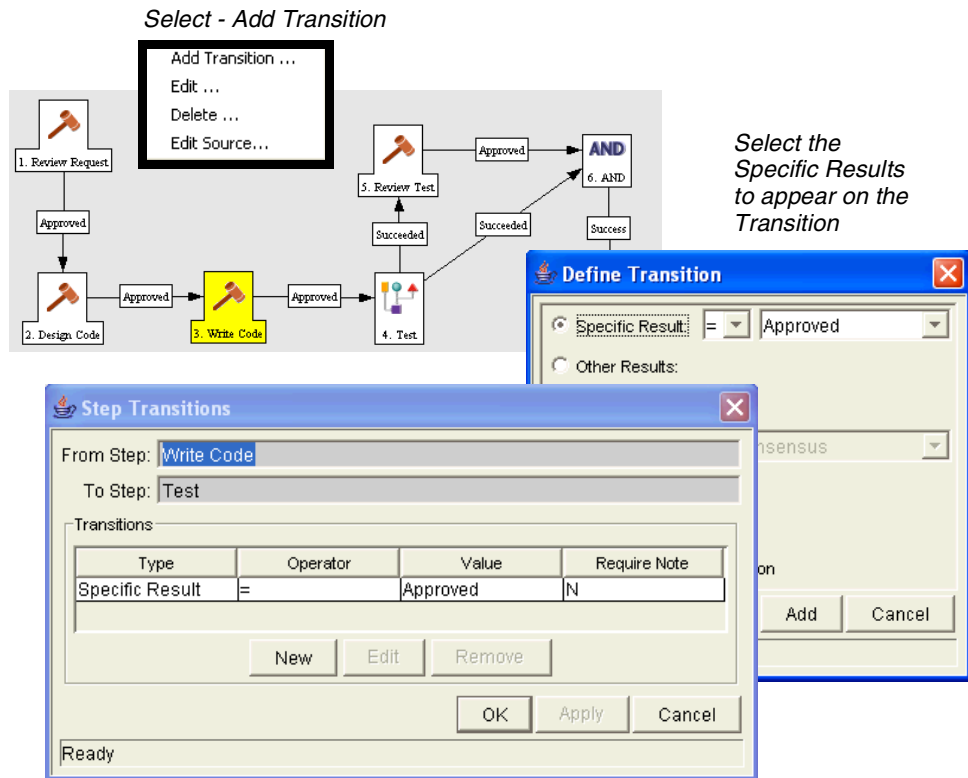
7. In the Define Transitions window, from the Specific Results drop-down list, select the appropriate transition.

8. In the Define Transition window, click **OK**.

The Define Transition window closes. The Step Transitions window is enabled.

9. In the Step Transitions window, click **Apply** or **OK**.

The transition is added to the Step Transitions window. Clicking **Apply** keeps the Step Transition window open. To add another validation to the transition, click **New** and 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.



10. At the bottom of the **Layout** tab, click **Save**.

The changes to the workflow are saved.

Adding Transitions not Based on Specific Results

Transitions are added to a workflow after a workflow step had 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

It is possible to 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 list are determined by a SQL query of a database table.

As with any execution step, configure this transition to be an immediate or manual step.

Adding Transitions Based on All But One Specific Value

It is possible to transition based on all but one specified value. You can use Other Results when multiple transitions are exiting a single step. Other Results will act 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 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.

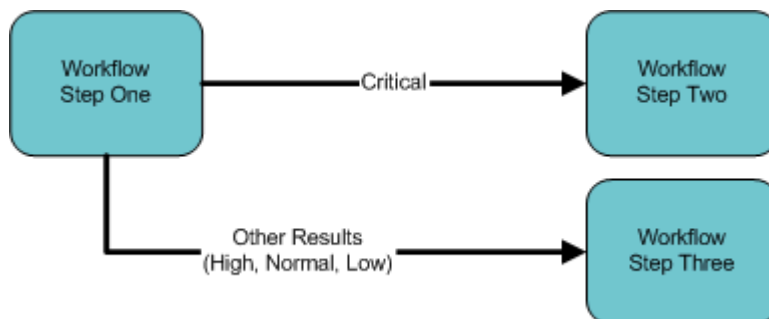


Figure 3-15. Transitions using other results

Adding Transitions Based on All Results

It is possible to 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. Since 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 select **All Results**.

Consider using an **All Results** transition when kicking off a sub-process. Note that you can still define transitions based on Specific Results or errors when you select **All Results**. You can bring the process together later using an AND step.

Adding Transitions Based on Errors

It is possible to transition based on a specific error that occurs during an execution step. The business process may then be branched based on likely execution errors such as **Timeout**, **Command Execution**, or **Invalid Token** (see [Table 3-3](#)). When adding a transition, select the Specific Error radio button in the Define Transition window. From the drop-down list, select the error.

Table 3-3. Workflow transition errors

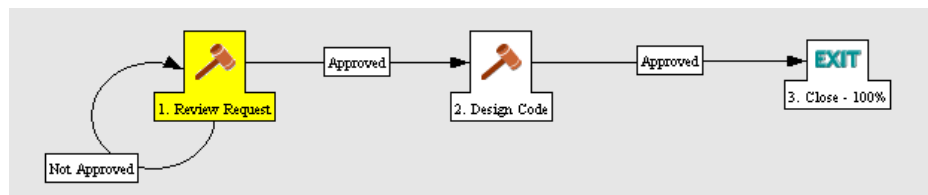
Transition Option	Meaning
Multiple Return Results	When the package level subworkflow receives multiple results from package lines that traversed through it.
No consensus	When all users of all security groups, or users linked to the workflow step need to vote, and there is no consensus.
No recipients	When none of the security groups linked to the workflow step has users linked to it, no user can act on the workflow step.

Table 3-3. *Workflow transition errors [continued]*

Transition Option	Meaning
Timeout	When 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 the Same Step

It is possible to retain the option of resetting failed execution workflow steps, rather than immediately transition along a failed path. This is often helpful when troubleshooting the execution (see [Figure 3-16](#)).

Figure 3-16. *Transitioning back to the same step*

When the commands execute successfully, they will follow the Success transition path. However, when the commands fail, they will not transition out of the step because no transition has been defined for the **FAILED** result. The user has to manually select the workflow step and select **FAILED - RETRY**. The execution will re-run.

Do not use an immediate execution workflow step when a **FAILED** result is feeding directly back into the execution workflow step. This would result 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. Open the Workflow Workbench.

To open the Workflow Workbench, see [Opening the Workflow Workbench on page 63](#). The Workflow Workbench window opens.

2. Open a workflow.

The Workflow window opens.

3. Open the **Layout** tab of the Workflow window.

4. Configure an immediate execution workflow step.

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

The Execution window opens.

- b. Complete the fields in the Execution window as specified in the following table:

Field in Execution Window	Value
Workflow Scope	Requests for request tracking and resolution processes, Packages for deployment processes, or Release Distributions for release processes.
Execution Type	Token
Processing Type	Immediate
Validation	Create a validation with the following validation values. <ul style="list-style-type: none"> • Succeeded • Failed • Failed - Reset • Failed - Rejected For details on how to create a validation, see <i>Commands, Tokens, and Validations Guide and Reference</i> .
Enabled	Yes

c. In the Execution window, click **OK**.

d. The new execution workflow step is saved and the Execution window closes.

5. Add the new execution workflow step to the workflow.

6. Right-click the immediate execution workflow step.

The workflow step is highlighted. A menu window opens.

7. Add the transition.

a. In the menu window, select **Add Transition**.

The menu window closes. The workflow step remains highlighted.

b. Select several points near the execution workflow step, then select the source workflow step.

The Define Transition and Step Transitions windows opens. The Define Transition window is enabled and has many options on how to define the transition.

- c. In the Define Transitions window, from the Specific Results drop-down list, select the appropriate transition.

The Validations in the Specific Results drop-down list are the validation created for the execution workflow step. For example, select **Failed - Reset**.

- d. In the Define Transition window, click **OK**.

The Define Transition window closes. The Step Transitions window is enabled.

- e. In the Step Transitions window, click **OK**.

The transition is added to the Step Transitions window. The Step Transitions window closes. The defined transition name is added to the transition line.

8. At the bottom of the **Layout** tab, click **Save**.

The changes to the workflow are saved.

Adding Transitions Based on Previous Workflow Step Results

It is possible to 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 of adding a transition based on a previous workflow step result are:

1. In the Workflow window, in 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, in the **Properties** tab of the Workflow Step window, specify the name of the workflow parameter in the Workflow Parameter field.

One step in this example process requires the user to route the request based on the type of change (code or database). The decision made at this step is then considered later in the process to correctly route rework of the specific type.

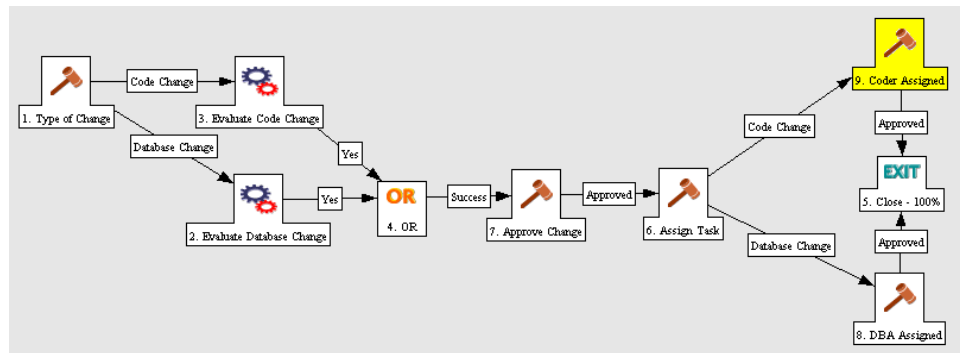


Figure 3-17. Add a transition based on a previous workflow step

To add a transition based on a previous workflow step:

1. Open the Workflow Workbench.

To open the Workflow Workbench, see [Opening the Workflow Workbench on page 63](#). The Workflow Workbench window opens.

2. Open a workflow.

The Workflow window opens. The **Workflow** tab is displayed.

3. In the **Workflow** tab, create a Workflow Parameter.

- a. In the **Workflow** tab, in the parameters section, click **Add**.

The Workflow Parameters window opens.

- b. Complete the fields in the Workflow Parameters window.

- c. In the Workflow Parameters window, click **OK**.

- d. The workflow parameter is saved and the Workflow Parameters window closes.

4. In the Workflow window, select the **Layout** tab.

The **Layout** tab opens.

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

Note that the validation used in this step should 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 as displayed in the following illustration:
- c. In the Execution window, click **OK**.
- d. The new execution workflow step is saved and the Execution window closes.

6. Add the new execution workflow step to the workflow.

- a. Add a workflow step to the workflow.

The Workflow Step window opens.

- b. In the Workflow Step window, in the **Properties** tab, select the workflow parameter from the Workflow Parameter drop-down list.
- c. In the **Properties** tab, click **OK**.

7. Add the steps and transitions as shown in [Figure 3-17 on page 109](#).

8. In the **Layout** tab, click **OK**.

The changes to the workflow are saved.

Adding Transitions To and 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 the two 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.

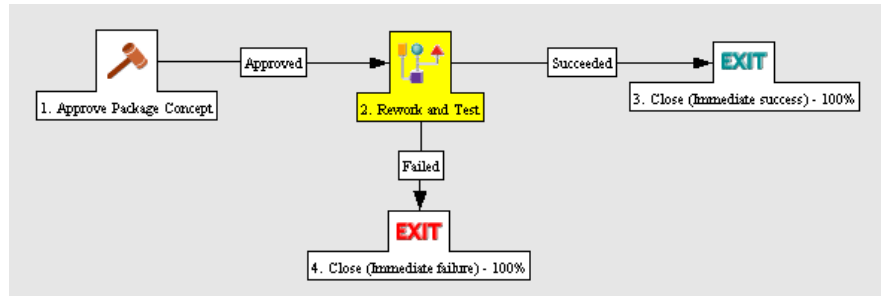


Figure 3-18. Transitioning to and from subworkflows

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. The subworkflow field cannot be edited 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 entered 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 Source 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 3-19 on page 112](#) illustrates the Decisions window of the Approve (One User) decision workflow step and the validation listed in the Decision window. In this example, the validation is **WF - Approval Step**. By checking the validation, **WF - Approval Step** has two validation values:

- **Approved**
- **Not Approved**

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

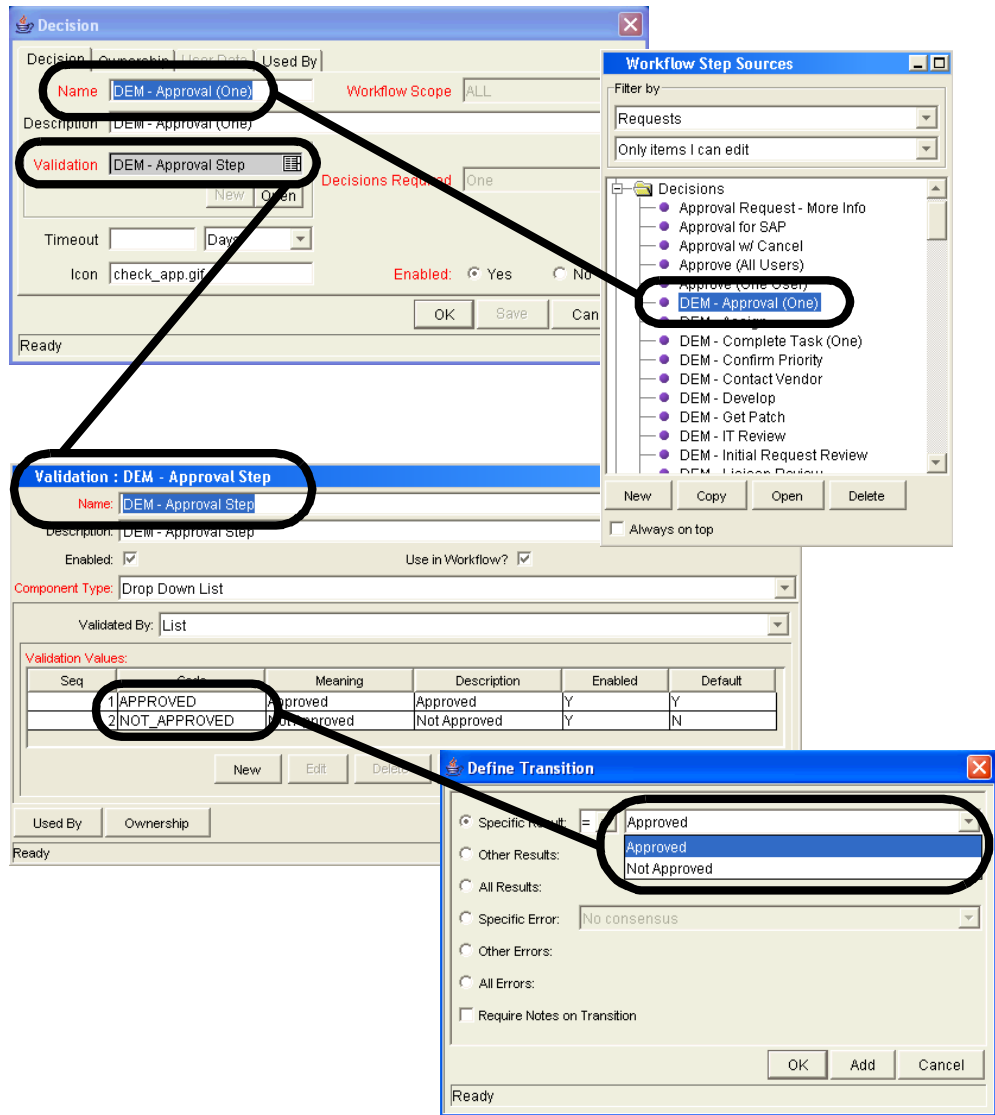


Figure 3-19. Workflow step sources and validations

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 execute without error, the result of **Success** is returned, otherwise, **Failure** is returned.

Table 3-4 summarizes this relationship between validations and execution types.

Table 3-4. 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: <ul style="list-style-type: none"> • 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 autocomplete list) minus the WHERE clause.

Integrating Object Types and Workflows

This section details the ways in which workflows and object types can integrate to work together.

Integrating Object Type Commands and Workflows

Object type commands are tightly integrated with Mercury IT Governance workflow engine. The commands contained in an object type are executed at execution workflow steps.

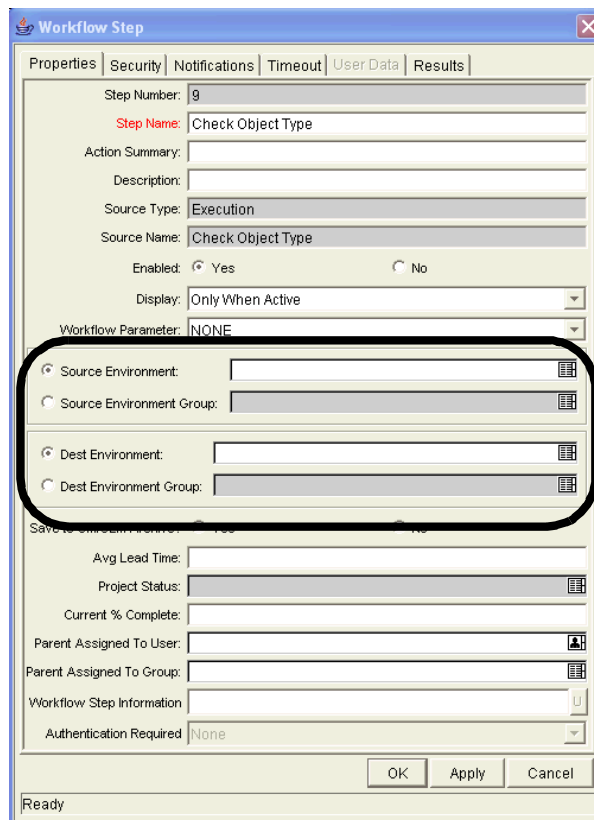
It is important to note the following concepts regarding command and workflow interaction:

- To execute object type commands at a particular workflow step, the workflow step must be configured with the following parameters:
 - Workflow step must be an execution type step.
 - Workflow Scope = **Packages**
 - Execution Type = **Built-in Workflow Event**
 - Workflow Command = **execute_object_commands**
- When the object reaches the workflow step (with Workflow Command = **execute_object_commands**), all object type commands whose conditions are satisfied will be run in the order they are entered in the object type's command panel (**Commands** tab).
- The object type can be configured to run only certain commands at a particular step.
- Each object type command can be configured so that only certain steps (within a command) are executed within a particular workflow step. This is set using conditional statements within the command.

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 Workflow window's **Layout** tab. Select the source and destination environment or environment groups from the fields shown in [Figure 3-20](#).



The screenshot shows the 'Workflow Step' dialog box with the 'Properties' tab selected. The 'Step Name' is 'Check Object Type' and the 'Source Type' is 'Execution'. The 'Source Name' is also 'Check Object Type'. The 'Enabled' checkbox is checked. The 'Display' dropdown is set to 'Only When Active'. The 'Workflow Parameter' is 'NONE'. A black oval highlights the environment selection fields: 'Source Environment', 'Source Environment Group', 'Dest Environment', and 'Dest Environment Group'. Below these are fields for 'Avg Lead Time', 'Project Status', 'Current % Complete', 'Parent Assigned To User', 'Parent Assigned To Group', 'Workflow Step Information', and 'Authentication Required'. The 'Ready' status bar is at the bottom left, and 'OK', 'Apply', and 'Cancel' buttons are at the bottom right.

Figure 3-20. Workflow step window for environments

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 picked 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. Open the Workflow Workbench.

To open the Workflow Workbench, see [Opening the Workflow Workbench on page 63](#). The Workflow Workbench window opens.

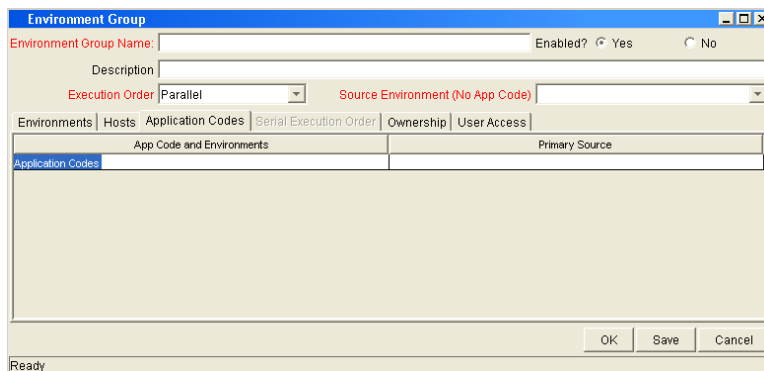
2. From the shortcut bar, select **Environments > Environment Groups**.

The Environments Workbench window opens.

3. Select an environment group or create a new environment group.

The Environment Group window opens.

4. In the Environment Group window, click the **Application Codes** tab.



5. In the **Application Codes** tab, select the Primary Source Environment for each application code.

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

6. In the **Application Codes** tab, click **OK**.

The changes to the environment group are saved.

Integrating Request and Package Workflows

Request (Demand Management) and package workflows (Change Management) can be configured to work together, communicating at key points in the request and package processes. A request workflow step can actually jump to a preselected package workflow step. The package workflow step receives the request workflow step and acts on it to go to the next step in the process.

Packages and requests can also be integrated at a level that does not rely on the workflow configuration. Attach packages and requests to each entity as references. Dependencies can then be set on these reference to control the behavior of the request or package. For example, you can specify that a request is a **Predecessor** to the package. This means the package will not continue until the request closes.

Two built-in workflow events facilitate this cross-product workflow integration. These workflow steps are **wf_jump** and **wf_receive**. Jump workflow step (**wf_jump**) and receive workflow step (**wf_receive**) are used at the points of interaction between workflows. Each jump workflow step must be coupled with a receive workflow step. Workflows can communicate through these jump and receive workflow step pairs.

As an example of when this kind of communication is 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. When the Approval step is successful, 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. Since the step has succeeded, the process can now migrate the code changes to the Production Environment.

This process is graphically represented in [Figure 3-21 on page 118](#).

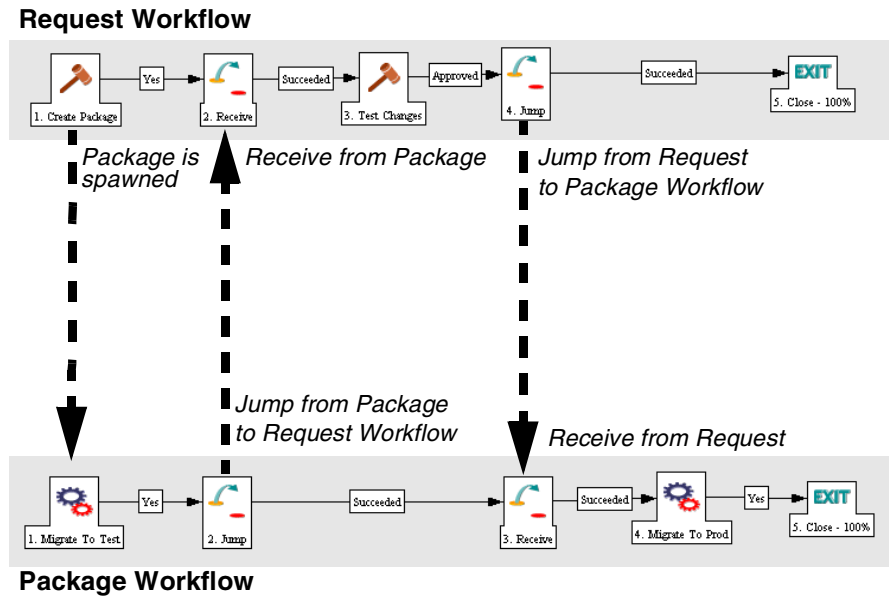


Figure 3-21. Jump/Receive workflow steps

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 for entering into and exiting 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 group 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 [Setting Up WF - Jump/Receive Step Label Validations](#) on page 119.

2. Create a jump workflow step using the **wf_jump** Built-in Workflow Event.

See [Generating Jump Step Sources](#) on page 121.

3. Create a receive workflow step using the **wf_receive** Built in Workflow Event.

See [Generating Receive Step Sources](#) on page 123.

4. Verify that both the jump and receive workflow steps specify the same **WF - Jump/Receive Step Labels**.

See [Including Jump and Receive Workflow Steps in Workflows](#) on page 124.

5. Verify that the transitions exiting the jump workflow step and receive workflow steps match the possible values entering the jump workflow step.

Setting Up WF - Jump/Receive Step Label Validations

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

1. Open the Workflow Workbench.

To open the Workflow Workbench, see [Opening the Workflow Workbench](#) on page 63. The Workflow Workbench window opens.

2. From the shortcut bar, select **Configuration > Validations**.

The Validation Workbench opens.

3. In the **Query** tab of the Validation Workbench, in the Validation Name field enter **WF - Jump/Receive Step Labels**.

4. In the Validation Workbench, click **List**.

5. In the Validation Workbench, click the **Results** tab.

The **WF - Jump/Receive Step Labels** is listed in the **Results** tab.

6. Highlight **WF - Jump/Receive Step Labels** and click **Open**.

The Validation window opens.

Seq	Code	Meaning	Description	Enabled	Default
1	QA_DEV	QA Fix in Development		Y	N
2	QA_PROD	QA Fix in Production		Y	N
3	DEV2QAENV	Migrate to QA		Y	N
4	DEV2TESTRET2REQ	Finish Migration to QA		Y	N
5	DEV2PROD	Migrate to Production		Y	N
6	DEV2PRODRET2R...	Finish Migration to production		Y	N

7. Click **New** to define a new validation value that is used to link two workflows together.

The Add Validation Value window opens.

Value Information | User Data

Code:

Meaning:

Desc:

Enable? Default:

OK Add Cancel

8. In the Add Validation Value window, enter the Code, Meaning and Description.

9. In the Add Validation Value window, click **OK**.

The Add Validation Value window closes. The Validation window is enabled.

10. In the Validation window, click **Ownership** to select which ownership groups will have the ability to edit this validation.

11. In the Validation window, click **OK** to close the Validation window.

The changes to the validation are saved.

The new validation value is now included in the Jump/Receive Step Label drop-down list in the Workflow Step window.

For More Information

For more information concerning configuring validations, see *Commands, Tokens, and Validations Guide and Reference*.

Generating Jump Step Sources

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

1. Open the Workflow Workbench.

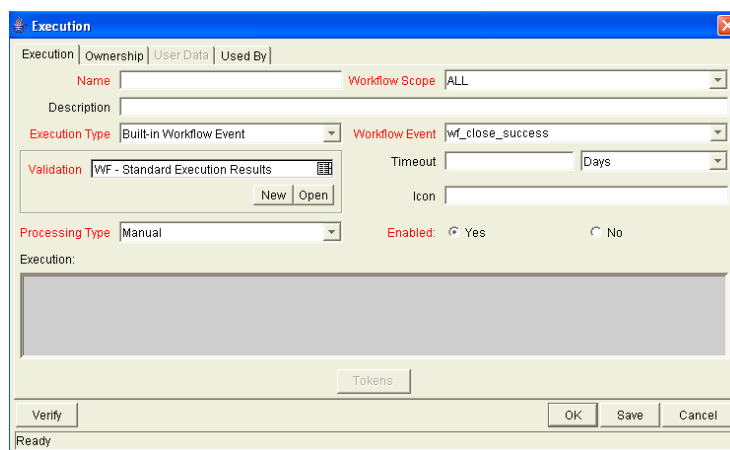
To open the Workflow Workbench, see [Opening the Workflow Workbench on page 63](#). The Workflow Workbench window opens.

2. Open a workflow.

The Workflow window opens.

3. In the Workflow Step Sources window, in the Executions folder, click **New**.

The Execution window opens.



4. Select either **Packages** or **Requests** from the Workflow Scope drop-down list, depending on the desired application of the workflow.

Package level subworkflows and Release Distribution workflows can not include jump and receive steps.

5. In the Execution window, from the Execution Type drop-down list, select **Built-in Workflow Event**.
6. In the Execution window, from the Workflow Event drop-down list, select **wf_jump**.
7. In the Execution window, from the Validation drop-down list, select or create a validation which will be used to transition out of this workflow step.

The validation values exiting the Jump workflow step must match the possible validation values entering the Jump workflow step.

8. In the Execution window, fill in any other required or optional information, such as Name, Description, or Processing Type.
9. In the Execution window, select the **Ownership** tab.
10. Select which Ownership Groups will have the ability to edit this execution workflow step.
11. In the Execution window, click **OK**.

The workflow step is added to the Workflow Step Sources window.

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

Generating Receive Step Sources

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

1. Open the Workflow Workbench.

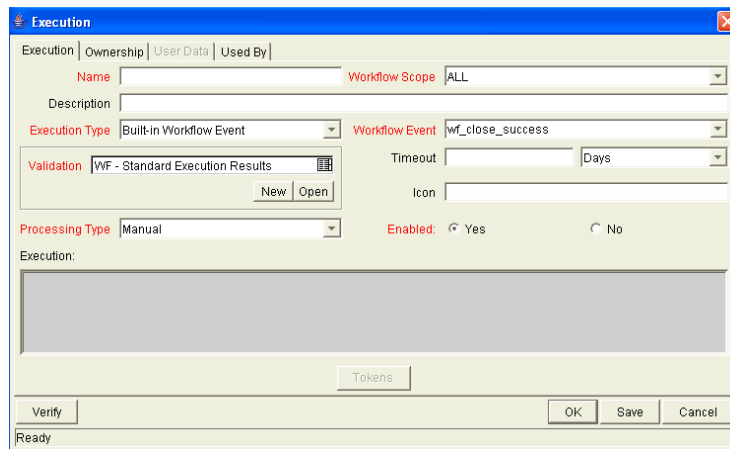
To open the Workflow Workbench, see [Opening the Workflow Workbench on page 63](#). The Workflow Workbench window opens.

2. Open a workflow.

The Workflow window opens.

3. In the Workflow Step Sources window, in the Executions folder, click **New**.

The Execution window opens.



4. In the Execution window, from the Workflow Scope drop-down list, select either **Packages** or **Requests**, depending on the desired application of the workflow.
5. In the Execution window, from the Execution Type drop-down list, select **Built-in Workflow Event**.
6. In the Execution window, from the Workflow Event drop-down list, select **wf_receive**.

7. Select or create a validation which will be used to transition out of this workflow step.

The validation values exiting the Receive workflow step must match the possible validation values entering and exiting the Jump workflow step.

8. In the Execution window, fill in any other required or optional information, such as Name, Description, or Processing Type.
9. In the Execution window, select the **Ownership** tab.
10. Select which Ownership Groups will have the ability to edit this execution workflow step.
11. In the Execution window, click **OK**.

The workflow step is added to the Workflow Step Sources window.

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

Including Jump and Receive Workflow Steps in Workflows

To include a Jump and Receive workflow step in a workflow:

1. Open the Workflow Workbench.

To open the Workflow Workbench, see [Opening the Workflow Workbench on page 63](#). The Workflow Workbench window opens.

2. Open a workflow.

The Workflow window opens.

3. In the Workflow Step Sources window, in the Executions folder, drag and drop either the Jump or Receive step from the Workflow Step Sources window into the workflow's **Layout** tab.

The Workflow Step window opens.

The screenshot shows the 'Workflow Step' configuration window with the following details:

- Step Number:** 9
- Step Name:** Check Object Type (highlighted with a black oval)
- Description:** (empty)
- Source Type:** Execution
- Source Name:** Check Object Type
- Enabled:** Yes No
- Display:** Only When Active
- Workflow Parameter:** NONE
- Source Environment:** (empty)
- Source Environment Group:** (empty)
- Dest Environment:** (empty)
- Dest Environment Group:** (empty)
- Save to OM/GLM Archive?:** Yes No
- Avg Lead Time:** (empty)
- Project Status:** (empty)
- Current % Complete:** (empty)
- Parent Assigned To User:** (empty)
- Parent Assigned To Group:** (empty)
- Workflow Step Information:** (empty)
- Authentication Required:** None

Buttons: OK, Apply, Cancel

4. In the Workflow Step window, select an item from the Jump/Receive Step Label drop-down list.

This item must be the same for a paired jump and receive workflow step. The Jump/Receive Step Label is the key communication link between separate workflows. The communicating jump and receive workflow steps must have a matching Jump/Receive Step Label. It is also important that the Jump/Receive Step Label is unique for any jump and receive pair.

5. In the Workflow Step window, enter any additional workflow step information.
6. In the Workflow Step window, click **OK**.

Repeat this process for the other paired workflow step (jump or receive workflow step), depending on which one you configured first.

Configuring Workflow Components

In This Chapter:

- *Overview of Workflow Step Sources*
 - *Configuring and Using Workflow Step Source Restrictions*
 - *Opening the Workflow Workbench*
 - *Overview of Creating Workflow Step Sources*
 - *Configuring Ownership of Workflow Step Sources*
 - *Creating Decision Workflow Step Sources*
 - *Creating Execution Workflow Step Sources*
 - *Setting Up Execution Steps*
 - *Defining Execution Types*
 - *Creating Subworkflow Workflow Step Sources*
 - *Subworkflows Returning to Change Management Workflows*
 - *Using Workflow Parameters*
 - *Creating Workflow Parameters*
 - *Modifying Workflows Already In Use*
 - *Performance Considerations when Modifying Security*
 - *Performance Considerations when Migrating Workflows*
 - *Copying and Testing Trial Versions of Workflows*
 - *Modifying Production Workflows*
-

Overview of Workflow Step Sources



Note

This chapter covers information concerning Demand Management workflows, Change Management workflows, and Release Management workflows.

Mercury IT Governance Center includes a number of standard workflow step sources that can be added to a workflow. These sources are pre-configured 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:

- The validation associated with the step (and thus the list of valid transition values out of the step).
- The voting requirements of the step.
- The default timeout value for the step. Each step can be configured to have a unique timeout value.
- The 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 Mercury IT Governance Center has a workflow step source that meets the process requirements, simply 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 versus 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:

- A step source that is being used in a workflow can not be deleted.
- A validation for a step source that is being used can not be changed. If the validation needs to change, copy the step source and configure a new validation.
- The workflow step source must be Enabled before it can be added to a workflow.
- Only add step sources to a workflow when the workflow has a matching workflow scope, or the step source has a scope of **All**.
- A workflow step in a workflow that has processed a request, package line, or release can not be deleted. This would compromise data integrity. Instead of deleting the workflow step, remove all transitions to and from the workflow step and disable the workflow step.

Opening the Workflow Workbench

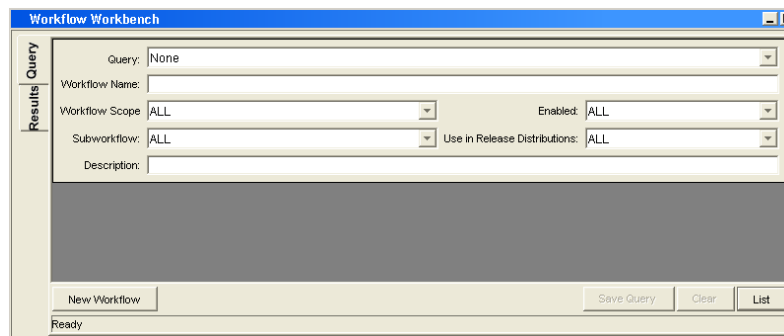
To open the Workflow Workbench:

1. Log on to the Mercury IT Governance Center.
2. From the menu bar, select **Administration > Open Workbench**.
3. A Workbench status window opens. A few minutes later, a Warning - Security window opens.
4. In the Warning - Security window, select **Yes**.

The Workbench opens.

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

The Workflow Workbench window opens.



For More Information

For information on how to search and select an existing workflow, copy a workflow, and delete a workflow, see *Getting Started*.

Overview of Creating Workflow Step Sources

It is possible to 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 153](#)). Condition steps cannot be added to, deleted or modified.

To create a new workflow step source:

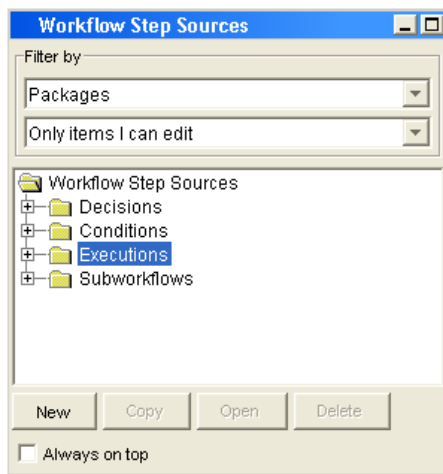
1. Open the Workflow Workbench.

To open the Workflow Workbench, see [Opening the Workflow Workbench on page 130](#). The Workflow Workbench window opens.

2. Open a workflow.

The Workflow window opens.

3. Select the Workflow Step Sources window.

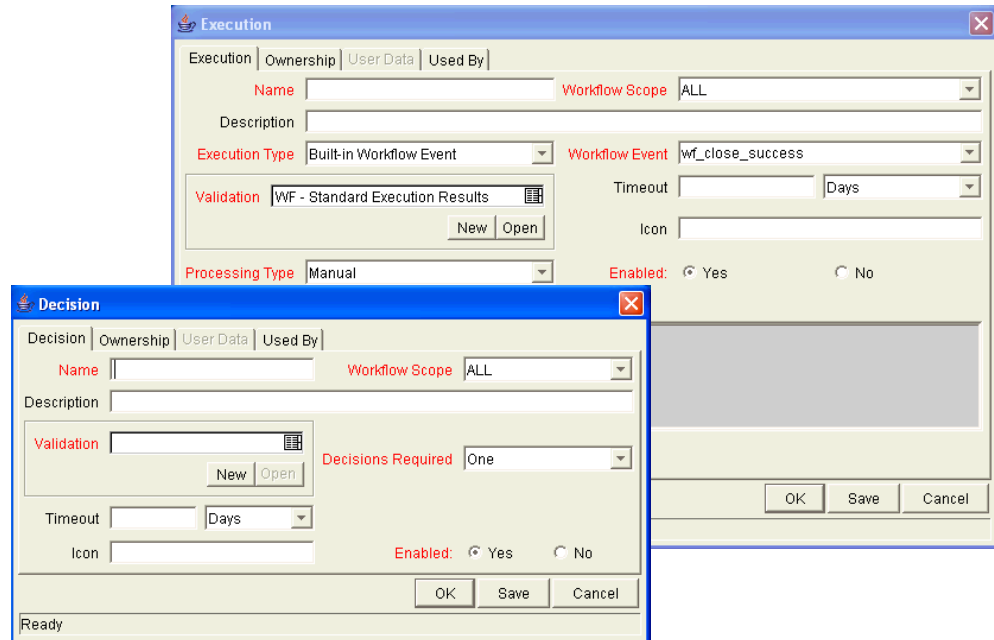


4. In Filter by, select **Requests**, **Packages**, or **Release Distributions**, depending on the type of workflow.

5. Select the Decisions or Executions folder.

6. At the bottom of the Workflow Step Sources window, click **New**.

A window corresponding to the selected workflow step source type opens. For decision workflow steps, the window is Decision. For execution workflow steps, the window is Execution.



7. Enter the required information and any optional information needed to define the workflow step.

For information on configuring a specific workflow step source, see:

- [Creating Decision Workflow Step Sources on page 135](#)
- [Creating Execution Workflow Step Sources on page 139.](#)

8. 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 page 133.](#)

9. In the Decision or Execution window, in the Enabled field, select **Yes**.

10. In the Decision or Execution window, 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.

Configuring Ownership of Workflow Step Sources

When configuring a workflow step source, you can specify who will be able to edit the workflow step source.

To configure the ownership of a new workflow step source:

1. Open the Workflow Workbench.

To open the Workflow Workbench, see [Opening the Workflow Workbench on page 130](#). The Workflow Workbench window opens.

2. Open a workflow.

The Workflow window opens.

3. Open a decision or execution workflow step source.

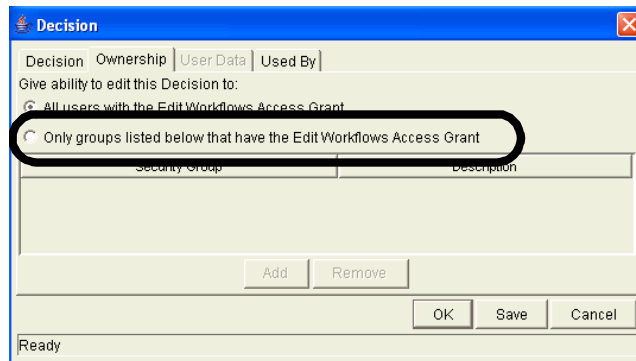
A window corresponding to the selected workflow step source type opens. For decision workflow steps, the window is Decision. For execution workflow steps, the window is Execution.

4. In the Decision or Execution window, click the **Ownership** tab.

The **Ownership** tab opens. Configuring the **Ownership** tab determines which security groups will have the ability to edit this Execution or Decision workflow step. The default is to allow all security groups with the Edit Workflows access grant to edit a workflow step source.

5. In the **Ownership** tab, select Only groups listed below that have the Edit Workflows access grant.

The **Add** button is enabled.



6. In the **Ownership** tab, click **Add**.

The Add Security Group window opens.

7. In the Add Security Group window, in Security Group, select a security group from the autocomplete list.

8. In the Add Security Group window, click **OK**.

The Add Security Group window closes. The security group is added to the workflow step source. The only users who can now edit this workflow step source must belong to a listed security group and the security group must have the Edit Workflow access grant.

9. In the **Ownership** tab, 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 Decision Workflow Step Sources

Before creating a decision workflow step source, check the Decision Step Worksheet. The Decision Step Worksheet contains the information required to properly configure the workflow step source. *Figure 4-1* illustrates the Decision Step Worksheet.

Decision Workflow Step Worksheets

Table A-5. Workflow step [decision], step number ____.

	Value
Step Name	
Goal / Result of Step	
Validation*	
Decisions Required (Vote on Step's outcome?)	<input type="checkbox"/> One <input type="checkbox"/> At Least One <input type="checkbox"/> All
Timeout (Days)	
Security (who can act on step):	
<input type="checkbox"/> Security Group <input type="checkbox"/> User Name <input type="checkbox"/> Standard Token <input type="checkbox"/> User Defined Token	
Include Notification (Yes/No)	
Notification Event	
Notification Recipient:	
<input type="checkbox"/> Username <input type="checkbox"/> Email Address <input type="checkbox"/> Security Group <input type="checkbox"/> Standard Token <input type="checkbox"/> User Defined Token	
Notification Message	
Request Status at Step	
Request % Complete at Step	
Authentication Required (Y/N)	
Authentication Type (if Y)	

Figure 4-1. Information used to create the decision step source.

To create a new decision workflow step source:

1. Open the Workflow Workbench.

To open the Workflow Workbench, see [Opening the Workflow Workbench on page 130](#). The Workflow Workbench window opens.

2. Open a workflow.

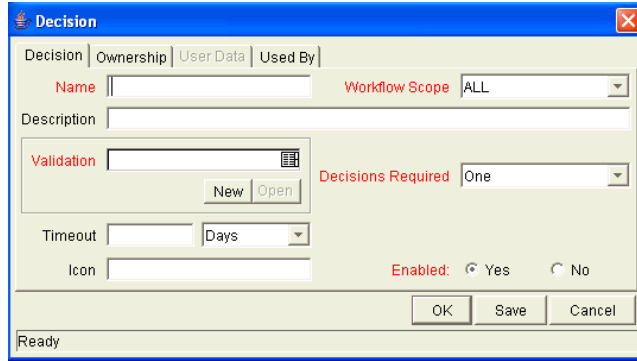
The Workflow window opens.

3. Select the Workflow Step Sources window.

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

5. Select the Decisions folder.
6. At the bottom of the Workflow Step Sources window, click **New**.

The Decision window opens.



7. In the Decision window, make sure you are in the **Decision** tab.

The **Decision** tab is the default tab showing.

8. Complete the fields in the **Decision** tab as specified in the following table:

Field	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 drop-down list to select a workflow scope. The following lists the possible values: <ul style="list-style-type: none"> • ALL. For all workflow types. • Requests. For Mercury Demand Management request workflows. • Packages. For Mercury Change Management package workflows. • Release Distributions. For Mercury Change Management release workflows.
Description	Description of the workflow step source.
Validation	Validations determine the transition values for the workflow step. Use the drop-down list to select a validation.

Field	Description
Decisions Required	<p>Defines the number of decisions required for the workflow step. Use the drop-down list to select a value. The following lists the possible values:</p> <ul style="list-style-type: none"> • One. If One is selected, the workflow step can progress if any one user who is eligible to act on this step makes a decision. • At Least One. If At Least One is selected, the workflow step waits for the voters to vote on this step for a predefined amount of time, designated as the timeout. If all voters mark their decisions before the timeout period, it takes the cumulative decision as the decision for the step and proceeds forward. If any of the voting results differ before the timeout period, the step will immediately result in a No consensus outcome. A timeout period must be defined to use this choice. It is possible to define Specific Errors in workflow steps such as Timeout and No consensus as either Success or Failure in the Define Transition window. If all voters decide on Approve, the final decision is Approve. If all voters decide on Not Approved, the final decision is Not Approved. If some voters decide on Approved and one voter decides on Not Approved, the result is 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 will be used. If at the end of the Timeout no one has voted, the step will result in a Timeout. • All. If All is selected, the workflow step waits for all of the voters to vote. This workflow step is used along with a specified timeout period. Selecting All makes it mandatory for all voters to vote on the workflow step. The workflow step waits until the timeout period for the voters to vote. If all voters vote, the cumulative 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. 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	<p>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).</p> <p>If this workflow step remains eligible for the value entered in the timeout value, the request, package, or release can be configured to send an appropriate notification. This field is often used in conjunction with the At Least One and All settings for Decisions Required.</p> <p>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.</p>

Field	Description
Icon	<p>A different graphic can be specified to represent steps of this source for use in the workflow Layout tab.</p> <p>The graphic needs to exist in the <code>icons</code> subdirectory in the Mercury IT Governance Server. All icons are in <code>.gif</code> format.</p>
Enabled	<p>The workflow step source must be enabled in order to add the workflow step to the workflow layout.</p>

9. In the Decision window, 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 with the Edit Workflows access grant 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 133](#).

10. In the Decision window, 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. In the Decision window, click the **Used By** tab.

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

12. At the bottom of the Decision window, 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, check the Execution Step Worksheet. The Execution Step Worksheet contains the information required to properly configure the workflow step source. *Figure 4-2* illustrates the Execution Step Worksheet.

Execution Workflow Step Worksheets

Table A-2. Workflow step [execution], step number ____.

	Value
Step Name	
Goal / Result of Step	
Validation*	
Execution Type**	
Processing Type	
Timeout (Days)	
Source Environment (Group)	
Dest Environment (Group)	
Security (who can act on step):	
☞ 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-3. Workflow step [execution], step number ____ validation.

Validation Information*	Value
Existing Validation?	
New Validation?	
Validation Type: (text field, autocomplete, dropdown list, etc.)	
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	
PL/SQL Function	
Token	
SQL Statement	
Workflow step commands	

Figure 4-2. Information used to create the execution step source.

To create a new execution workflow step source:

1. Open the Workflow Workbench.

To open the Workflow Workbench, see [Opening the Workflow Workbench on page 130](#). The Workflow Workbench window opens.

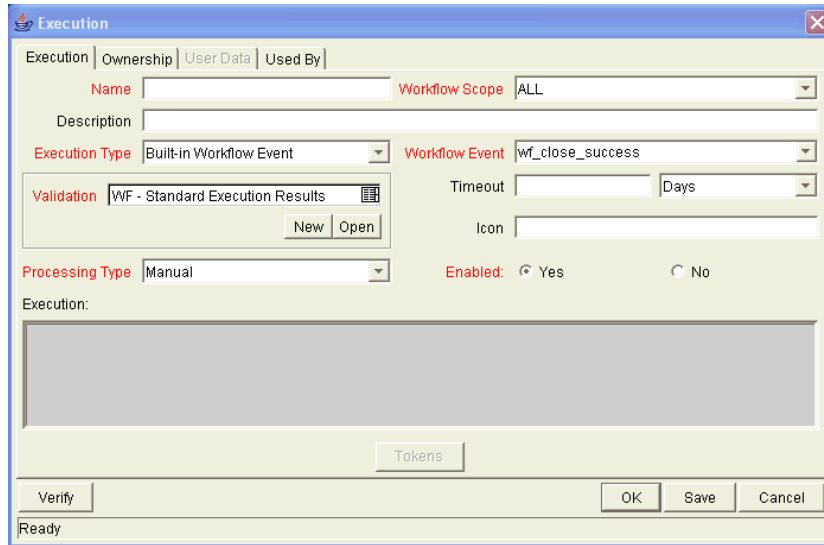
2. Open a workflow.

The Workflow window opens.

3. Select the Workflow Step Sources window.

4. In Filter by, select **Requests**, **Packages**, or **Release Distributions**, depending on the type of workflow.
5. Select the Executions folder.
6. At the bottom of the Workflow Step Sources window, click **New**.

The Execution window opens.



7. In the Execution window, make sure you are in the **Execution** tab.

The **Execution** tab is the default tab showing.

8. Complete the fields in the **Execution** tab as specified in the following table:

Field	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 drop-down list to select a workflow scope. The following lists the possible values: <ul style="list-style-type: none"> ● ALL. For all workflow types. ● Requests. For Mercury Demand Management request workflows. ● Packages. For Mercury Change Management package workflows. ● Release Distributions. For Mercury Change Management release workflows.
Description	Description of the step source.

Field	Description
Execution Type	<p>Used to select the type of execution to be performed. Use the drop-down list to select an execution type. The following lists the possible values:</p> <ul style="list-style-type: none"> ● Built-in Workflow Event. Executes a predefined command and returns its result as the result of the step. ● SQL Statement. Executes a 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. Executes a set of commands, independent of an object, at a workflow step.
Workflow Event	<p>For Execution Type Built-in Workflow Event, the specific event to perform must be selected. The available choices in the drop-down list depend on the workflow scope selected. The choices include:</p> <ul style="list-style-type: none"> ● execute_object_commands. Executes the object type commands for a package line. ● execute_request_commands. Executes the request type commands for a request. ● create_package. Generates a Mercury Change Management package. ● rm_ready_for_release. Generates a Mercury Demand Management request. ● create_package_and_wait. Generates a Mercury Change 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. (Mercury Change Management and Mercury Demand Management) Instructs the workflow to proceed to a corresponding Receive Workflow Step in another workflow. ● wf_receive. (Mercury Change Management and Mercury 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. (Mercury Change Management and Mercury Demand Management) Used to route a subworkflow process back to its parent workflow.
PL/SQL Function	<p>For Execution Type PL/SQL Function, the actual function to run. The results of the function will determine the outcome of the step.</p> <p>The results of the function must be a subset of the validation values for that workflow step.</p>

Field	Description
Token	For Execution Type Token , the token that will be resolved. The results of the token resolution will determine the outcome of the workflow step.
SQL Statement	For Execution Type SQL Statement , the actual query to run. The results of the query will 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 will result with a Succeeded or Failed value. Use a validation with those values to enable transitioning out of the step based on the execution results.
Processing Type	Defines when the execution is performed. Use the drop-down list to select a processing type. The following lists the possible values: <ul style="list-style-type: none"> • Immediate executes the workflow step when the workflow step becomes eligible. • Manual executes the workflow step manually by a user.
Validation	Validations determine the transition values for the workflow step. Use the drop-down list to select a validation.
Timeout	The amount of time that a step is eligible before completing with an error. Timeouts can be by minute, hour, weekday or week. 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 entered in the timeout value, the request, package line, or release can be configured to send an appropriate notification. 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 also 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 <code>icons</code> subdirectory in the Mercury IT Governance server. All icons are in <code>.gif</code> format.
Enabled	The workflow step source must be enabled in order to add it to the workflow layout.

9. In the Execution window, 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 with the Edit Workflows access grant 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 133](#).

10. In the Execution window, 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. In the Execution window, click the **Used By** tab.

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

12. At the bottom of the Execution window, 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 will not be able to proceed.

Defining Execution Types

Execution workflow steps are used to perform specific actions. Mercury Change Management provides a number of 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 execute 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.

Mercury Change Management includes the execution workflow step source **DLV Execution w/ Reset** that executes 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 execution workflow step source **DLV Execution w/ Reset** and changes the field values as defined in [Table 4-1 on page 144](#).

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

Field in Execution Window	Value
Name	Enter 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 Success

It is possible to create an execution workflow step that close a package line and marks the package line as **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.

Mercury Change 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 execution workflow step source **Close (Immediate success)** or **Close (Manual success)** and changes the field values as defined in [Table 4-2 on page 145](#).

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

Field in Execution Window	Value
Name	Enter 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.

Mercury Change 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 execution workflow step source **Close (Immediate failure)** and changes the field values as defined in [Table 4-3 on page 146](#).

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

Field in Execution Window	Value
Name	Enter 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 Release Management.

Mercury Change 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 execution workflow step source **Ready for Release** and changes the field values as defined in [Table 4-4](#).

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

Field in Execution Window	Value
Name	Enter 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 the 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](#).

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

Field in Execution Window	Value
Name	Enter 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 of 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	Enter the SQL query.
Enabled	Yes

Executing SQL Statements and Creating Transitions Based on the 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 a 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 as defined in [Table 4-6](#).

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

Field in Execution Window	Value
Name	Enter a descriptive name for the workflow step source.
Workflow Scope	Packages

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

Field in Execution Window	Value
Execution Type	SQL Statement
Processing Type	Manual or Immediate
Validation	Selects or creates a validation that includes all of 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	Enter the SQL query.
Enabled	Yes

Evaluating Tokens and Creating Transitions Based on the Results

Mercury Demand 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](#).

Table 4-7. Execution window values for evaluating tokens

Field in Execution Window	Value
Name	Enter 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	Enter 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.

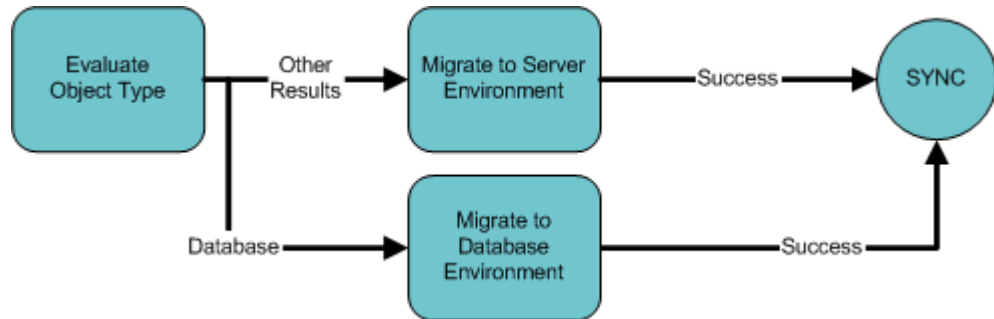


Figure 4-3. Transitioning based on a token

IT decides to use an execution workflow step to automatically evaluate the type of object and route the package line accordingly. To accomplish this, an execution workflow step source, **Evaluate Object Type**, has been configured with the parameters listed in [Table 4-8](#).

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

Field in Execution Window	Value
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 execute at this step, the commands will 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.

A subworkflow can be selected from the Workflow Step Sources window and dragged onto the **Layout** tab. When the package, request, or release reaches the subworkflow step, it follows the path defined in that subworkflow. The subworkflow will either close within that workflow or return 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:

- The Workflow window contains a Sub-workflow radio button which should be set to **Yes**.
- The validation for the step leaving the subworkflow layout should match the subworkflow step in the parent workflow.

Subworkflows Returning to Change Management Workflows

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

Mercury Change Management includes the execution workflow step source **Return from Subworkflow** 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 execution workflow step source **Return from Subworkflow** and changes the field values as defined in [Table 4-9](#).

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

Field in Execution Window	Value
Name	Enter 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 another existing or create a new validation.)
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:

- Workflow parameters can be referenced using the WF.P token prefix.
- Workflow parameters can be used in PL/SQL and SQL workflow step executions.

Creating Workflow Parameters

To create a workflow parameter:

1. Open the Workflow Workbench.

To open the Workflow Workbench, see [Opening the Workflow Workbench on page 130](#). The Workflow Workbench window opens.

2. Open a workflow.

The Workflow window opens.

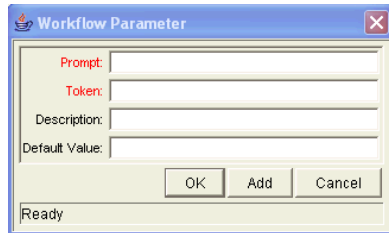
3. In the Workflow window, make sure you are in the **Workflow** tab.

The **Workflow** tab is the default tab opened.

The screenshot shows the 'Workflow: ExampleWorkflow' window. The 'Workflow' tab is selected. The 'Name' field contains 'Example Workflow' and the 'Workflow Scope' is set to 'Packages'. The 'Description' field is empty. The 'Enabled' checkbox is checked. The 'First Step' dropdown is set to 'Review Request'. The 'Reopen Step' dropdown is empty. The 'Subworkflows' section has 'Subworkflow' unchecked and 'Use in Release Distributions' checked. The 'Validation' field is empty with 'New' and 'Open' buttons. The 'Icon Name' field is empty. The 'Parameters' section is a table with columns for Prompt, Token, Description, and Default Value. At the bottom, there are 'Add', 'Edit', and 'Remove' buttons, and a 'Verify' button on the left and 'OK', 'Save', and 'Cancel' buttons on the right. The status bar at the bottom left says 'Ready'.

4. In the **Workflow** tab, click **Add**.

The Workflow Parameter window opens.



5. Complete the fields in the Workflow Parameter window as specified in the following table:

Field	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.

6. In the Parameters section of the **Workflow** tab, click **Add**.
7. The Workflow Parameter window, click **OK**.
8. The Workflow Parameter window closes. The workflow parameter appears in the Parameters section of the **Workflow** tab.
9. In the **Workflow** tab, click **OK**.

The changes to the workflow are saved.

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. Open the Workflow Workbench.

To open the Workflow Workbench, see [Opening the Workflow Workbench on page 130](#). The Workflow Workbench window opens.

2. Open a workflow.

The Workflow window opens.

3. In the Workflow window, make sure you are in the **Workflow** tab.

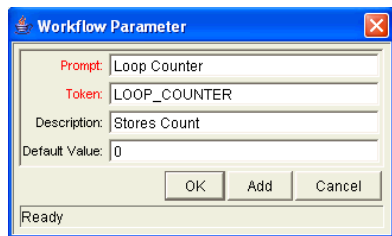
The **Workflow** tab is the default tab opened.

4. In the **Workflow** tab, click **Add**.

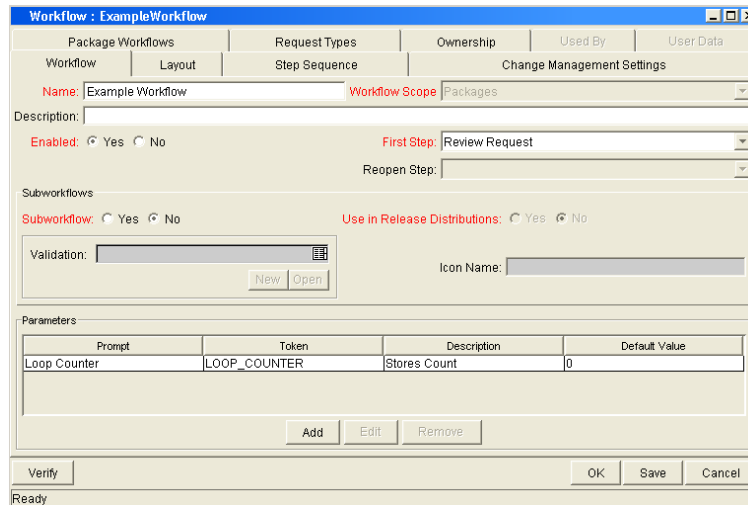
The Workflow Parameter window opens.

5. Complete the fields in the Workflow Parameter window as specified in the following table:

Field	Description
Prompt	Loop Counter
Token	LOOP_COUNTER
Description	Stores count.
Default Value	0



6. In the Parameters section of the **Workflow** tab, click **Add**.
7. The Workflow Parameter window, click **OK**.
8. The Workflow Parameter window closes. The workflow parameter appears in the Parameters section of the **Workflow** tab.



9. In the **Workflow** tab, click **OK**.

The changes to the workflow are saved.

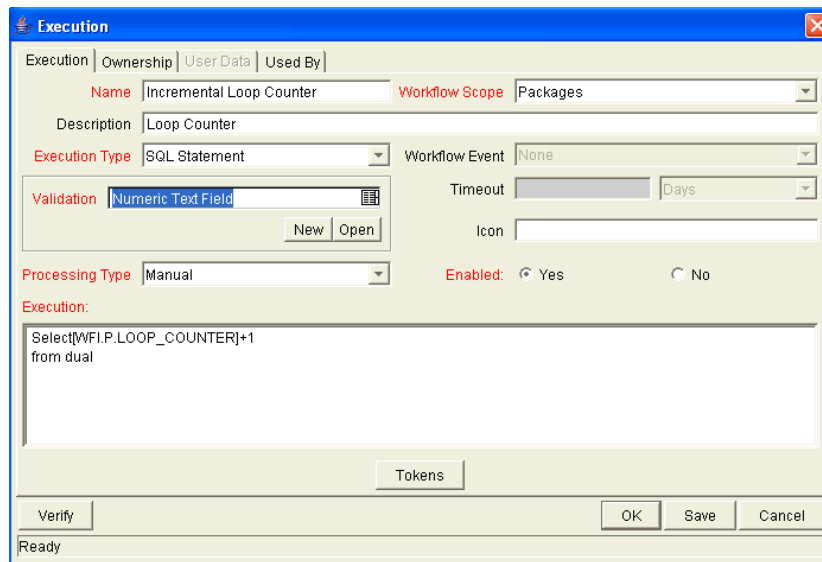
10. Create a new immediate SQL execution workflow step.

For details on how to create an SQL execution workflow step, see [Creating Execution Workflow Step Sources](#) on page 139.

There are two key concepts to note about the new step definition.

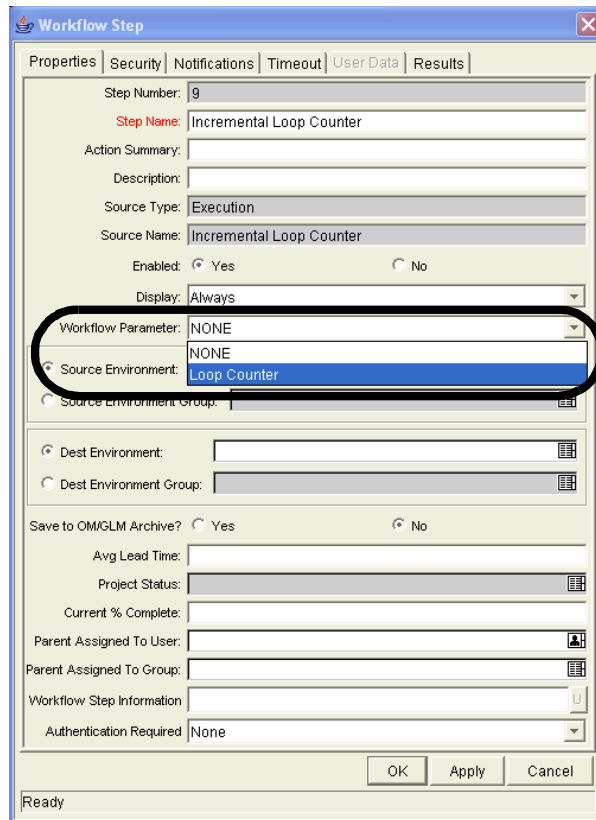
- The result of the SQL execution workflow step returns the result **LOOP_COUNTER + 1**. This return value is linked back into the parameter when the workflow step is generated on a workflow.
- A validation for a **Numeric** text field is used. This allows \leq , $<$, \geq , and $>$ comparisons to be used in transitions off this step.

The following illustrates the Execution window for the SQL execution workflow step.



11. 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.



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

It is possible to make changes to workflows 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](#).

Table 4-10. Rules for modifying production workflows

Entity	Procedure
Transitions Security Notifications Workflow Steps Workflow Parameters	All of these entities can be modified or added to a workflow in use.
Transitions Security Notifications Workflow Parameters	All of these entities can be deleted from a workflow in use.
Workflow Steps	This entity cannot be deleted from a workflow in use, but can be renamed. Transitions coming into or going out of a workflow step can be deleted, effectively removing it from the workflow.

When a workflow that is in use is modified 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 when Modifying Security

Updating an existing workflow step's security with a specific configuration can impact system performance. When adding dynamic security to a step, such as based on a standard or user defined token, in the Workflow Step window in the **Layout** tab, product database tables are updated to handle this new configuration. Due to the scope of these database changes, Database Statistics need to be re-run on your database.

Instructions for this operation are included in the *System Administration Guide and Reference*. Contact your application administrator for help with this procedure.

This also applies when migrating a workflow with these types of changes into an instance of the Mercury IT Governance Center.

Performance Considerations when Migrating Workflows

Migrating a workflow with these types of changes into an instance of the Mercury IT Governance Center can impact system performance. Product database tables must be updated to handle this new workflow. Due to the scope of these database changes, Database Statistics need to be re-run on your database.

Instructions for this operation are included in the *System Administration Guide and Reference*. Contact your application administrator for help with this procedure.

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.
5. 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](#), a step can not 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.

To disable a and add a new step:

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. Since 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 shown in [Figure 4-4](#).

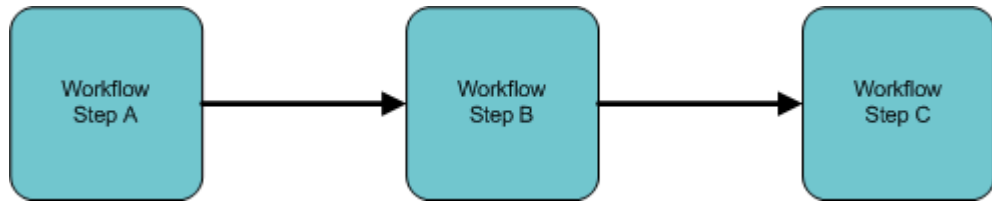


Figure 4-4. Redirecting the workflow, step 1

After removing the incoming and outgoing transitions to B, add a new workflow step D which would connect steps A and C and let the workflow continue to process requests or package lines (see [Figure 4-5](#)).

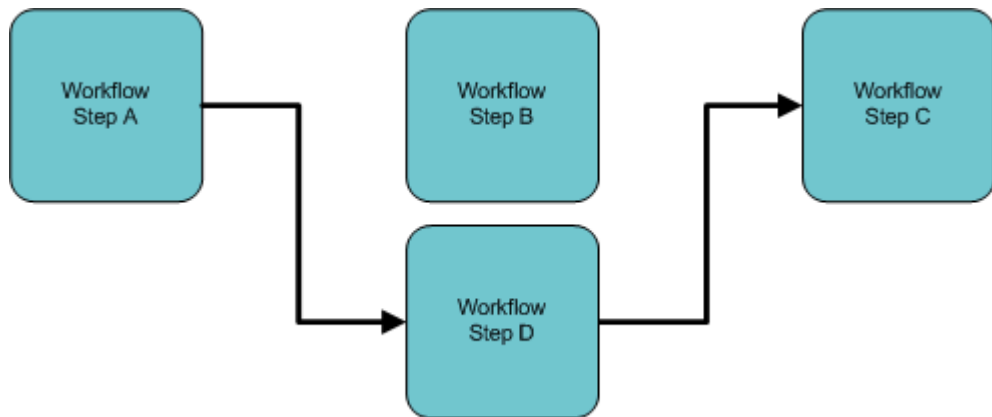


Figure 4-5. Redirecting the workflow, step 2

Run the appropriate report(s) 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

In This Chapter:

- *Overview of Object Types*
- *Opening the Object Type Workbench*
- *Configuring General Information for Object Types*
- *Creating Object Type Fields*
 - *Overview of Object Type Field Validations*
 - *Creating Object Type Fields*
 - *Configuring Field Dependencies*
 - *Copying Object Type Fields*
 - *Editing Object Type Fields*
 - *Removing Fields*
- *Configuring Layouts for Object Types*
 - *Changing Field Widths*
 - *Moving Fields*
 - *Setting Object Names*
 - *Setting Object Revisions*
- *Configuring Commands for Object Types*
 - *Adding Commands to Object Types*
 - *Editing Commands of Object Types*
 - *Copying Commands in Object Types*
 - *Deleting Commands in Object Types*
 - *Command Conditions*
- *Configuring Ownership for Object Types*
 - *Adding Ownerships to Object Types*
 - *Deleting Ownerships from Object Types*
 - *Using Commands to Change Field Values*

Overview of Object Types

Mercury Change Management automates complex software deployment processes. While Mercury IT Governance 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 a SQL Script object type might address the migration and execution of database scripts.

Object types are used by users who create and process packages. Each package line in a package consists of one object of a specific object type. When defining a package line, the user will select an object type in the Add Line window in the package screen (see [Figure 5-1](#)). Fields dynamically appear that are required to process that type of object.

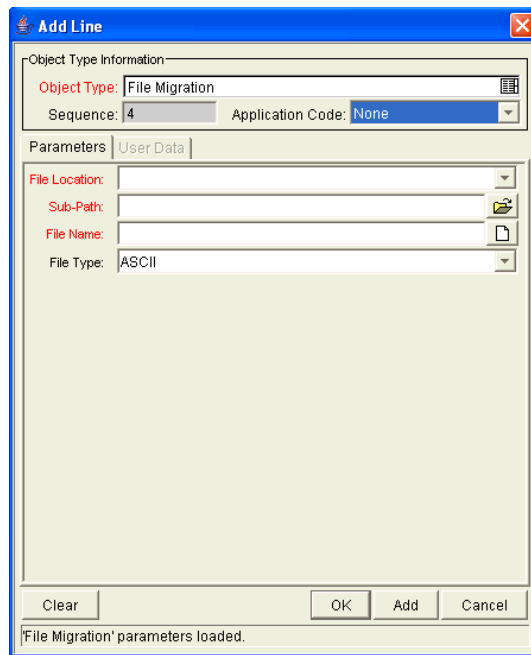


Figure 5-1. Example of an object type

Object types are created and configured in the Object Type window (see [Figure 5-2](#)).

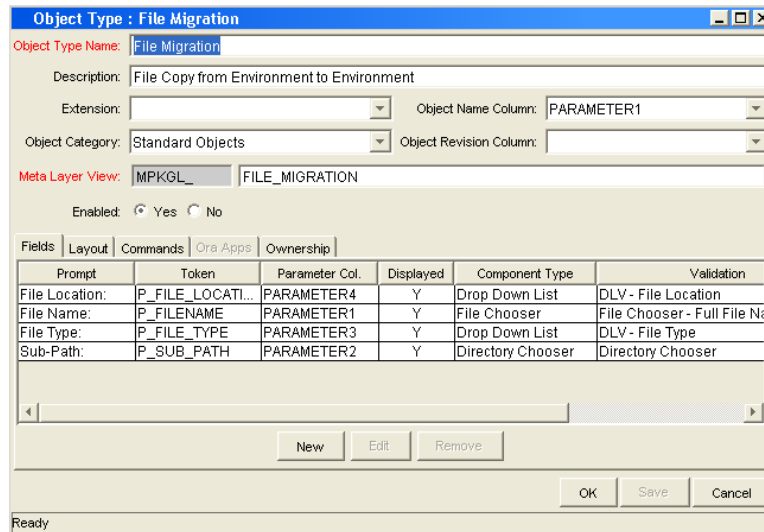


Figure 5-2. Object Type window

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

- **General information.** General information includes 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 page 171](#).
- **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 172](#).
- **Layout.** Once all of the fields are created for a object type, the layout of those fields can be configured using the **Layout** tab. See [Configuring Layouts for Object Types on page 183](#).
- **Commands.** Commands can also be used to control certain behavior of object type fields. At specific workflow execution steps in a deployment process, it is possible to select to run the commands stored in the object type. These commands can then manipulate the data inside a object type field. This provides an advantage over the defaulting features in the **Field** tab, which can only default based on a single parameter stored on the same object type. See [Configuring Commands for Object Types on page 189](#).
- **OraApps.** Mercury Change Management Extensions for Oracle E-Business Suites™ requires specially configured object types. If Mercury Change Management Extensions for Oracle E-Business Suites is not installed, the **OraApps** tab is disabled.

- **Ownership.** Configure who can edit the object type. See [Configuring Ownership for Object Types](#) on page 195.

Opening the Object Type Workbench

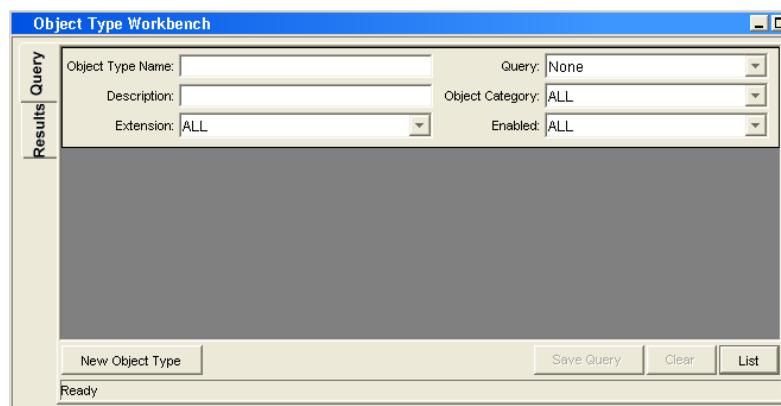
To open the Object Type Workbench:

1. Log on to the Mercury IT Governance Center.
2. From the menu bar, select **Administration > Open Workbench**.
3. A Workbench status window opens. A few minutes later, a Warning - Security window opens.
4. In the Warning - Security window, select **Yes**.

The Workbench opens.

5. From the shortcut bar, select **Change Mgmt > Object Types**.

The Object Type Workbench window opens.



For More Information

For information on how to search and select an existing object type, copy an object type, and delete an object type, see *Getting Started*.

Configuring General Information for Object Types

To configure the general information of an object type:

1. Open the Object Type Workbench.

To open the Object Type Workbench, see [Opening the Object Type Workbench on page 170](#). The Object Type Workbench window opens.

2. Open an object type.

The Object Type window opens.

3. Complete the fields in the Object Type window as specified in the following table:

Field	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 a Mercury Change Management extension. Select the extension from the drop-down list.
Object Category	Object categories provide a way of categorizing object types in Mercury Change Management. The default values are Custom Object 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 shout 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 187 .

Field	Description
Object Revision Column	<p>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.</p> <p>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.</p> <p>Typically, selecting an entry for the Object Revision Column is done after all of the object type fields are created. For more information, see Setting Object Names on page 187.</p>
Meta Layer View	Meta layer views relate information specific Mercury IT Governance Center.
Enabled	Indicates whether or not the object type is available to Mercury IT Governance Center. Selecting Yes makes the object type available to the system.

4. Save the changes to the object type.

Click **OK** to save the changes and close the Object Type window. Click **Save** to save the changes and leave the Object Type window open. Click **Cancel** to lose the changes and close the Object Type window.

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. The Field window contains three tabs:

- **Attributes tab.** The **Attributes** tab is used to set basic display, edit, and requirement field properties.
- **Default tab.** The **Default** tab is used to set the value in the field.
- **Dependencies tab.** The **Dependencies** tab is used 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 entered in the field and the field type (such as text field, drop-down list, or date field).

For example, ACME requires a File Type field to describe the objects to be deployed. On their object type, they add a field as illustrated in the following figure (*Figure 5-3*).

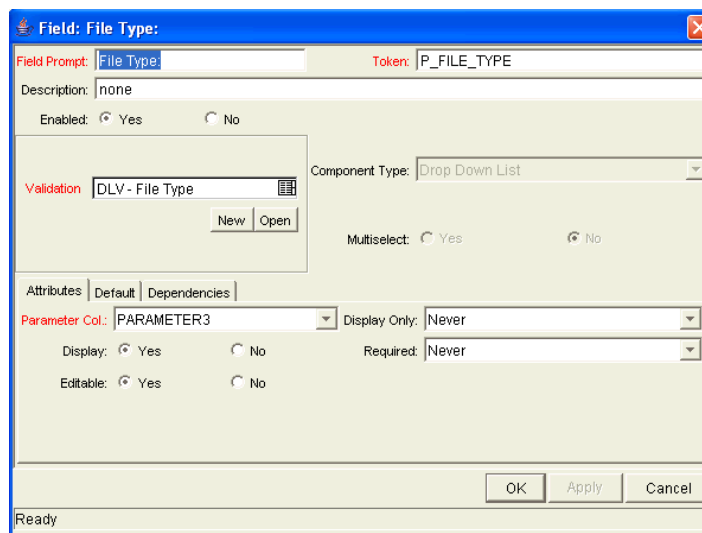


Figure 5-3. Field window

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

For More Information

For more information concerning validations, see *Commands, Tokens, and Validations Guide and Reference*.

Selecting Validations

If a validation does not exist that meets the necessary requirements (such as having the appropriate values), you will need to create one. You can also select a validation that has been configured for use at your site.

Be careful when using a validation that has been configured for use in another process. If the owner of the other process changes the validation, it will also be changed for the items in your process. Consider creating a new validation by copying the existing one. You can then control who can alter the validation values by setting ownership on that validation.

Creating Object Type Fields

To create an object type field:

1. Open the Object Type Workbench.

To open the Object Type Workbench, see [Opening the Object Type Workbench on page 170](#). The Object Type Workbench window opens.

2. Open an object type.

The Object Type window opens.

3. In the Object Type window, click **New**.

The Field window opens.

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

Field	Description
Field Prompt	Enter the name of the new field.
Token	Enter the name of the token. Tokens must have unique names. Make sure the name you choose for this token is unique.
Description	Enter an explanation of the new field and how it works.
Enabled	Make the field available to the system. Select Yes to make the field available to the system.

5. In the Field window, in Validation, select a validation from the auto-complete list.

See [Overview of Object Type Field Validations on page 173](#) for information regarding the choosing of a validation. If a validation does not exist that meets the necessary requirements, one must be created. See *Commands, Tokens, and Validations Guide and Reference* for details on how to create a validation.

6. Configure the field’s behavior.

This consists of setting options in the field’s **Attributes**, **Default**, and **Dependencies** tabs. See [Creating Object Type Fields on page 172](#). Note that some field behavior is dependent on other object type fields. This step may need to be revisited after creating the other fields in the object type.

To configure the field’s behavior:

- a. Complete the fields in the **Attributes** tab as specified in the following table:

Field	Description
Parameter Col.	<p>Indicates the internal database column that the field value will be stored in. These values are then stored in the corresponding column in the package lines table for each Line of the given object type.</p> <p>Information can be stored in up to 30 columns and thus allow up to 30 fields/parameters. No two fields in an object type can use the same column.</p>
Display Only	<p>Indicates whether a field should be displayed using the following options: Always, Never or Use Dependency Rules. Select Use Dependency Rules to use the logic defined in the Dependencies tab.</p> <p>Display Only: Always means that the field is not editable.</p> <p>Display Only: Never means that the field is always editable.</p>
Display	<p>Indicates if this field is visible in the package line region of the package window.</p>
Required	<p>Indicates if a value needs to be specified for this field using the following options: Always, Never, or Use Dependency Rules. Select Use Dependency Rules to use the logic defined in the Dependencies tab.</p>
Updateable	<p>After a package line has been entered 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 updateable once package lines of File object type start getting processed.</p>

b. Complete the fields in the **Default** tab as specified in the following table:

Field	Description
Default Type	Defines if the field will have a default value. Either 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, the constant value can be entered here. This value should be what the user would normally enter in the field.
Depends On	If defaulting from another field, enter the token name of that field. At runtime, when using this object type, every time a value is entered or updated in the source field, it will automatically be entered or updated in this destination field.

c. Complete the fields in the **Dependencies** tab as specified in the following table:

Field	Description
Clear When _ __ 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. This field functions with two adjacent fields: a drop-down list containing the logical qualifier and a text field. To use this functionality, select Use Dependency Rules from the first drop-down list.
Required When	Indicates that the current field should be required when certain logical criteria are satisfied. This field functions with two adjacent fields: a drop-down list containing logical qualifier and a text field. To use this functionality, select Use Dependency Rules from the first drop-down list.

7. At the bottom of the Field window, 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 the following table ([Table 5-1](#)).
- Become required when another field meets a logical condition defined in [Table 5-1](#).

Table 5-1. Field dependencies

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

To configure a field dependency:

1. Open the Object Type Workbench.

To open the Object Type Workbench, see [Opening the Object Type Workbench on page 170](#). 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. In the Field window, click the **Dependencies** tab.

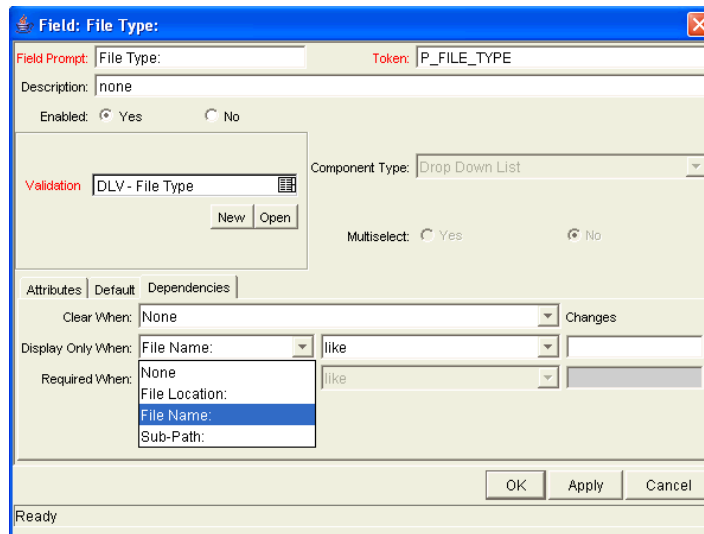
5. In the **Dependencies** tab, set the field dependencies, using one of the following options:

- From the Clear When drop-down list, select a field name to indicate that the current field should be cleared when the selected field changes.
- From the Display Only When drop-down list, select a field name to indicate that the current field should not be editable when certain logical criteria are satisfied.

This field functions with two adjacent fields. These fields are a drop-down list containing logical qualifier, and a field which dynamically changes to a date field, drop-down list, or text field, depending on the selected field's validation.

- From the Required When drop-down list, select a field name to indicate that the current field should be required when certain logical criteria are satisfied.

This field functions with two adjacent fields. These fields are a drop-down list containing logical qualifier, and a field which dynamically changes to a date field, drop-down list, or text field, depending on the selected field's validation.



6. In the Field window, click **OK**.

This adds the field dependencies to the **Fields** tab of the Object Type window and closes the Field: New window.

7. In the Field tab, click **OK**.

The changes to the object type are saved.

Copying Object Type Fields

To copy an object type field:

1. Open the Object Type Workbench.

To open the Object Type Workbench, see [Opening the Object Type Workbench on page 170](#). The Object Type Workbench window opens.

2. Open an object type.

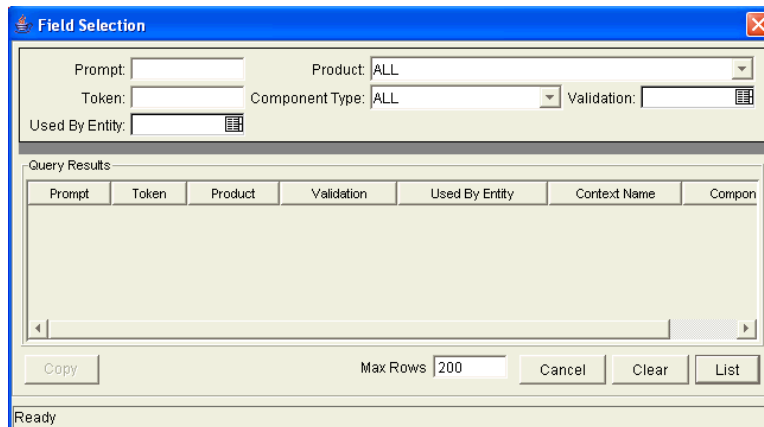
The Object Type window opens. The **Fields** tab is the default tab displayed.

3. In the **Fields** tab, click **New**.

The Field window opens.

- In the Field window, click **Copy From**.

The Field Selection window opens.



- In the Field Selection window, query for the field to be copied.

Fields can be queried by a number of criteria, such as the token name or field prompt. More complex queries can also be performed, such as listing all fields that reference a certain validation or are used by a certain entity. Due to the large number of Mercury IT Governance Center fields, limit the list of fields by one or more of the query criteria.

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

This closes the Field Selection window and copies the definition of the selected field into the New Field window.

- In the New Field window, modify the fields as required.

- In the New Field window, click **OK**.

The New Field window closes. The new field is added to the **Fields** tab.

- In the **Fields** tab, click **OK**.

The changes to the object type are saved.

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 in 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. Open the Object Type Workbench.

To open the Object Type Workbench, see [Opening the Object Type Workbench on page 170](#). The Object Type Workbench window opens.

2. Open an object type.

The Object Type window opens.

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

The Field window opens.

4. Modify the field as required and click **OK**.

The Fields window closes. The changes to the field appear in the **Fields** tab.

5. In the **Fields** tab, click **OK**.

The changes to the object type are saved.

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. Open the Object Type Workbench.

To open the Object Type Workbench, see [Opening the Object Type Workbench on page 170](#). The Object Type Workbench window opens.

2. Open an object type.

The Object Type window opens.

3. In the **Fields** tab, select a field and click **Remove**.

The field is removed.

4. In the **Fields** tab, click **OK**.

The changes to the object type are saved.

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. [Figure 5-4 on page 184](#) illustrates the **Layout** tab of an object type and what the object type looks like.

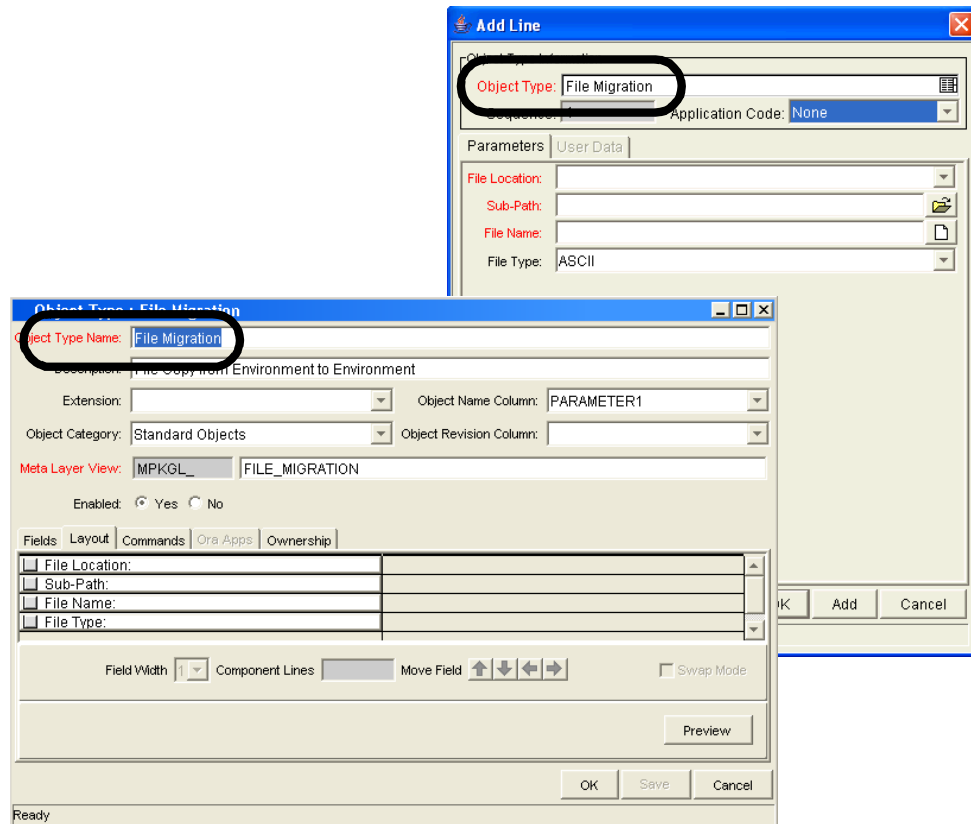


Figure 5-4. Example of an object type and Layout tab

Changing Field Widths

To change the width of an object type field:

1. Open the Object Type Workbench.

To open the Object Type Workbench, see [Opening the Object Type Workbench on page 170](#). The Object Type Workbench window opens.

2. Open an object type.

The Object Type window opens.

3. In the Object Type window, click the **Layout** tab.

The **Layout** tab opens.

4. In the **Layout** tab, select a field.
5. From the Field Width drop-down list, select either **1** or **2**.

The Layout editor will not allow changes to be made if the change conflicts with another field in the layout (for example, a field's width cannot be changed from one to two if another field exists in column two on the same row).

Additionally, for fields of component type Text Area, the number of lines the text area displays can be determined 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 will be blank and non-updatable.

The screenshot shows the 'Object Type : File Migration' window. The 'Object Type Name' is 'File Migration' and the 'Description' is 'File Copy from Environment to Environment'. The 'Extension' is blank, and the 'Object Name Column' is 'PARAMETER1'. The 'Object Category' is 'Standard Objects' and the 'Object Revision Column' is blank. The 'Meta Layer View' is 'MPKGL_ FILE_MIGRATION'. The 'Enabled' checkbox is checked. The 'Layout' tab is selected, showing a list of fields: 'File Location', 'Sub-Path', 'File Name', and 'File Type'. The 'Sub-Path' field is selected. Below the list, the 'Field Width' is set to '1', 'Component Lines' is blank, and there are 'Move Field' buttons (up, down, left, right) and a 'Swap Mode' checkbox. A 'Preview' button is also present. At the bottom, there are 'OK', 'Save', and 'Cancel' buttons. The status bar at the bottom left says 'Ready'.

6. In the **Layout** tab, click **OK**.

The changes to the object type are saved.

Moving Fields

To move an object type field:

1. Open the Object Type Workbench.

To open the Object Type Workbench, see [Opening the Object Type Workbench on page 170](#). The Object Type Workbench window opens.

2. Open an object type.

The Object Type window opens.

3. In the Object Type window, click the **Layout** tab.

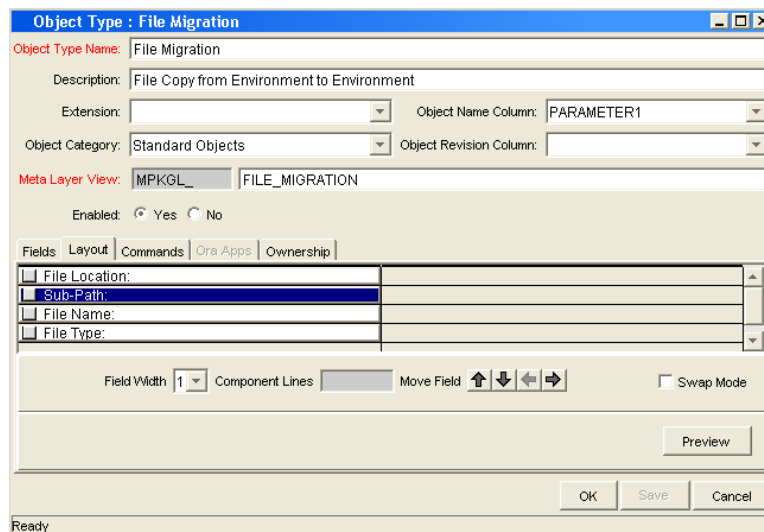
The **Layout** tab opens.

4. In the **Layout** tab, select a field.

To select more than one field, use the Shift-key while selecting a range. It is only possible to select a continuous set of fields (for example, the Ctrl-Select functionality is not supported).

5. Use the **Arrow** icons to move the fields to the desired location in the layout builder.

A field or a set of fields cannot be moved to an area where other fields already exist. The other field(s) must be moved out of the way first.



6. To switch the positions of two fields:

- a. Select the first field and check the Swap Mode check box.

An “S” appears in the check box area of the selected field.

- b. Double-click the second field that will switch positions with the first.

This causes the two fields to change positions. Following the switch, the Swap Mode check box is turned off. To swap another set of fields, repeat this procedure.

7. To check what the layout looks like in actual use, click **Preview**.

This opens a small window that shows the fields as they will appear. It is important to note that:

- If all the fields have a width of one column, all displayed columns will automatically span the entire available area when a package line of the given object type is viewed or generated.
- Any rows with no fields are ignored. They do not show up as a blank line.
- Any non-displayed fields do not affect the layout. They are considered the same as a blank field.

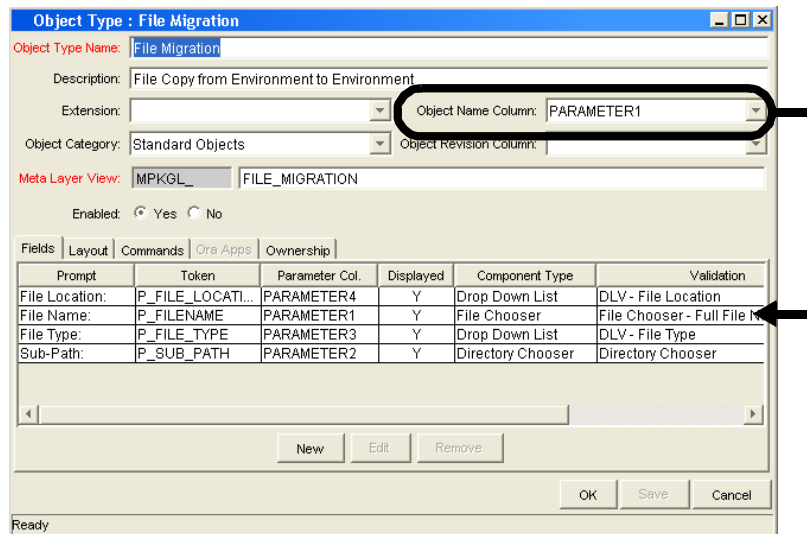
8. In the **Layout** tab, click **OK**.

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 drop-down list.

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



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

The object name field drives additional functionality:

- If the object name field is a file chooser or an auto-complete field, multi-selection will automatically be 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.
- The Object Name can be queried using the Object Type Workbench.

Setting Object Revisions

It is also possible to create a field on 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.

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 when some commands must only be executed in certain cases. This is powered by the condition field of the commands and is discussed in [Command Conditions on page 194](#).

For More Information

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

Adding Commands to Object Types

To add commands to object types:

1. Open the Object Type Workbench.

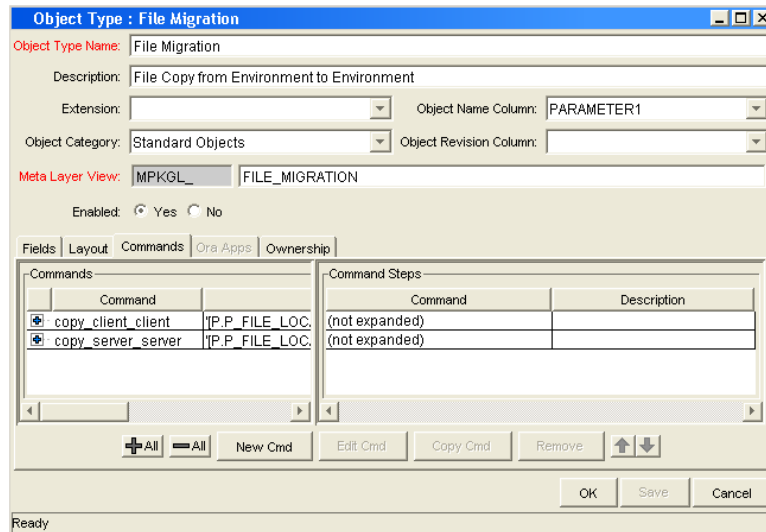
To open the Object Type Workbench, see [Opening the Object Type Workbench on page 170](#). The Object Type Workbench window opens.

2. Open an object type.

The Object Type window opens.

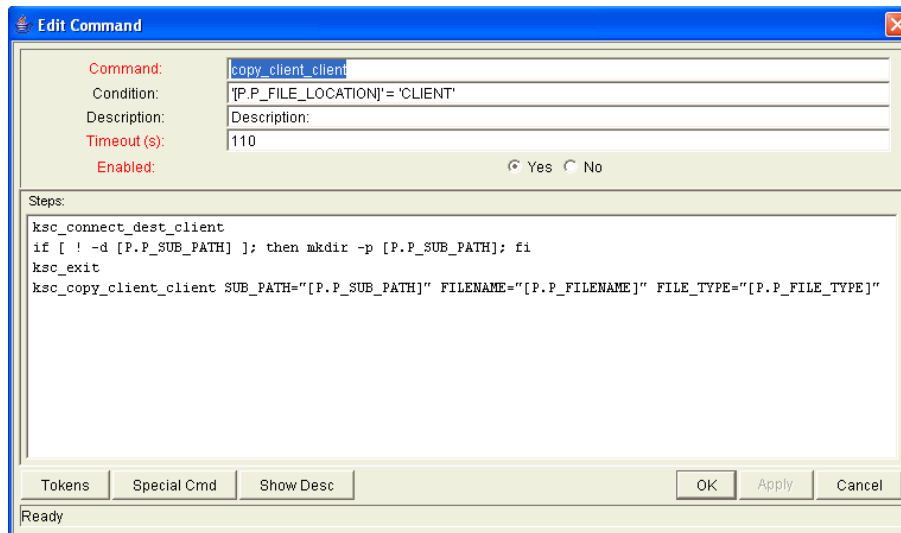
3. In the Object Type window, click the **Commands** tab.

The **Commands** tab opens.



4. In the **Commands** tab, click **New Cmd**.

The New Command window opens.



5. Complete the fields in the New Command window as specified in the following table:

Field	Description
Command	A simple name for the command.
Condition	A condition that determines whether the steps for the command are executed or not. (See Command Conditions on page 194 for more information).
Description	A description of the command.
Timeout	The amount of time the command will be allowed to run before its 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. In the New Command window, click **OK**.

The New Command window closes. The **Commands** tab lists the new command.

7. In the **Commands** tab, click **OK**.

The changes to the object type are saved.

Editing Commands of Object Types

To edit a command on an object type:

1. Open the Object Type Workbench.

To open the Object Type Workbench, see [Opening the Object Type Workbench on page 170](#). The Object Type Workbench window opens.

2. Open an object type.

The Object Type window opens.

3. In the Object Type window, click the **Commands** tab.

The **Commands** tab opens.

4. In the **Commands** tab, click **Edit Cmd.**

The Edit Command window opens.

5. Select the command to edit.

6. Complete the fields in the Edit Command window as specified in the following table:

Field	Description
Command	A simple name for the command.
Condition	A condition that determines whether the steps for the command are executed or not. (See Command Conditions on page 194 for more information).
Description	A description of the command.
Timeout	The amount of time the command will be allowed to run before its 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.

7. In the Edit Command window, click **OK.**

The Edit Command window closes. The **Commands** tab lists the edited command.

8. In the **Commands** tab, click **OK.**

The changes to the object type are saved.

Copying Commands in Object Types

To copy a command in an object types:

1. Open the Object Type Workbench.

To open the Object Type Workbench, see [Opening the Object Type Workbench on page 170](#). The Object Type Workbench window opens.

2. Open an object type.

The Object Type window opens.

3. In the Object Type window, click the **Commands** tab.

The **Commands** tab opens.

4. In the **Commands** tab, select the command to copy and click **Copy Cmd**.

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

5. In the **Commands** tab, click **OK**.

The changes to the object type are saved.

Deleting Commands in Object Types

To copy a command in an object types:

1. Open the Object Type Workbench.

To open the Object Type Workbench, see [Opening the Object Type Workbench on page 170](#). The Object Type Workbench window opens.

2. Open an object type.

The Object Type window opens.

3. In the Object Type window, click the **Commands** tab.

The **Commands** tab opens.

4. In the **Commands** tab, select the command to delete and click **Remove**.

The command is deleted.

5. In the **Commands** tab, click **OK**.

The changes to the object type are saved.

Command Conditions

In many situations, it might be necessary to run a different set of commands depending on the context of execution. This flexibility is achieved through the use of conditional commands. The Condition field for a command is used to define the situation under which the associated command steps execute.

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

Table 5-2. Example conditions

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

For More Information

The condition can include tokens. For more information concerning tokens, see *Commands, Tokens, and Validations Guide and Reference*.

Configuring Ownership for Object Types

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 with the Edit Access access grant for the entity can edit, copy or delete it. Refer to *Security Model Guide and Reference* for more information on access grants.

If a security group is disabled or loses the Edit Access access grant, that group will no longer be able to edit the entity.

Adding Ownerships to Object Types

To add an ownership:

1. Open the Object Type Workbench.

To open the Object Type Workbench, see [Opening the Object Type Workbench on page 170](#). The Object Type Workbench window opens.

2. Open an object type.

The Object Type window opens.

3. In the Object Type window, click the **Ownership** tab.

The **Ownership** tab opens.

The screenshot shows the 'Object Type : File Migration' window. The 'Ownership' tab is selected. The configuration includes:

- Object Type Name: File Migration
- Description: File Copy from Environment to Environment
- Extension: (empty dropdown)
- Object Name Column: PARAMETER1
- Object Category: Standard Objects
- Object Revision Column: (empty dropdown)
- Meta Layer View: MPKGL_ FILE_MIGRATION
- Enabled: Yes No

Under the 'Ownership' tab, there are two radio buttons for 'Give ability to edit this Object Type to:':

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

Below these options is a table with two columns: 'Security Group' and 'Description'. The table is currently empty. At the bottom of the table are 'Add' and 'Remove' buttons. At the bottom of the window are 'OK', 'Save', and 'Cancel' buttons.

4. In the **Ownership** tab, select the ownership option.

The All user with the Edit Object Type access grant option give all users with the Edit Object Type access grant ownership of the object type. The Only groups listed below that have the Edit Object Type access grant option requires selected groups to be added to the ownership of the object type.

If Only groups listed below that have the Edit Object Type access grant is selected:

- a. In the **Ownership** tab, click **Add**.

The Add Security Groups window opens.

- b. In the Add Security Groups window, in the Security Groups field, select the security groups.

The Validate window opens.

- c. In the Validate window, select one or more security groups and click **OK**.

The Validate window closes. The Add Security Groups window lists the selected security groups.

- d. In the Add Security Groups window, click **OK**.

The Add Security Groups window closes. The selected security groups are display in the **Ownership** tab under the Security Group column.

5. In the **Ownership** tab, click **OK**.

The changes to the object type are saved.

Deleting Ownerships from Object Types

To delete an ownership:

1. Open the Object Type Workbench.

To open the Object Type Workbench, see [Opening the Object Type Workbench on page 170](#). The Object Type Workbench window opens.

2. Open an object type.

The Object Type window opens.

3. In the Object Type window, click the **Ownership** tab.

The **Ownership** tab opens.

4. In the **Ownership** tab, select an ownership.

The All user with the Edit Object Type access grant option give all users with the Edit Object Type access grant ownership of the object type. The Only groups listed below that have the Edit Object Type access grant option requires selected groups to be added to the ownership of the object type.

5. In the **Ownership** tab, click **Remove**.

The ownership is deleted.

6. In the **Ownership** tab, click **OK**.

The changes to the object type are saved.

Using Commands to Change Field Values

Commands can also be used to control certain behavior of object type fields. At specific points (execution workflow steps) in the deployment process, it is possible to 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 provides an advantage over the defaulting features in the Field window, which can only 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 *Commands, Tokens, and Validations Guide and Reference*.

Controlling field values using commands can be useful in the following situations:

- Storing a value from an execution into a custom field
- Clearing a field after evaluating a number of parameters

Configuring Releases and Distributions

In This Chapter:

- *Overview of Releases and Distributions*
 - *Workflow Scope*
 - *Release Management and Package Workflows*
 - *Release Distribution Workflows*
 - *Package Level Subworkflows*
 - *Dependencies and Run Groups*
 - *Opening Releases*
 - *Submitting Releases*
 - *Overview of Using Release Management - Process*
 - *Distributions*
- *Overview of Configuring Releases*
- *Opening the Release Workbench*
- *Creating Releases*
- *Adding Packages to Releases*
 - *Adding Packages Through the Release Window*
 - *Adding Packages Through the Package Window*
 - *Adding Packages Through the Release Window*
 - *Adding Packages from Requests*
- *Adding Requests to Releases*
 - *Adding Requests Through the Release Window*
 - *Adding Requests Through the Requests Window*
- *Verifying Releases*
- *Creating Distributions*
 - *Enabling/Disabling Package Lines in a Distribution*
 - *Running Distributions through a Workflow*
 - *Completing Distributions*

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. Packages can be added to a release either by a Ready for Release Step in the package workflow or by the release manager through the Release window.

For example, a software company has a product update release scheduled five months from now. In order to ensure a smooth product delivery, they decide to track all changes to their original code using release management. As developers complete their packages, those packages are included in a release and processed together. By grouping every required change in the release, the company is able to quickly and easily 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. Certain workflow configuration restrictions are enforced by the workflow scope. For example, release distribution workflows can not 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 of the package lines are confirmed in the Ready for Release step, the package is ready to be used in the release.

Figure 6-1 on page 201 illustrates the process by which developers can add packages to a release.

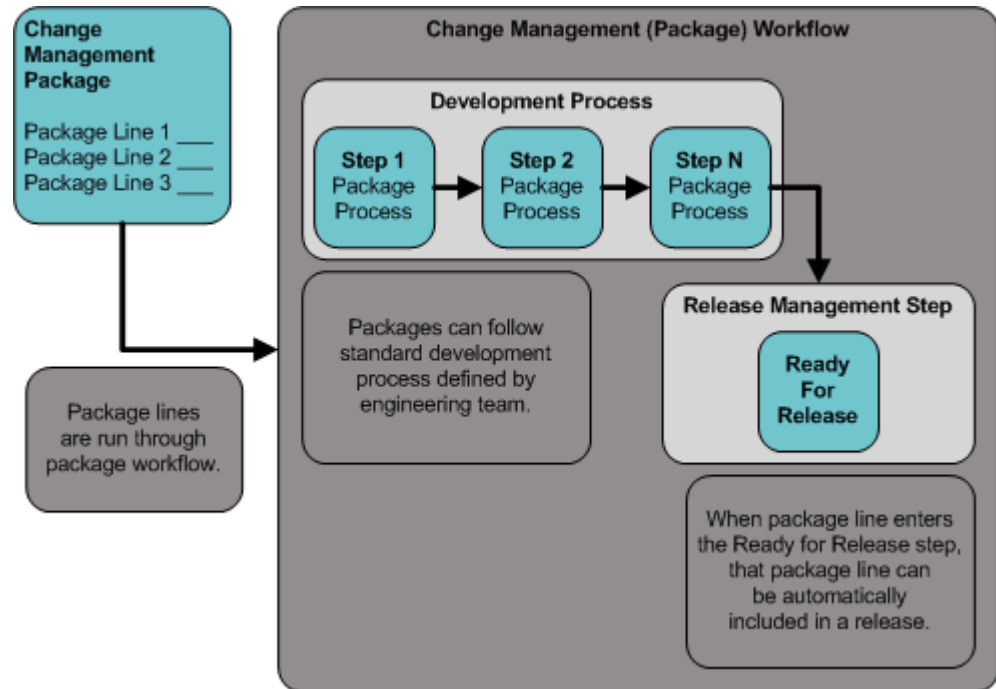


Figure 6-1. Ready for release step in workflow

Release Distribution Workflows

Just as the inclusion of appropriate 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 executed against any required environments.

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

[Figure 6-2 on page 202](#) illustrates the relationships between packages and release workflows.

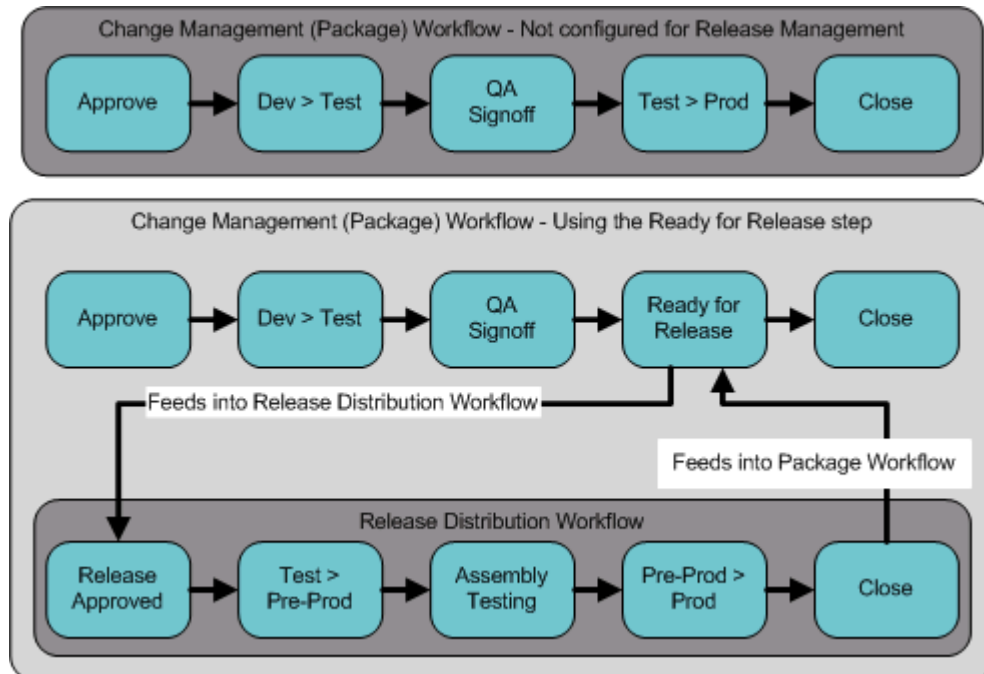


Figure 6-2. Role of the distribution workflow

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 will occur in these subworkflows. Also, all package and package line tokens will be resolved when traversing through these workflows.

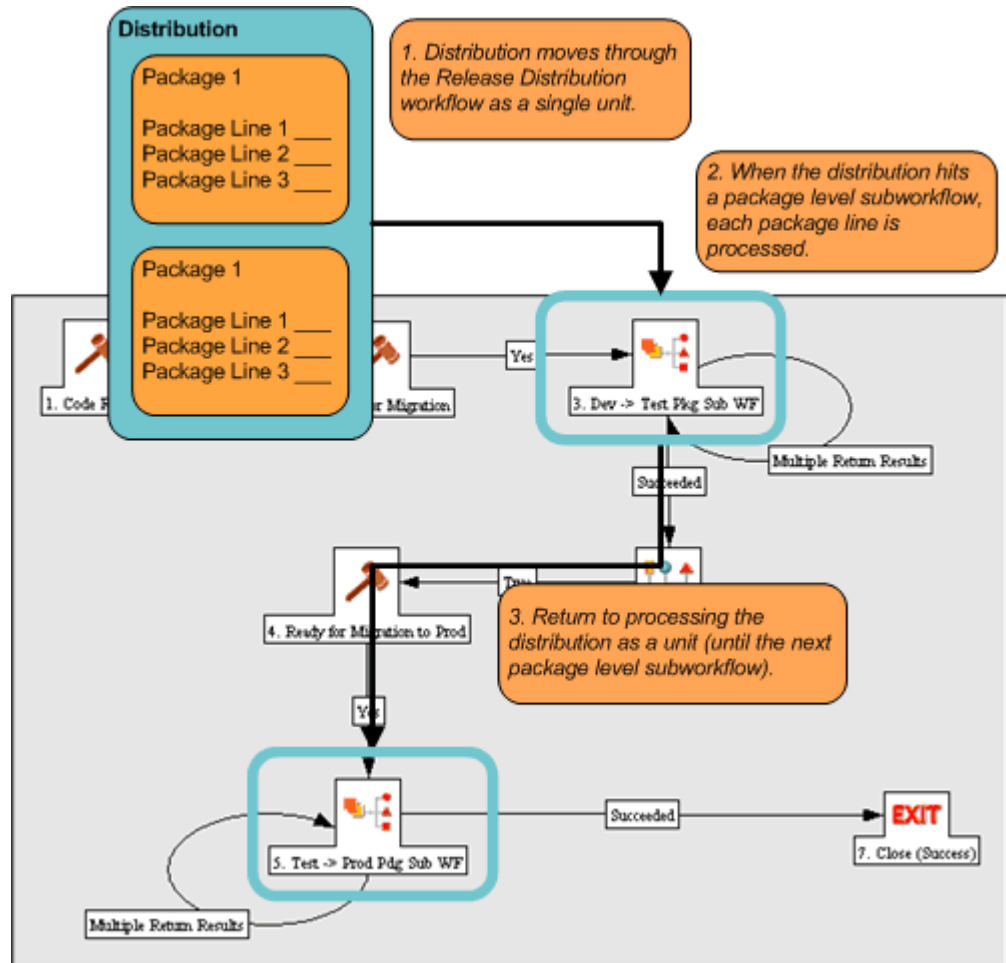


Figure 6-3. Distribution workflow

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 execute 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 executed 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 non-related dependencies. [Figure 6-4 on page 205](#) illustrates 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.

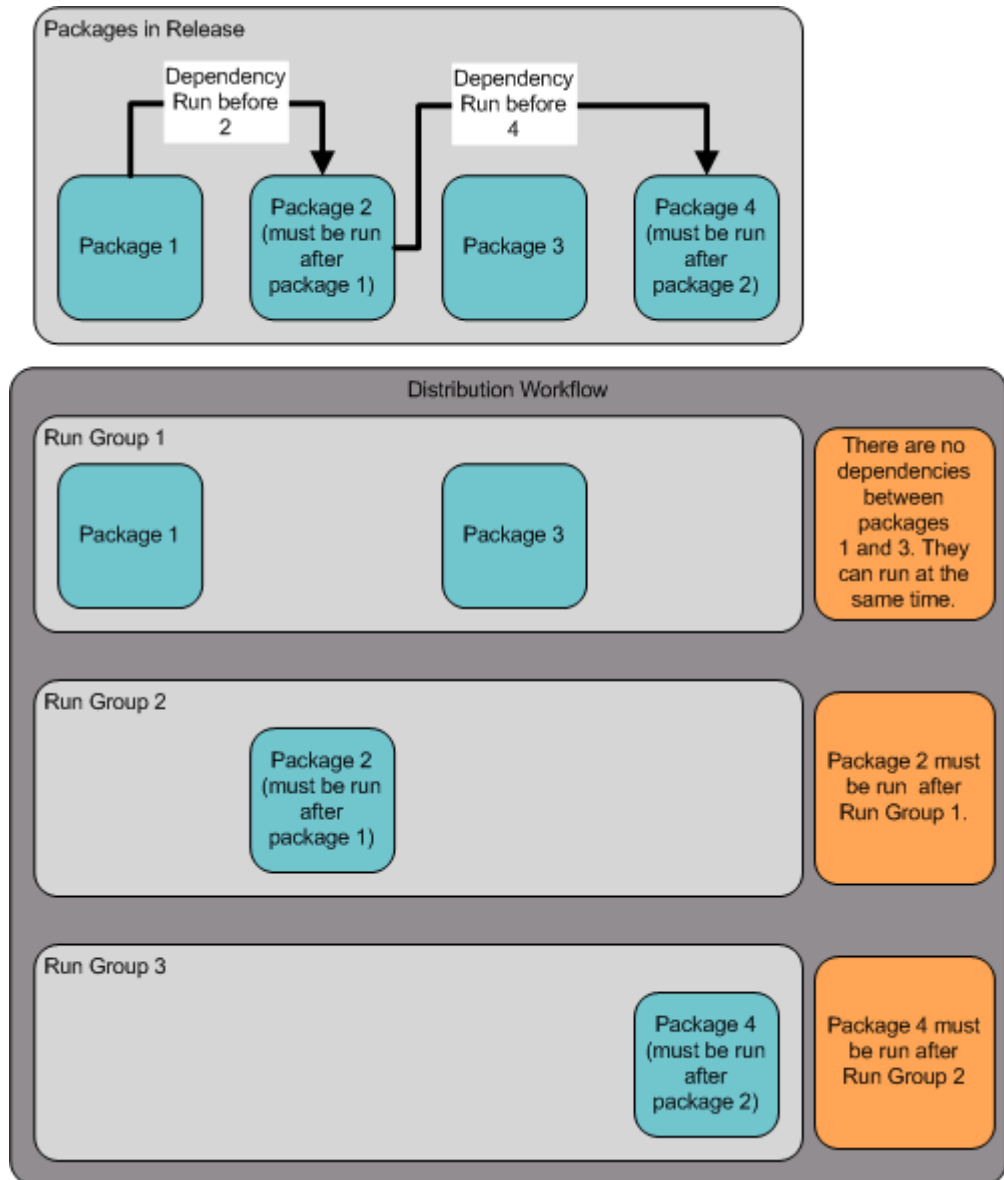


Figure 6-4. Dependencies and run groups

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 the **Open Release** button 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 will not appear 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 Pre-Configuration

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 pre-configure 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 will 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 executed 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.
2. Open the release by clicking **Open Release**.

This allows developers working in the Change Management Package window to add packages to the release via the Ready for Release workflow step.

3. Add packages and requests to the release.

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 appropriate data is collected in the release (packages, requests and dependencies) and the appropriate 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 you do not want run with that distribution.
 - b. Submit the distribution.
2. Send feedback to packages. You can 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 you do not 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 has not been submitted, it will not be cancelled when the release is closed. However, 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 different environments differs in the following ways:

- Not all of the packages in the release need to be applied to each instance, such as 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, such as 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.
 - Which 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. See [Overview of Using Release Management - Process on page 206](#) for an overview of the items and processes involved in creating a release.

One of the first steps in establishing a controlled release is to create a new release in the Release Workbench.

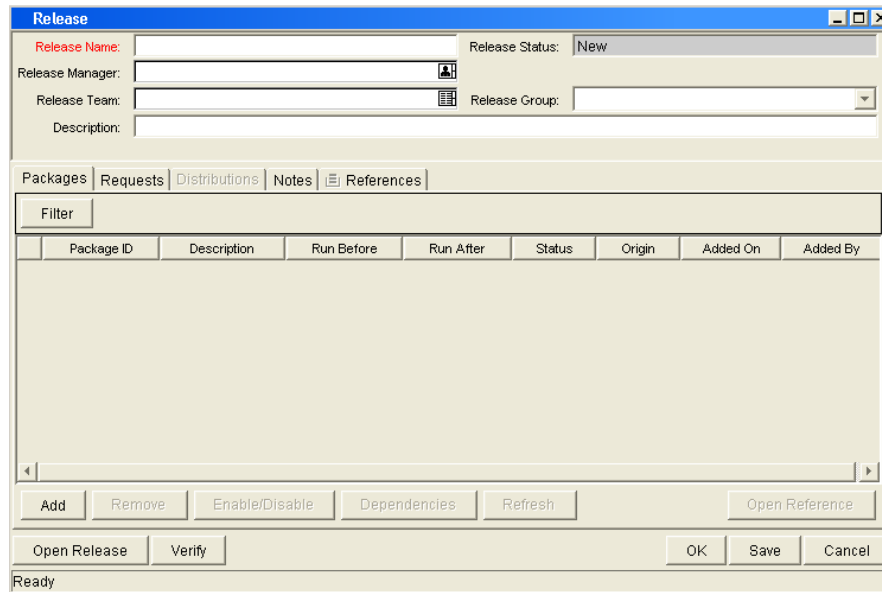


Figure 6-5. Release window

The following lists the major section found in the Release window:

- **General information.** General information includes basic information concerning the environment, such as the environment name and description. See [Creating Releases on page 212](#).
- **Packages.** The **Packages** tab allows packages to be added to the release. See [Adding Packages to Releases on page 214](#).
- **Requests.** The **Requests** tab allows requests to be added to the release. See [Adding Requests to Releases on page 219](#).
- **Distributions.** The **Distributions** tab creates distributions for releases. See [Creating Distributions on page 223](#).
- **Notes.** The **Notes** tab allows you to add notes to a release.
- **References.** The **References** tab tracks information regarding the release.

Opening the Release Workbench

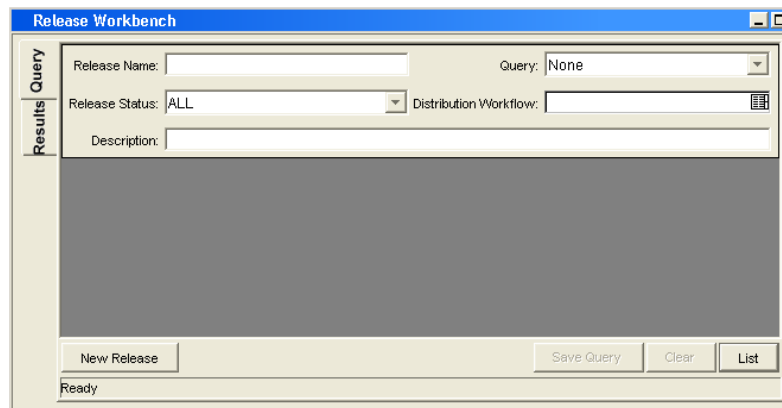
To open the Release Workbench:

1. Log on to the Mercury IT Governance Center.
2. From the menu bar, select **Administration > Open Workbench**.
3. A Workbench status window opens. A few minutes later, a Warning - Security window opens.
4. In the Warning - Security window, select **Yes**.

The Workbench opens.

5. From the shortcut bar, select **Change Mgmt > Releases**.

The Release Workbench window opens.



For More Information

For information on how to search and select an existing release, copy a release, and delete a release, see *Getting Started*.

Creating Releases

To configure general information and create a release:

1. Open the Release Workbench.

To open the Release Workbench, see [Opening the Release Workbench on page 211](#). The Release Workbench window opens.

2. In the Release Workbench, click **New Release**.

The Release window opens.

The screenshot shows the 'Release' window with the following details:

- Title Bar:** Release
- Metadata Fields:**
 - Release Name: [Empty]
 - Release Manager: [Empty]
 - Release Team: [Empty]
 - Release Group: [Dropdown]
 - Release Status: New
 - Description: [Empty]
- Navigation Tabs:** Packages (selected), Requests, Distributions, Notes, References
- Table:**

Package ID	Description	Run Before	Run After	Status	Origin	Added On	Added By
[Empty]							
- Action Buttons:** Add, Remove, Enable/Disable, Dependencies, Refresh, Open Reference
- Bottom Buttons:** Open Release, Verify, OK, Save, Cancel
- Status Bar:** Ready

3. In the Release window, complete the fields as specified in the following table:

Field	Description
Release Name	The name of the release.
Release Status	The current status of the Release: New, Open, Code Freeze, or Closed.
Release Manager	The name of the release management user who has control over the particular release. Only release management users that have the Manage Releases access grant will appear in this list.
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. In the Release window, add packages to the release.

See [Adding Packages to Releases on page 214](#).

5. In the Release window, add requests to the release.

See [Adding Requests to Releases on page 219](#).

6. In the Release window, click **Open Release** to allow other release management users to add packages and requests to the release.

7. In the Release window, save the changes to the release.

Click **OK** to save the changes and close the Release window. Click **Save** to save the changes and leave the Release window open. Click **Cancel** to lose the changes and close the Release window.

Adding Packages to Releases

You can add packages to release in one of the following ways.

Adding Packages Through the 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. Open the Release Workbench.

To open the Release Workbench, see [Opening the Release Workbench on page 211](#). The Release Workbench window opens.

2. Open a release.

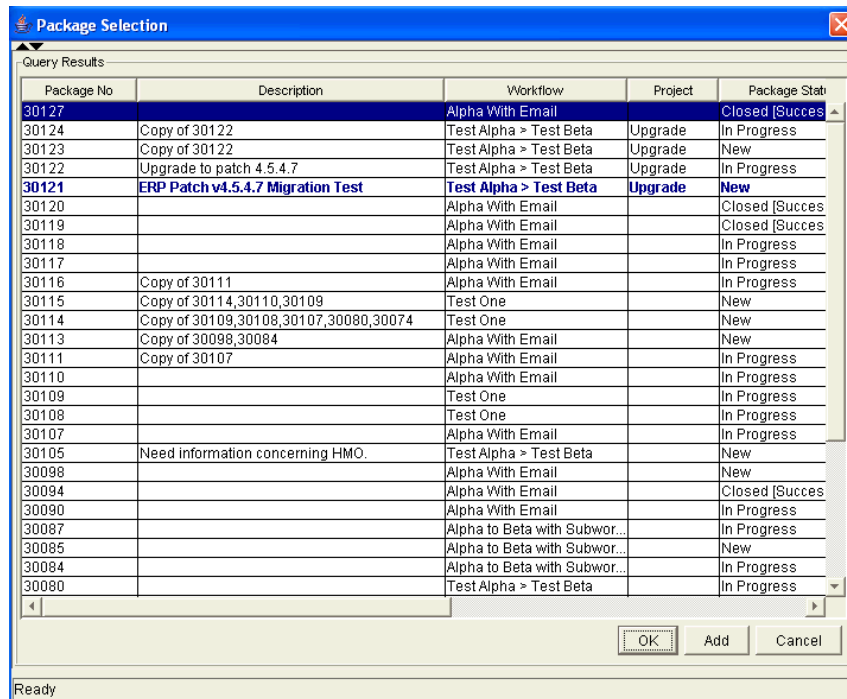
The Release window opens. The **Packages** tab is displayed.

3. In the **Packages** tab, click **Add**.

The Package Selection window opens.

4. In the Package Selection window, enter any desired search criteria and click **List**.

Packages matching your search criteria are dynamically listed in the Query Results list.



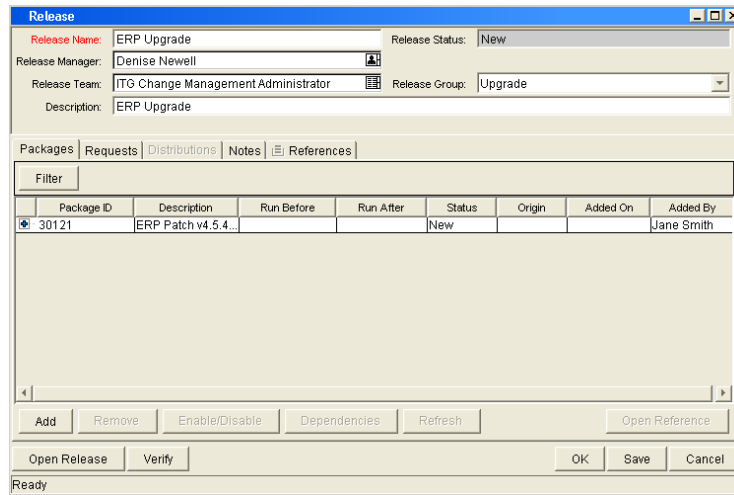
5. In the Query Results list, select the packages to be added to the release.

6. In the Package Selection window, click **Add**.

If the packages reference any other packages or requests, you will then be prompted to include or exclude them.

7. In the Package Selection window, click **Close**.

This returns you to the Release window. The newly added packages are displayed.



8. In the Release window, click **Save**.

The changes to the release are saved.

Adding Packages Through the Package Window

Packages can be added to a release through the Package window. To add a package to a release through the Package window, reference the release through the package **References** tab.

The References window is shown in *Figure 6-6*.

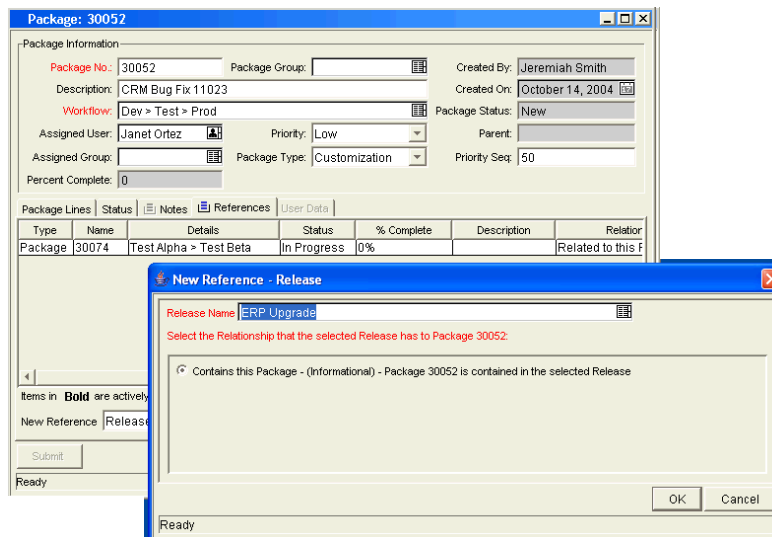


Figure 6-6. Package added to a release through the package reference

Adding Packages by the 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 executed, the status of the package line is changed to Confirmed. As soon as all of the lines in a package are Confirmed, the entire package's status becomes Ready for Release. Once an entire package becomes Ready for Release, the release distribution can feed back to each of its associated packages so that each of them can transition to the next workflow step.

You can reject the Ready for Release workflow step by choosing **Bypass Execution** or **Override Status**. This will stop the package from getting released but will still allow it to continue through its own workflow.

To act on a Ready for Release workflow step:

1. Open the Release Workbench.

To open the Release Workbench, see [Opening the Release Workbench on page 211](#). The Release Workbench window opens.

2. From the shortcut menu, select **Change Mgmt > Packages**.

The Package Workbench opens.

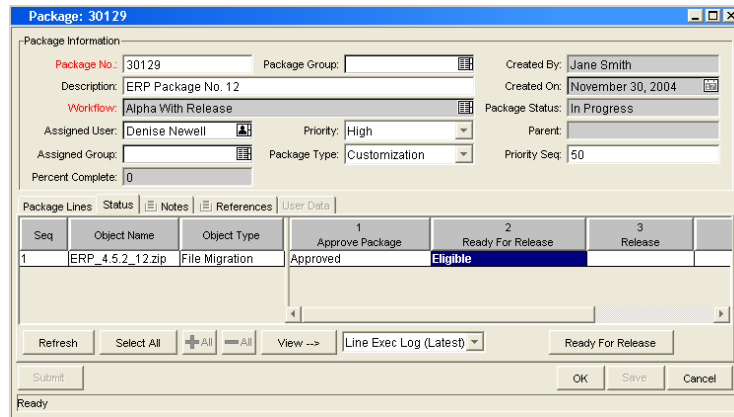
3. Open a package.

The Package window opens.

4. In the Package window, click the **Status** tab.

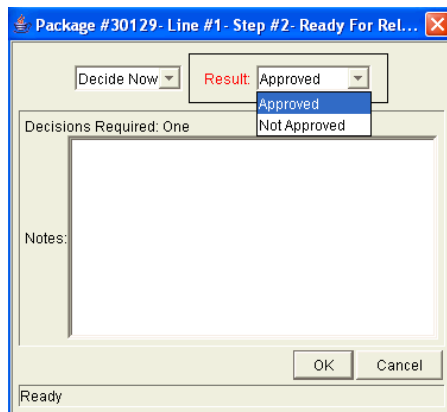
5. In the **Status** tab, select the workflow step to be acted upon.

Workflows that are eligible for action are displayed in bold. Once an eligible workflow step is selected, the button at the bottom right of the **Status** tab will change its title (originally **Action**) to the name of the step.



6. In the **Status** tab, click **Ready for Release**.

The Package Action window opens.



7. In the Package Action window, select the workflow step's result from the drop-down list.

You can also enter any relevant notes in this window.

8. In the Package Action window, select the release you want the package to be associated with from the Add to Release autocomplete list.

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

9. In the Package Action window, click **OK** to save the result.

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

The package is now ready to be released.

Adding Packages from Requests

When a request that is included in a release spawns a package, that package is automatically included in the release. 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. That request's workflow can be configured to automatically create a package to migrate changes into production. That package will automatically be included in the release.

Adding Requests to Releases

Requests can be added to a release to track information associated with the release. For example, the release manager may want to track which software bugs or enhancements (captured using a request) were implemented during the release timeframe.

Adding Requests Through the Release Window

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

1. Open the Release Workbench.

To open the Release Workbench, see [Opening the Release Workbench on page 211](#). The Release Workbench window opens.

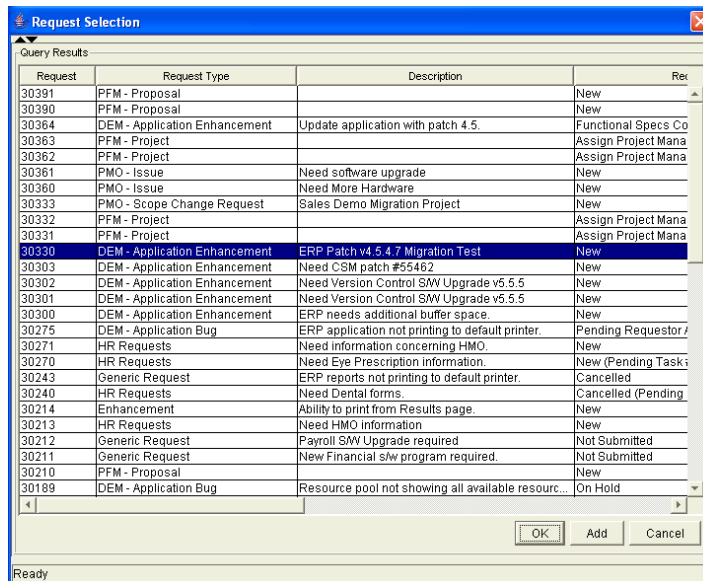
2. Open a release.

The Release window opens.

3. In the Release window, select the **Requests** tab.

4. In the **Requests** tab, click **Add**.

The Request Selection window opens.



5. In the Request Selection window, enter search criteria and click **List**.

Requests matching your search criteria are dynamically listed in the Query Results list of the window.

6. In the Query Results list, select the request to be added to the release.

7. In the Request Selection window, click **Add**.

If there are any referenced entities, you will then be prompted to include or exclude them.

8. In the Request Selection window, click **Close**.

This will return you to the Release window which displays the newly added request.

9. In the Release window, click **Save**.

The changes to the release are saved.

Adding Requests Through the Requests Window

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

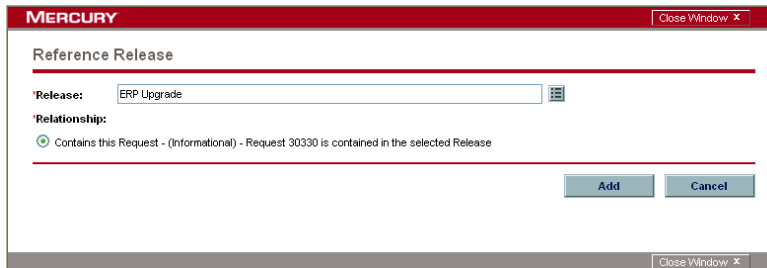
To add a request to a release:

1. From the Dashboard, open a request.
2. In the request, select the References section.
3. In the References section, from the New Reference drop-down list, select **Release** and click **Add**.

The Reference Release window opens.

4. In the Reference Release window, select the desired release from the Release autocomplete list.

Only releases that the release manager has opened by clicking **Open Release** will appear in the Release list.



The screenshot shows a window titled "Reference Release" with a red header bar containing the "MERCURY" logo and a "Close Window" button. The main content area has a "Release:" label followed by a text input field containing "ERP Upgrade" and a dropdown arrow. Below this is a "Relationship:" label with a checked radio button and the text "Contains this Request - (Informational) - Request 30330 is contained in the selected Release". At the bottom right of the main area are "Add" and "Cancel" buttons. A second "Close Window" button is visible at the very bottom of the window frame.

Verifying Releases

When you finish assembling the release (packages, requests, and setting dependencies), you can verify that your release is properly configured.

To verify your release:

1. Open the Release Workbench.

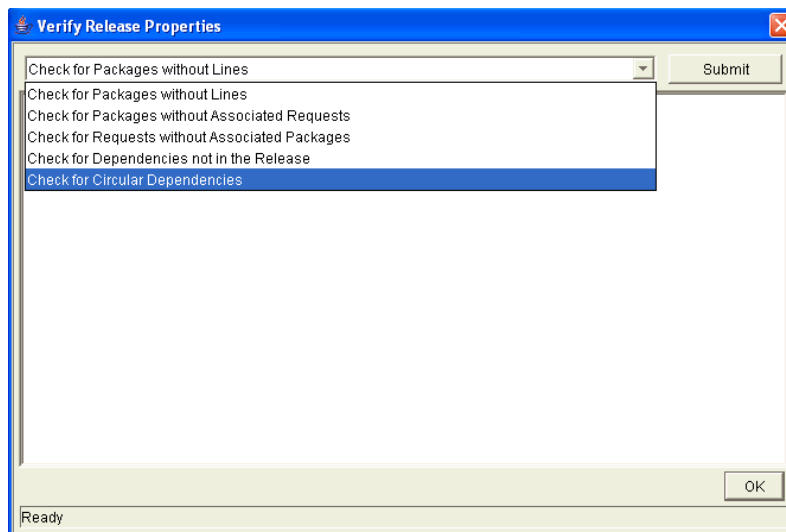
To open the Release Workbench, see [Opening the Release Workbench on page 211](#). The Release Workbench window opens.

2. Open a release.

The Release window opens. The **Packages** tab is displayed.

3. In the Release window, click **Verify**.

The Verify Release Properties window opens.



4. In the Verify Release Properties window, select the property that you would like to check for from the drop-down list. The verify options include:

- 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. In the Verify Release Properties window, click **Submit**.

Any errors will be reported in the window.

6. In the Verify Release Properties window, click **OK**.

The Verify Release Properties window closes.

Creating 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.

To create a distribution:

1. Open the Release Workbench.

To open the Release Workbench, see [Opening the Release Workbench on page 211](#). The Release Workbench window opens.

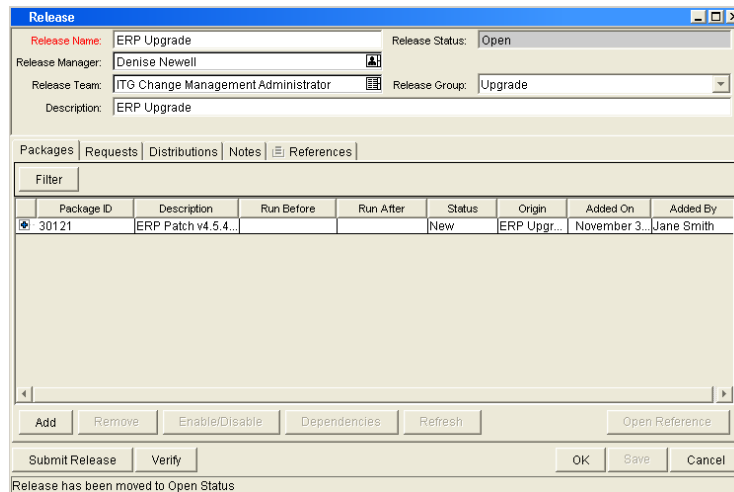
2. Open a release.

The Release window opens.

3. In the Release window, click **Open Release**.

The **Distributions** tab is enabled.

4. In the Release window, select the **Distributions** tab.



5. In the **Distributions** tab, click **New**.

The Distribution window opens.

6. In the Distribution window, enter the new Distribution Name and Description.

7. In the Distribution window, select the workflow that you would like this distribution to follow.

8. In the Distribution window, select any packages to disable and click **Enable/Disable**.

Disabled packages will appear in italics.

9. In the Distribution window, click **Submit Release**.

The distribution runs through the workflow specified. The release will begin running along the assigned workflow.

Enabling/Disabling Package Lines in a Distribution

To disable a package line in a distribution:

1. Open the Release Workbench.

To open the Release Workbench, see [Opening the Release Workbench on page 211](#). The Release Workbench window opens.

2. Open a release.

The Release window opens.

3. In the Release window, select the **Distributions** tab.

The **Distributions** tab is displayed.

4. In the **Distributions** tab, click the **Package Status** tab.

The **Package Status** tab is displayed.

5. In the **Package Status** tab, locate the package line that you want to disable.

6. In the **Package Status** tab, click the expand **Run Groups** button to display all of the packages.

Note that the packages may be filtered. You may have to change your filter to show the desired package.

7. In the **Package Status** tab, select the package line to disable.

You can also select to disable an entire package. Disabling a package line in an active run group (within a package-level subworkflow) will cancel the package line.

8. In the **Package Status** tab, click **Disable**.

The disabled package line is displayed in italics. You can re-enable the package lines by selecting the lines and clicking **Enable**. If you disable a package line in an active run group, you can't re-enable the package line until the run group completes. If the disabled package line was not in an active run group, you can re-enable it immediately.

Running Distributions through a Workflow

The last step involved in creating a release is running the release through a Change Management workflow. Running the release through a workflow implies running any decisions, commands, token evaluations, or other tasks for the distribution as a whole.

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

Processing Distribution Steps

Active workflow steps appear in bold text. Select the active line and click the **Action** button to process that workflow step. From 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.

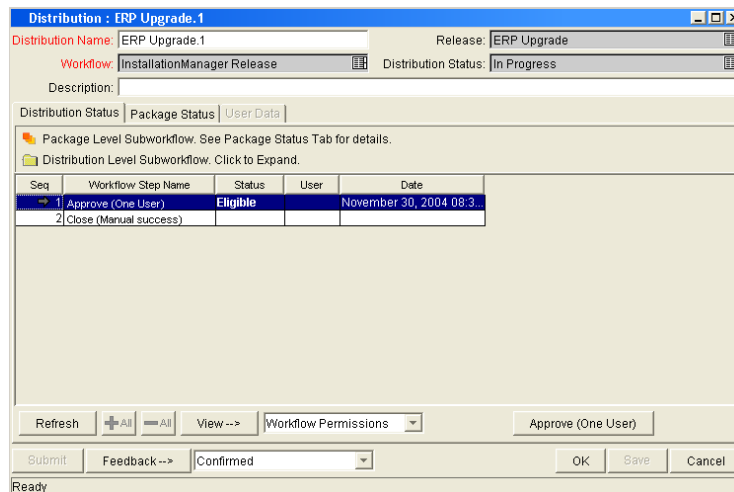


Figure 6-7. Distribution Status tab

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 the **Action** button 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. Open the Release Workbench.

To open the Release Workbench, see [Opening the Release Workbench on page 211](#). The Release Workbench window opens.

2. Open a release.

The Release window opens.

3. In the Release window, select the **Distributions** tab.

The **Distributions** tab is displayed.

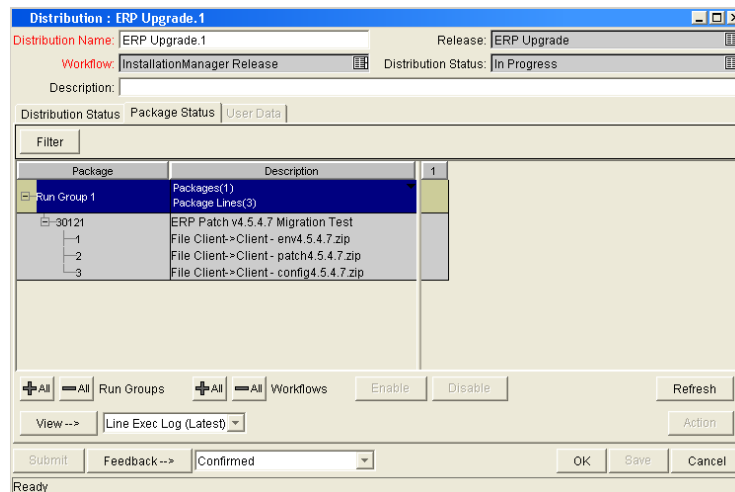
4. In the **Distributions** tab, click the **Package Status** tab.

The **Package Status** tab is displayed.

5. In the **Package Status** tab, expand all run groups and workflows.

6. In the **Package Status** tab, click the plus sign in the Description column.

This displays a status summary of all the package lines in each workflow step.



7. In the **Package Status** tab, 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. In the **Package Status** tab, click the **Action** button to process all of the selected package lines.

9. In the **Package Status** tab, proceed to the next workflow step in the package process or distribution process (depending on your pre-configured process).

To view updates in the **Distribution Status** tab, click in the **Distribution Status** and 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. That value is sent by clicking **Feedback** in the Distribution window.

Chapter

7

Configuring Environments

In This Chapter:

- *Overview of Environments*
 - *Overview of Configuring Environments*
 - *Opening the Environments Workbench*
 - *Configuring General Information for Environments*
 - *Creating Environments*
 - *Using Application Codes Environments*
 - *Copying Application Codes from Other Environments*
 - *Setting Ownership for Environments*
 - *Adding Ownerships to Environments*
 - *Deleting Ownerships from Environments*
 - *Adding Participants to Environments*
 - *Deleting Participants from Environments*
 - *Environment Maintenance and Utilities*
 - *Testing Environment Setups*
 - *Mass Updates of Base Paths*
 - *Environment Password Management Utility*
-

Overview of Environments

When migrating objects, Mercury Change Management logs onto remote computers in the same way any other user would (using FTP, SCP, SSH, or Telnet). Mercury Change Management can log on using any existing username and password. However, it is recommended that a new user (such as Mercury IT Governance) be generated on each computer that Mercury Change Management will access. This helps to clarify the setup and relieve some administrative burden.

You should have full access to the *ITG_Home* directory on the Mercury IT Governance Center Server as well as the correct read and write permissions on other required directories. In addition, on Windows NT computers, the **Administrators** group must have read access to the Mercury IT Governance Center Server home directory. Any Windows NT computer that Mercury Change Management will access should have been configured as directed in *System Administration Guide and Reference*.

Environment Connection Protocols

The communication protocol that will be used to connect to the server or client must be specified in the environment. This protocol will be 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.

Mercury Change Management supports the following connection protocols:

- Telnet
- SSH
- SSH2

Environment Transfer Protocols

The transfer protocol that will be used to transfer files to the server or client must be specified in the environment.

Mercury Change Management supports the following transfer protocols:

- FTP
- FTP (active)
- FTP (passive)
- Secure Copy
- Secure Copy 2

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 the 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 on page 232](#).
 - 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

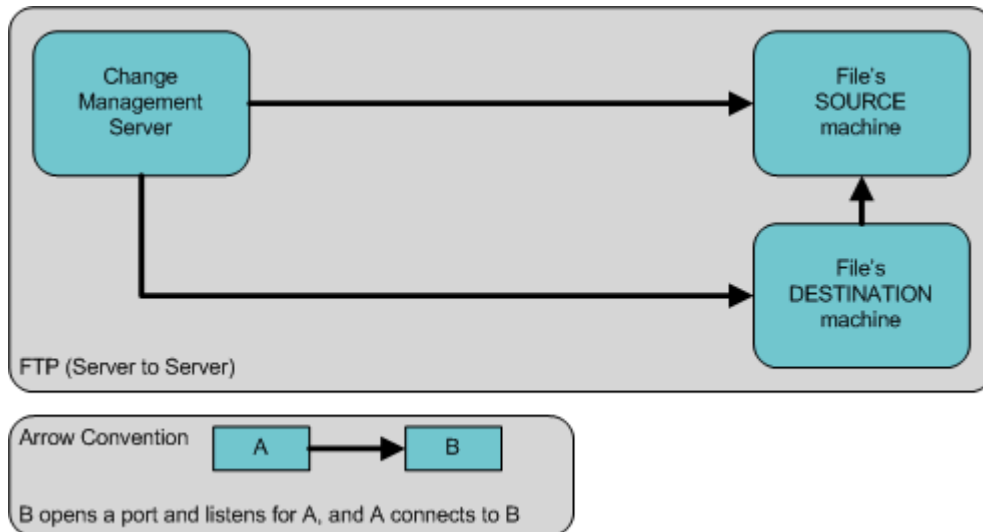


Figure 7-1. FTP (server to server)

- **FTP (active).** See [Figure 7-2](#).
 - PORT command must be enabled on both the source and destination environments (allows outgoing back to the requestor)

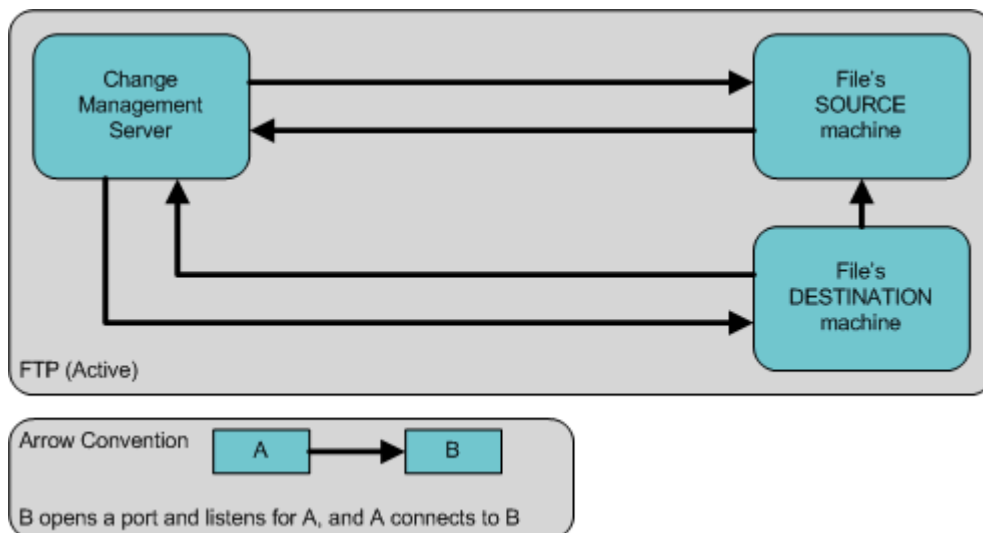


Figure 7-2. FTP (active)

- **FTP (passive).** See *Figure 7-3*.
 - PASV must be enabled on both the source and destination environments. In this configuration, the Mercury Change Management server sends a command to the source or destination instructing that environment to open a port. The Mercury Change Management server then connects to that port.

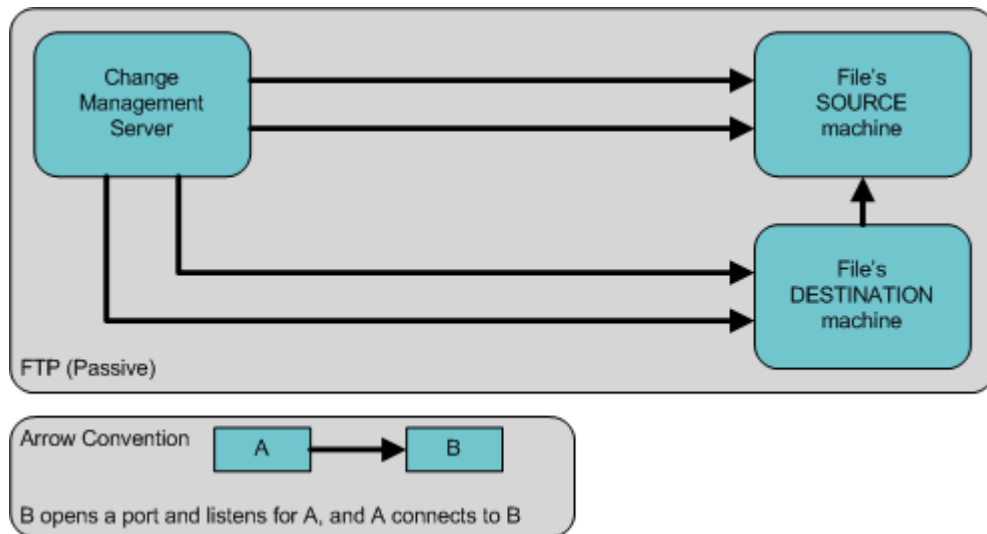


Figure 7-3. FTP (passive)

Overview of Configuring Environments

Environments are configured in the Environments window. Some of the information entered on the Environments window can be gathered from the appropriate Workflow Step Worksheets (see *Figure 7-4*).

Execution Workflow Step Worksheets

Table A-2. Workflow step [execution], step number ____.

	Value
Step Name	
Goal / Result of Step	
Validation*	
Execution Type**	
Processing Type	
Source Environment (Group)	
Dest Environment (Group)	
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)	

Figure 7-4. Worksheet and Environments window

The following lists the major section found in the Environments window:

- **General information.** General information includes basic information concerning the environment, such as the environment name and description. See *Configuring General Information for Environments* on page 237.
- **Host.** The **Host** tab 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 238.

- **Applications.** Every Mercury Change Management environment can contain its own set of applications. See [Using Application Codes Environments on page 243](#).
- **Extension Data.** Mercury Change Management Extensions requires specially configured environments. If Mercury Change Management Extensions products are not installed, the **Extension Data** tab is disabled.
- **Ownership.** Configure who can edit the environment. See [Setting Ownership for Environments on page 248](#).
- **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 251](#).
- **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 the Environments Workbench

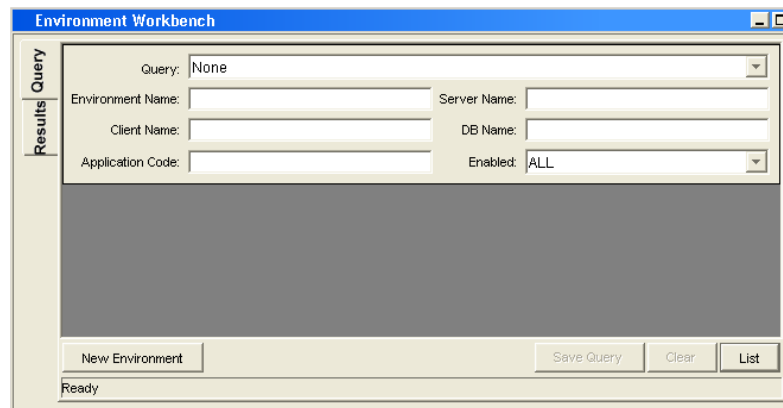
To open the Environments Workbench:

1. Log on to the Mercury IT Governance Center.
2. From the menu bar, select **Administration > Open Workbench**.
3. A Workbench status window opens. A few minutes later, a Warning - Security window opens.
4. In the Warning - Security window, select **Yes**.

The Workbench opens.

5. From the shortcut bar, select **Environments > Environments**.

The Environments Workbench window opens.



For More Information

For information on how to search and select an existing environment, copy an environment, and delete an environment, see *Getting Started*.

Configuring General Information for Environments

To configure the general information of an environment:

1. Open the Environments Workbench.

To open the Environments Workbench, see [Opening the Environments Workbench on page 236](#). The Environments Workbench window opens.

2. Open an environment.

The Environment window opens.

3. Complete the fields in the Environment window as specified in the following table:

Field	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	Makes the environment available to the system. Select Yes to make the environment available to the system.

4. In the Environments window, save the changes to the environment.

Click **OK** to save the changes and close the Environment window. Click **Save** to save the changes and leave the Environment window open. Click **Cancel** to lose the changes and close the Environment window.

Creating Environments

To define a new environment:

1. Open the Environments Workbench.

To open the Environments Workbench, see [Opening the Environments Workbench on page 236](#). The Environments Workbench window opens.

2. Open an environment.

The Environment window opens. The **Hosts** tab is displayed. The **Hosts** tab is divided into three parts:

- Server
- Client
- Database

The screenshot shows the 'Environment : test_beta' window. At the top, the 'Environment Name' is 'test_beta' and the 'Description' is 'computer and database hosting the application server'. Below this, there are fields for 'Location', 'Enabled' (radio buttons for Yes and No), and a tabbed interface with 'Host' selected. The 'Host' tab contains three sections: 'Server', 'Client', and 'Database'. The 'Server' section has fields for Name, Username, NT Domain, Connection Protocol, Type, Password, Base Path, and Transfer Protocol. The 'Client' section has similar fields. The 'Database' section has a 'Server Type' dropdown and fields for Host Name, Username, Oracle SID, DB Link, JDBC URL, Connect String, Password, Port Number, and Version. On the right side of each section, there is an 'Enable' checkbox. At the bottom, there are 'Check...', 'OK', 'Save', and 'Cancel' buttons, and a 'Ready' status indicator.

3. Complete the fields in the Server section as specified in the following table:

Fields	Description
Name	The DNS name or IP address of the computer.
Type	A drop-down 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 Mercury Change Management uses to log onto the server/client in order to transfer files or execute commands. This will usually be "Mercury ITG" 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 Windows NT computer.
Base Path	The path for applications on this computer. In many instances, this is the home directory of the defined username. When Mercury Change Management transfers a file or executes 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. In order to use SSH as your connection protocol, you must first setup 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 setup SCP on the destination machine.
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:

Fields	Description
Name	The DNS name or IP address of the computer.
Type	A drop-down 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 Mercury Change Management uses to log onto the server/client in order to transfer files or execute commands. This will usually be "Mercury ITG" 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 Windows NT computer.
Base Path	The path for applications on this computer. In many instances, this is the home directory of the defined username. When Mercury Change Management transfers a file or executes 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. In order to use SSH as your connection protocol, you must first setup 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 setup SCP on the destination machine.
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, enter the database server type in the Server Type field.

- If **Oracle Server** is selected, complete the fields as specified in the following table:

Fields	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 Mercury Change 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 in conjunction 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:

Fields	Description
Server Name	The DNS name or IP Address of the system running the SQL database.
DB Name	The DB name used by Mercury Change Management when connecting to the SQL database.
User Login	The login ID used by Mercury Change 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.

6. At the bottom of the Environment window, click **OK**.

The changes to the environment are saved.

Using Application Codes 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 in the **Host** tab.

When adding a line to a package, there is the option of choosing the application code to specify the application that the migration object belongs to. When that 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 will be inherited by each application code and will appear in 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. Therefore, required Environment User Data fields are not also required at the application code level.

Each environment can contain its own set of applications. The **Applications** tab of the Environment window is shown in [Figure 7-5](#).

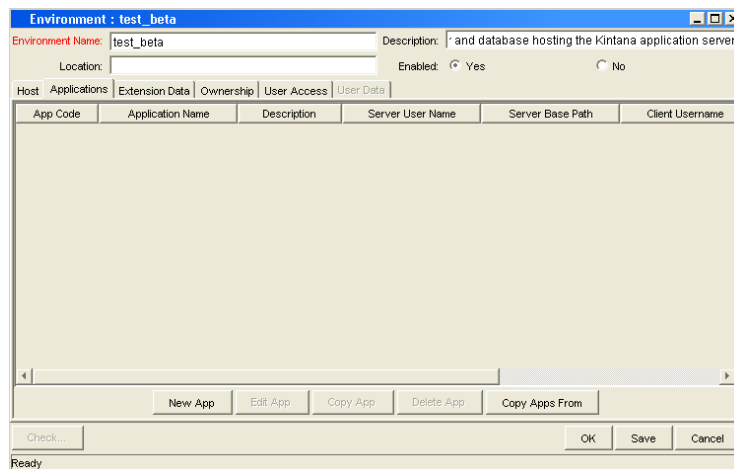


Figure 7-5. Applications tab

The **Applications** tab consists of the various fields and buttons shown in the [Figure 7-5 on page 243](#). Application fields do not always have to be populated. Click **New App** to open the New Application Code window, where a new application code can be defined.

To define the application codes:

1. Open the Environments Workbench.

To open the Environments Workbench, see [Opening the Environments Workbench on page 236](#). The Environments Workbench window opens.

2. Open an environment.

The Environment window opens.

3. Select the **Applications** tab.

The **Applications** tab opens.

4. In the **Applications** tab, click **New App**.

The New Application Code window opens.

The screenshot shows the 'New Application Code' dialog box. It features a title bar with a close button. The main area is divided into sections for 'Application Code', 'Application Name', 'Description', 'Server Username', 'Server Password', 'Server Base Path', 'Client Username', 'Client Password', 'Client Base Path', 'DB Username', 'DB Password', and 'DB Link'. The 'Server Base Path' and 'Client Base Path' fields are pre-filled with '/usr/itg'. There are also 'OK', 'Add', and 'Cancel' buttons at the bottom. The status bar at the bottom left indicates 'Ready'.

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 that Mercury Change Management should log on as 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 Mercury Change Management 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.
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	Identifies if this application environment is currently enabled.

6. In the New Application Code window, click **OK**.

The New Application Code window closes. The **Applications** tab opens.

7. In the **Applications** tab, click **OK**.

The changes to the environment are saved.

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. Open the Environments Workbench.

To open the Environments Workbench, see [Opening the Environments Workbench on page 236](#). The Environments Workbench window opens.

2. Open an environment.

The Environment window opens.

3. Select the **Applications** tab.

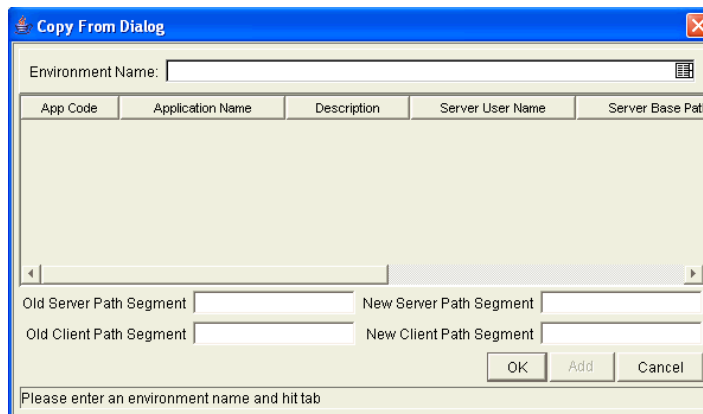
The **Applications** tab opens.

4. In the **Applications** tab, click **New App**.

The New Application Code window opens.

5. In the New Application Code window, click **Copy Apps From**.

The Copy From Dialog window opens.



6. Select the environment from which the applications are to be copied from the Environment Name autocomplete list.

The names of application codes are displayed in the list.

7. To change the base path segment of all the application codes selected, enter the old server and client base path segment and the new server and client base path segment in the appropriate fields.

8. Select all the applications to be copied and 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, suppose that 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, enter **u2** in Old Server Base Path and **u3** in New Server Base Path before copying these applications. Every occurrence of the old server and client base path segment will be changed to the new base path segment in all the applications selected. The changes will be reflected in the applications in the environment into which they were copied.

9. After copying the application codes, any necessary modifications such as adding additional applications, deleting applications, or editing any of the applications can be made.

10. In the **Applications** tab, click **OK**.

The additions to the new environment are saved.

Setting Ownership 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 with the Edit Access access grant for the entity can edit, copy or delete it. Refer to *Security Model Guide and Reference* for more information on access grants.

If a security group is disabled or loses the Edit Access access grant, that group will no longer be able to edit the entity.

Adding Ownerships to Environments

To add an ownership:

1. Open the Environments Workbench.

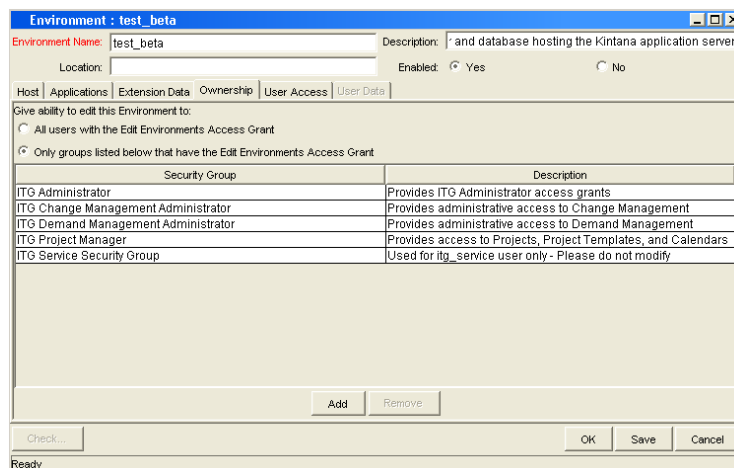
To open the Environments Workbench, see [Opening the Environments Workbench on page 236](#). The Environments Workbench window opens.

2. Open an environment.

The Environment window opens.

3. In the Environment window, click the **Ownership** tab.

The **Ownership** tab opens.



4. In the **Ownership** tab, select the ownership option.

The All user with the Edit Environment access grant option give all users with the Edit Environment access grant ownership of the environment. The Only groups listed below that have the Edit Environments access grant option requires selected groups to be added to the ownership of the environment.

To select ownerships:

- a. In the **Ownership** tab, deselect Only groups listed below that have the Edit Environments.

- b. In the **Ownership** tab, click **Add**.

The Add Security Groups window opens.

- c. In the Add Security Groups window, in the Security Groups field, select the security groups.

The Validate window opens.

- d. In the Validate window, select one or more security groups and click **OK**.

The Validate window closes. The Add Security Groups window lists the selected security groups.

- e. In the Add Security Groups window, click **OK**.

- f. The Add Security Groups window closes. The selected security groups are display in the **Ownership** tab under the Security Group column.

5. In the **Ownership** tab, click **OK**.

The changes to the environment are saved.

Deleting Ownerships from Environments

To delete an ownership:

1. Open the Environments Workbench.

To open the Environments Workbench, see [Opening the Environments Workbench on page 236](#). The Environments Workbench window opens.

2. Open an environment.

The Environments window opens.

3. In the Environments window, click the **Ownership** tab.

The **Ownership** tab opens.

4. In the **Ownership** tab, select an ownership.

The All user with the Edit Environments access grant option give all users with the Edit Environments access grant ownership of the object type. The Only groups listed below that have the Edit Environments access grant option requires selected groups to be added to the ownership of the environment.

5. In the **Ownership** tab, click **Remove**.

The ownership is deleted.

6. In the **Ownership** tab, click **OK**.

The changes to the environment are saved.

Adding Participants to Environments

It is possible to control which users can access an environment for use in environment groups and workflows.

To add participants to the environment:

1. Open the Environments Workbench.

To open the Environments Workbench, see [Opening the Environments Workbench on page 236](#). The Environments Workbench window opens.

2. Open an environment.

The Environment window opens.

3. In the Environment window, click the **User Access** tab.

The **User Access** tab opens.

The screenshot shows the 'Environment : test_beta' window with the 'User Access' tab selected. The 'Environment Name' is 'test_beta' and the 'Description' is 'and database hosting the Kintana application server'. The 'Enabled' checkbox is checked. The 'User Access' tab is active, showing the 'This Environment can be used in Environment Groups and Workflows by:' section. The 'Only users in the groups listed below' radio button is selected. Below this is a table with two columns: 'Security Group' and 'Description'.

Security Group	Description
ITG Change Management Administrator	Provides administrative access to Change Management
ITG Administrator	Provides ITG Administrator access grants
ITG Service Security Group	Used for itg_service user only - Please do not modify

At the bottom of the window, there are 'Add' and 'Remove' buttons, and a 'Check...' button. The status bar at the bottom left says 'Ready'.

4. In the **User Access** tab, in the This Environment can be used in Environment Groups and Workflows by field, select one of the options.

- Select All Users so that all users can edit this environment.
- Select Only Users in the groups listed below so that only the users belonging to a listed security group can edit the environment.

If **Select Only Users** in the groups listed below, is selected:

- a. In the **User Access** tab, click **Add**.

The Add Security Group window opens.

- b. In the Add Security Group window, select the group from the drop-down list.

- c. In the Add Security Group window, click **OK**.

The security group is added to the **User Access** tab.

5. In the **User Access** tab, click **OK**.

The changes to the environment are saved.

Deleting Participants from Environments

To delete participants from the environment:

1. Open the Environments Workbench.

To open the Environments Workbench, see [Opening the Environments Workbench on page 236](#). The Environments Workbench window opens.

2. Open an environment.

The Environment window opens.

3. In the Environment window, click the **User Access** tab.

The **User Access** tab opens.

4. In the **User Access** tab, select a participant to delete.

5. In the **User Access** tab, select, click **Remove**.

The participant is deleted.

6. In the **User Access** tab, click **OK**.

The changes to the environment are saved.

Environment Maintenance and Utilities

This section provides information on validating and maintaining the environment definitions in Mercury Change Management.

Testing Environment Setups

To check the validity of the Environment:

1. Open the Environments Workbench.

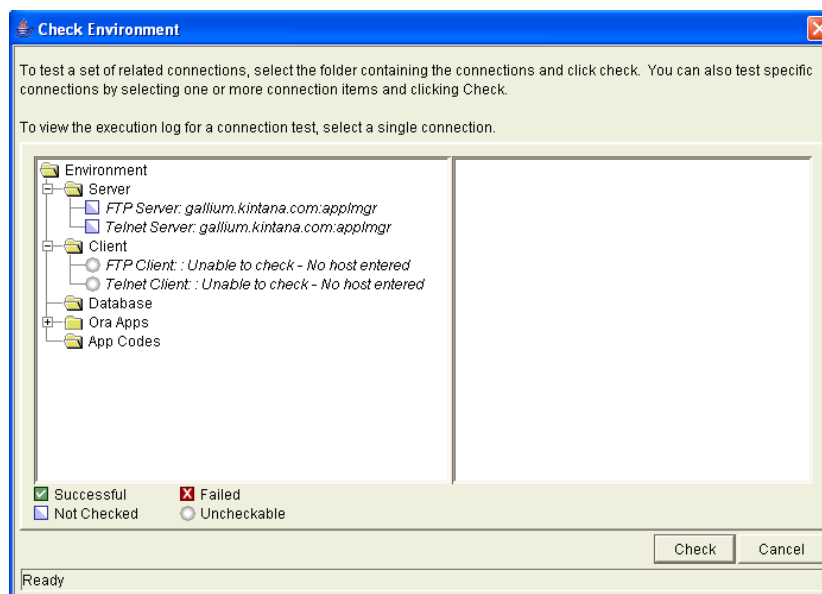
To open the Environments Workbench, see [Opening the Environments Workbench on page 236](#). The Environments Workbench window opens.

2. Open an environment.

The Environment window opens.

3. In the Environment window, click **Check**.

The Check Environment window opens.



4. In the Check Environment window, 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 check boxes by the connection item.

5. In the Check Environment window, click **Check**.

The system verifies the environment definition. Results from the environment check commands are displayed in the Log File area 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 once all the data for it is entered. Additionally, it is good practice to periodically check all environments to catch any obvious problems, such as changed passwords or disabled accounts.

6. In the Check Environment window, click **Cancel**.

The Check Environment window closes.

7. In the Environment window, click **Cancel**.

The Environment window closes.

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. Open the Environments Workbench.

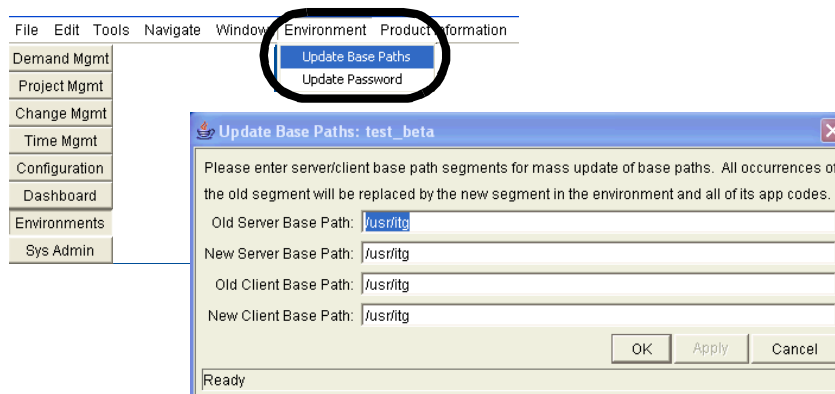
To open the Environments Workbench, see [Opening the Environments Workbench on page 236](#). The Environments Workbench window opens.

2. Open an environment.

The Environment window opens.

3. From the menu, select **Environment > Update Base Paths**.

The Update Base Path window opens.



4. In the Update Base Path window, enter 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. In the Update Base Path window, 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, suppose that 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`, enter **u2** in Old Server Base Path and **u3** in New Server Base Path before copying these applications. Every occurrence of the old server/client base path segment will be changed to the new base path segment in all the applications selected. The changes will be reflected in the applications in the Environment into which they have been copied.

The Environment window opens.

6. In the Environment window, click **OK**.

The changes to the environment are saved.

Environment Password Management Utility

A single user can have access to multiple password protected environments. It is often convenient to use a single username and password for all of the environments that a single user encounters. If the user decided to change their password or if that user's job functions were transferred to another user, it would take hours to update the environment passwords in each environment window.

The Environment Password Management Utility enables a user to update their password in all of 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 Mercury IT Governance Environment interface.

To change a user's environment password:

1. Open the Environments Workbench.

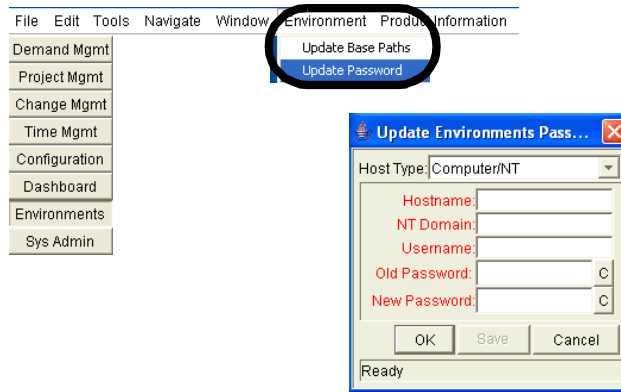
To open the Environments Workbench, see [Opening the Environments Workbench on page 236](#). The Environments Workbench window opens.

2. Open an environment.

The Environment window opens.

3. From the menu, select **Environments > Update Password**.

The Update Environments Password window opens.



4. In the Update Environments Password window, select the Host Type from the drop-down list.

The required fields will change to match requirements of the selected host type.

5. Complete the fields in the Update Environments Password window as specified in the following table:

Fields	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 NT computer.
Username	The username for the schema in the database that will be used by Mercury Change Management 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. In the Update Environments Password window, click **OK**.

The changes are implemented. The Update Environments Password window closes. The Environment window opens.

7. In the Environment window, click **OK**.

The changes to the environment are saved.

Configuring Environment Groups

In This Chapter:

- *Overview of Environment Groups*
 - *Overview of Configuring Environment Groups*
 - *Opening the Environment Group Workbench*
 - *Configuring General Information for Environment Groups*
 - *Creating Environment Groups*
 - *Setting the Order of Executions*
 - *Setting Ownership for Environment Groups*
 - *Adding Ownerships to Environment Groups*
 - *Deleting Ownerships from Environment Groups*
 - *Setting Participants for Environment Groups*
 - *Deleting Participants from Environment Groups*
-

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 it is desirable to execute 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 in one environment group (see [Figure 8-1](#)).

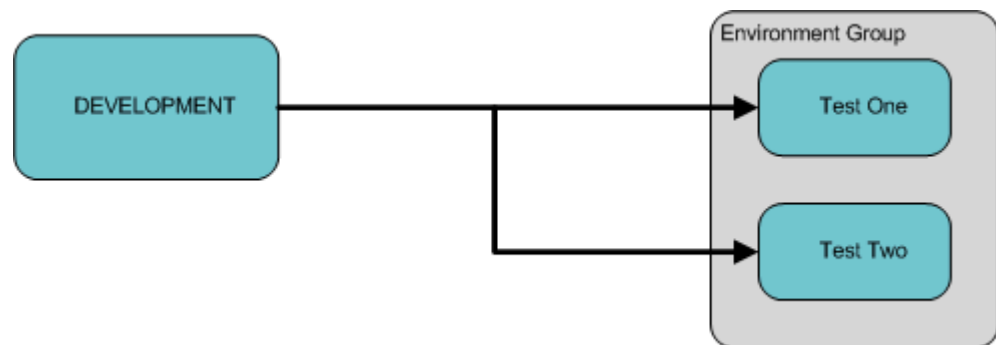


Figure 8-1. Environment group

Overview of Configuring Environment Groups

Environment groups are configured in the Environment Group window. The following lists the main sections found in the Environment Group window:

- **General information.** General information includes basic information concerning the environment group, such as the environment group name and description. See [Opening the Environment Group Workbench on page 262](#).
- **Environments.** The **Environments** tab is used to add existing Mercury Change Management environments to an environment group.
- **Host.** The **Host** tab lists basic information about the client, server, and database for the associated environments. See [Creating Environment Groups on page 264](#).
- **Applications Codes.** The **Applications Codes** tab lists basic information concerning the application codes linked to the associated environments.
- **Serial Execution Order.** The **Serial Execution Order** tab displays the order in which environments are acted on. See [Setting the Order of Executions on page 265](#).
- **Ownership.** Configure who can edit the environment group. See [Setting Ownership for Environment Groups on page 266](#).
- **User Access.** Configures participants of the environment group. Participants can then be given specific access rights to the environment group. See [Setting Participants for Environment Groups on page 269](#).

Opening the Environment Group Workbench

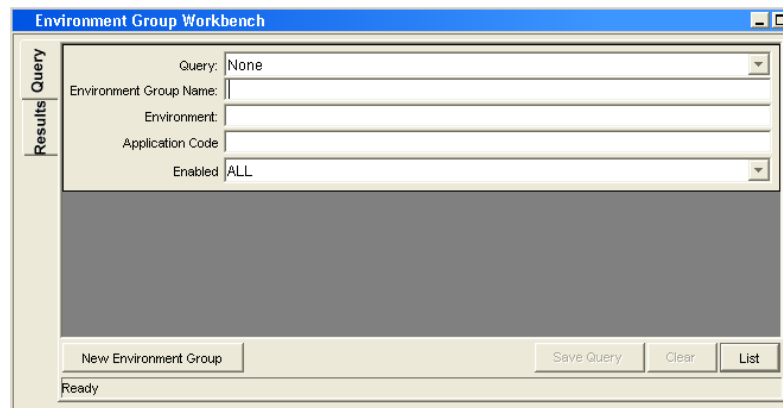
To open the Environment Group Workbench:

1. Log on to the Mercury IT Governance Center.
2. From the menu bar, select **Administration > Open Workbench**.
3. A Workbench status window opens. A few minutes later, a Warning - Security window opens.
4. In the Warning - Security window, select **Yes**.

The Workbench opens.

5. From the shortcut bar, select **Environments > Environment Group**.

The Environment Group Workbench window opens.



For More Information

For information on how to search and select an existing environment group, copy an environment group, and delete an environment group, see *Getting Started*.

Configuring General Information for Environment Groups

To configure the general information of an environment group:

1. Open the Environment Groups Workbench.

To open the Environment Groups Workbench, see [Opening the Environment Group Workbench on page 262](#). The Environment Group Workbench window opens.

2. Open an environment group.

The Environment Group window opens.

3. Complete the fields in the Environment Group window as specified in the following table:

Field	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.
Execution Order	Indicates whether the environments associated with an environment group are executed in parallel order (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. In the Environment Group window, save the changes to the environment.

Click **OK** to save the changes and close the Environment Group window. Click **Save** to save the changes and leave the Environment Group window open. Click **Cancel** to lose the changes and close the Environment Group window.

Creating Environment Groups

To define a new environment group:

1. Open the Environment Groups Workbench.

To open the Environment Groups Workbench, see [Opening the Environment Group Workbench on page 262](#). The Environment Group Workbench window opens.

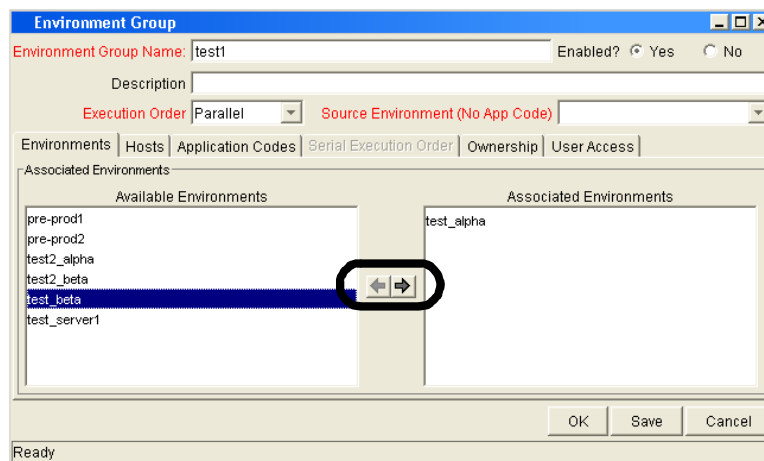
2. Open an environment group.

The Environment Group window opens. The **Environments** tab is displayed.

3. In the **Environments** tab, select an existing environment in the Available Environments field.

4. In the **Environments** tab, click a right-pointing **Arrow** icon.

The selected environment is moved to the Associated Environments field



5. In the **Environments** tab, click **OK**.

The changes to the environment group are saved.

Setting the Order of Executions

The environments are executed in sequential order until all executions have completed. Each environment execution waits for the previous environment execution to complete (success or fail) before beginning.

To set the environment execution order:

1. Open the Environment Groups Workbench.

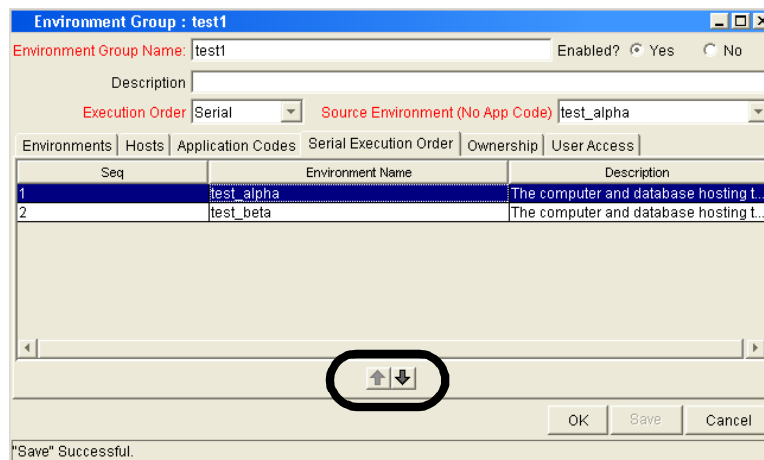
To open the Environment Groups Workbench, see [Opening the Environment Group Workbench on page 262](#). The Environment Group Workbench window opens.

2. Open an environment group.

The Environment Group window opens.

3. In the Environment Group window, click the **Serial Application Order** tab.

The **Serial Application Order** tab opens.



4. In the **Serial Application Order** tab, select a row to be moved.
5. At the bottom of the **Serial Application Order** tab, click the **Up or Down** icons to move the selected environment to a new sequence position.
6. At the bottom of the **Serial Application Order** tab, click **OK**.

The changes to the environment group are saved.

Setting Ownership for Environment Groups

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 with the Edit Access access grant for the entity can edit, copy or delete it. Refer to *Security Model Guide and Reference* for more information on access grants.

If a security group is disabled or loses the Edit Access access grant, that group will no longer be able to edit the entity.

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 the ownership group are the only users who 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. Open the Environment Groups Workbench.

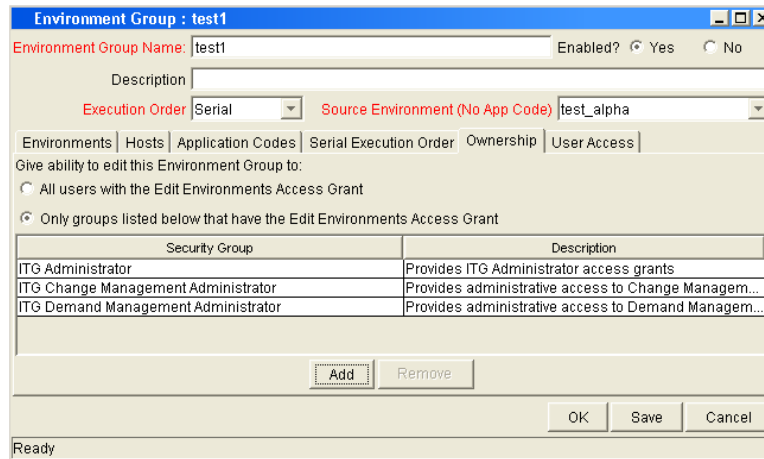
To open the Environment Groups Workbench, see [Opening the Environment Group Workbench on page 262](#). The Environment Group Workbench window opens.

2. Open an environment group.

The Environment Group window opens.

3. In the Environment Group window, click the **Ownership** tab.

The **Ownership** tab opens.



4. In the **Ownership** tab, select the ownership option.

The All user with the Edit Environment Group access grant option give all users with the Edit Environment Group access grant ownership of the environment group. The Only groups listed below that have the Edit Environment Group access grant option requires selected groups to be added to the ownership of the environment group.

To select ownerships:

- a. In the **Ownership** tab, deselect Only groups listed below that have the Edit Environment Group.
- b. In the **Ownership** tab, click **Add**.

The Add Security Groups window opens.

- c. In the Add Security Groups window, in the Security Groups field, select the security groups.

The Validate window opens.

- d. In the Validate window, select one or more security groups and click **OK**.

The Validate window closes. The Add Security Groups window lists the selected security groups.

- e. In the Add Security Groups window, click **OK**.
 - f. The Add Security Groups window closes. The selected security groups are display in the **Ownership** tab under the Security Group column.
5. In the **Ownership** tab, click **OK**.

The changes to the request type are saved.

Deleting Ownerships from Environment Groups

To delete an ownership:

1. Open the Environment Groups Workbench.

To open the Environment Groups Workbench, see [Opening the Environment Group Workbench on page 262](#). The Environment Group Workbench window opens.

2. Open an environment group.

The Environment Group window opens.

3. In the Environment Group window, click the **Ownership** tab.

The **Ownership** tab opens.

4. In the **Ownership** tab, select an ownership.

The All user with the Edit Environment Group access grant option give all users with the Edit Environment Group access grant ownership of the environment group. The Only groups listed below that have the Edit Environment Group access grant option requires selected groups to be added to the ownership of the environment group.

5. In the **Ownership** tab, click **Remove**.

The ownership is deleted.

6. In the **Ownership** tab, click **OK**.

The changes to the environment group are saved.

Setting Participants for Environment Groups

To add participants to the environment:

1. Open the Environment Groups Workbench.

To open the Environment Groups Workbench, see [Opening the Environment Group Workbench on page 262](#). The Environment Group Workbench window opens.

2. Open an environment group.

The Environment Group window opens.

3. In the Environment Group window, click the **User Access** tab.

The **User Access** tab opens.

The screenshot shows the 'Environment Group : test1' window with the 'User Access' tab selected. The 'Enabled?' checkbox is checked. The 'Description' field is empty. The 'Execution Order' is set to 'Serial' and the 'Source Environment (No App Code)' is 'test_alpha'. Below the tabs, there are radio buttons for 'All users' (unselected) and 'Only users in the groups listed below' (selected). A table lists security groups and their descriptions:

Security Group	Description
ITG Change Management Administrator	Provides administrative access to Change Managem...
ITG Demand Management Administrator	Provides administrative access to Demand Managem...
ITG Service Security Group	Used for itg_service user only - Please do not modify

At the bottom of the table are 'Add' and 'Remove' buttons. At the bottom of the window are 'OK', 'Save', and 'Cancel' buttons. The status bar at the bottom left says 'Ready'.

4. In the **User Access** tab, in the This Environment Group can be used in Workflows by field, select one of the options.

- Select All Users so that all users can edit this environment.
- Select Only Users in the groups listed below so that only the users belonging to a listed security group can edit the environment.

If Only Users in the groups listed below, is selected:

- a. In the **User Access** tab, click **Add**.

The Add Security Group window opens.

- b. In the Add Security Group window, select the group from the drop-down list.

- c. In the Add Security Group window, click **OK**.

The security group is added to the **User Access** tab.

5. In the **User Access** tab, click **OK**.

The changes to the environment group are saved.

Deleting Participants from Environment Groups

To delete participants from an environment group:

1. Open the Environment Groups Workbench.

To open the Environment Groups Workbench, see [Opening the Environment Group Workbench on page 262](#). The Environment Group Workbench window opens.

2. Open an environment group.

The Environment Group window opens.

3. In the Environment Group window, click the **User Access** tab.

The **User Access** tab opens.

4. In the **User Access** tab, select a participant to delete and click **Remove**.

The participant is deleted.

5. In the **User Access** tab, click **OK**.

The changes to the environment group are saved.

Chapter

9

Configuring Notification Templates

In This Chapter:

- *Overview of Notification Templates*
 - *Opening the Notification Templates Workbench*
 - *Deleting Notification Templates*
 - *Creating Notification Templates*
 - *Configuring Ownership of Notification Templates*
 - *Deleting Ownerships from Notification Templates*
 - *Configuring Notification Intervals*
 - *Checking the Usage of Notification Templates*
-

Overview of Notification Templates

Notification templates are pre-configured notifications that can be used to quickly construct the body of your message (see *Figure 9-1*). Notification templates are used with the following Mercury IT Governance Center entities:

- Tasks
- Projects
- Requests
- Packages
- Releases
- Workflows
- Reports

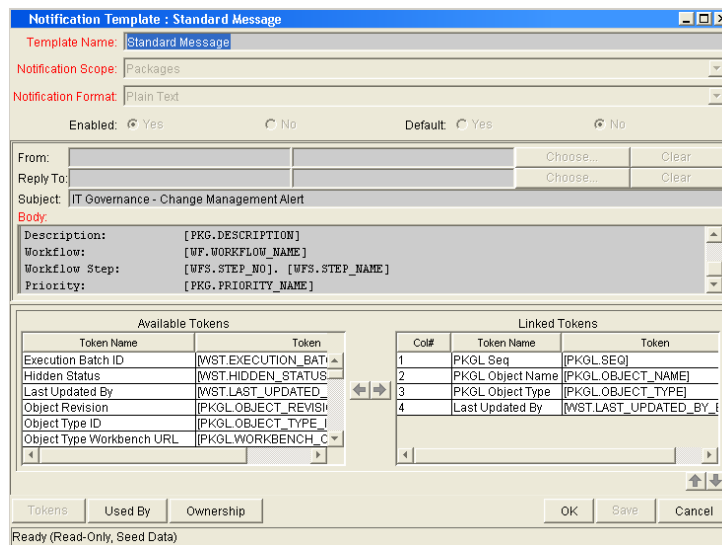


Figure 9-1. Notifications Template window

Opening the Notification Templates Workbench

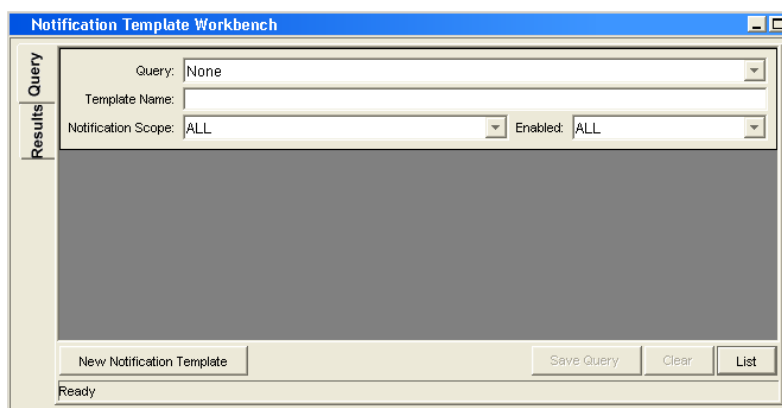
To open the Notification Template Workbench:

1. Log on to the Mercury IT Governance Center.
2. From the menu bar, select **Administration > Open Workbench**.
3. A Workbench status window opens. A few minutes later, a Warning - Security window opens.
4. In the Warning - Security window, select **Yes**.

The Workbench opens.

5. From the shortcut bar, select **Configuration > Notification Templates**.

The Notification Template Workbench window opens.



For More Information

For information on how to search and select an existing notification template, and copy a notification template, see *Getting Started*.

Deleting Notification Templates

You can not delete notification templates that are referenced from an existing notification. To delete a notification template you must first remove these references. Referenced notification templates can be disabled. To see if a notification template is references, see [Checking the Usage of Notification Templates on page 285](#).

For information on how to delete a notification template type, see *Getting Started*.

Creating Notification Templates

To create a new notification template:

1. Open the Notification Template Workbench.

To open the Notification Template Workbench, see [Opening the Notification Templates Workbench on page 273](#). The Notification Template Workbench window opens.

2. Click **New Notification Template**.

A Notification Template window opens.

Token Name	Token
Execution Batch ID	[WST.EXECUTION_BAT]
Hidden Status	[WST.HIDDEN_STATUS]
Last Updated By	[WST.LAST_UPDATED]
Object Revision	[PKGL.OBJECT_REVISION]
Object Type ID	[PKGL.OBJECT_TYPE]
Object Type Workbench URL	[PKGL.WORKBENCH_C]

Col#	Token Name	Token
1	PKGL Seq	[PKGL.SEQ]
2	PKGL Object Name	[PKGL.OBJECT_NAME]
3	PKGL Object Type	[PKGL.OBJECT_TYPE]
4	Last Updated By	[WST.LAST_UPDATED_BY]

3. Complete the fields in the Notification Template window as specified in the following table:

Field	Description
Template Name	Enter the name of the new notification template.
Notification Scope	<p>Include the product area where this notification template will be used. Select an entry from the drop-down list. Entries include:</p> <ul style="list-style-type: none"> • Packages • Projects • Release Distribution • Reports • Request Field Changes • Requests • Task Dates • Task Exceptions <p>The default notification scope is Packages. Selecting another notification scope changes the format of the notification template.</p>
Notification Format	<p>Include the format of the body of the notification. Select an entry from the drop-down list. Entries include:</p> <ul style="list-style-type: none"> • Plain Text • HTML
Enabled	Make the notification template available to the system. Select Yes to make the notification available to the system.
Default	Make the notification template the default notification template for the system. Select Yes to make the notification template the default notification template.

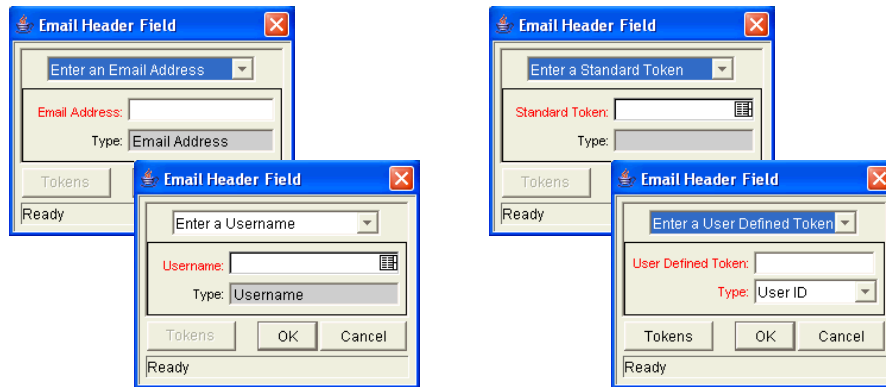
4. In the Notification Template window, enter a From address.

a. In the Notification Template window, in From, click **Choose....**

The Email Header Field window opens.

- b. Select the recipient category from the drop-down list (**Username, Email Address, Standard Token, or User Defined Token**).

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 drop-down list, then it is necessary to enter an Email Address. If a **User Defined Token** is selected, click **Tokens** to bring up a full list of available tokens or type in a specific token.



- c. Enter the appropriate information in the required field.
- d. If a **User Defined Token** has been entered, select the token type that corresponds with the evaluated token value.
- e. In the Email Header Field window, click **OK**.

The Email Header Field window closes.

- 5. In the Notification Template window, enter a Reply address.

- a. In the Notification Template window, in From, click **Choose....**

The Email Header Field window opens.

- b. Select the recipient category from the drop-down list (**Username, Email Address, Standard Token, or User Defined Token**).

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 drop-down list, then it is necessary to enter an Email Address. If a **User Defined Token** is selected, click **Tokens** to bring up a full list of available tokens or type in a specific token.

- c. Enter the appropriate information in the required field.
- d. If a **User Defined Token** has been entered, select the token type that corresponds with the evaluated token value.
- e. In the Email Header Field window, click **OK**.

The Email Header Field window closes.

6. In the Notification Template window, in Body, enter the body of the notification text.

Make sure the format of the body of the notification is the same as specified in Notification Format. HTML notifications for Mercury Change Management should include the token '[NOTIF.NOTIFICATION_DETAILS]' within the **<body>** tags to incorporate linked tokens.

Notification Template : Standard HTML Message

Template Name: Standard HTML Message

Notification Scope: Packages

Notification Format: HTML

Enabled: Yes No Default: Yes No

From: Choose... Clear

Reply To: Choose... Clear

Subject: IT Governance - Change Management Alert

Body:

```
<td colspan="2">[NOTIF.NOTIFICATION_DETAILS]</td>
</table>
```

Use the token [NOTIF.NOTIFICATION_DETAILS] to include an HTML table of linked tokens for associated Package lines.

Available Tokens		Linked Tokens	
Token Name		Col#	Token Name
Execution Batch ID	[WST.EXEC	1	PKGL Seq
Hidden Status	[WST.HIDDI	2	PKGL Object Name
Last Updated By	[WST.LAST	3	PKGL Object Type
Object Revision	[PKGL.OBJ	4	Last Updated By

Tokens Used By Ownership OK Save Cancel

Ready (Read-Only, Seed Data)

7. In the Notification Template window, in Body, add tokens to the body of the text.

If you need to add tokens to the body of the notification template:

- a. At the bottom of the Notification Template window, click **Tokens**.

The Token Builder window opens.

- b. From the Token Builder window, select a token.

- c. In the Token Builder window, in the Token field, copy the name of the token and paste the name in the Body field.

- d. In the Token Builder window, click **Close**.

The Token Builder window closes.

8. In the Notification Template window, configure the ownership of the notification template.

For detailed information on how to configure the ownership of the notification template, see [Configuring Ownership of Notification Templates](#) on page 279.

9. In the Notification Template window, click **OK**.

The changes to the notification template are saved.

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 access grant for the entity can edit, copy or delete it. Refer to *Security Model Guide and Reference* for more information on access grants.

If a security group is disabled or loses the Edit Access access grant, that group will no longer be able to edit the entity.

To configure the ownership of a notification template:

1. Open the Notification Template Workbench.

To open the Notification Template Workbench, see [Opening the Notification Templates Workbench on page 273](#). The Notification Template Workbench window opens.

2. Open a notification template.

A Notification Template window opens.

3. At the bottom of the Notification Template window, click **Ownership**.

The Ownership window opens.

4. In the **Ownership** window, select the ownership option.

- All user with the Edit Notification Template access grant gives all users with the Edit Notification Template access grant can have ownership of the notification template.
- Only groups listed below that have the Edit Notification Template access grant requires selected groups to be added to the ownership of the notification template.

To select ownerships:

- a. In the **Ownership** window, deselect Only groups listed below that have the Edit Notification Template.
- b. In the **Ownership** window, click **Add**.

The Add Security Groups window opens.

- c. In the Add Security Groups window, in the Security Groups field, select the security groups.

The Validate window opens.

- d. In the Validate window, select one or more security groups and click **OK**.

The Validate window closes. The Add Security Groups window lists the selected security groups.

- e. In the Add Security Groups window, click **OK**.

- f. The Add Security Groups window closes. The selected security groups are display in the **Ownership** tab under the Security Group column.

5. In the **Ownership** window, click **OK**.

The changes to the notification template are saved.

Deleting Ownerships from Notification Templates

To delete an ownership:

1. Open the Notification Template Workbench.

To open the Notification Template Workbench, see [Opening the Notification Templates Workbench on page 273](#). The Notification Template Workbench window opens.

2. Open a notification template.

A Notification Template window opens.

3. At the bottom of the Notification Template window, click **Ownership**.

The Ownership window opens.

4. In the **Ownership** window, select an ownership.

The All user with the Edit Notification Template access grant option give all users with the Edit Notification Template access grant ownership of the notification template. The Only groups listed below that have the Edit Notification Template access grant option requires selected groups to be added to the ownership of the notification template.

5. In the **Ownership** window, click **Remove**.

The ownership is deleted.

6. In the **Ownership** window, click **OK**.

The changes to the notification template are saved.

Configuring Notification Intervals

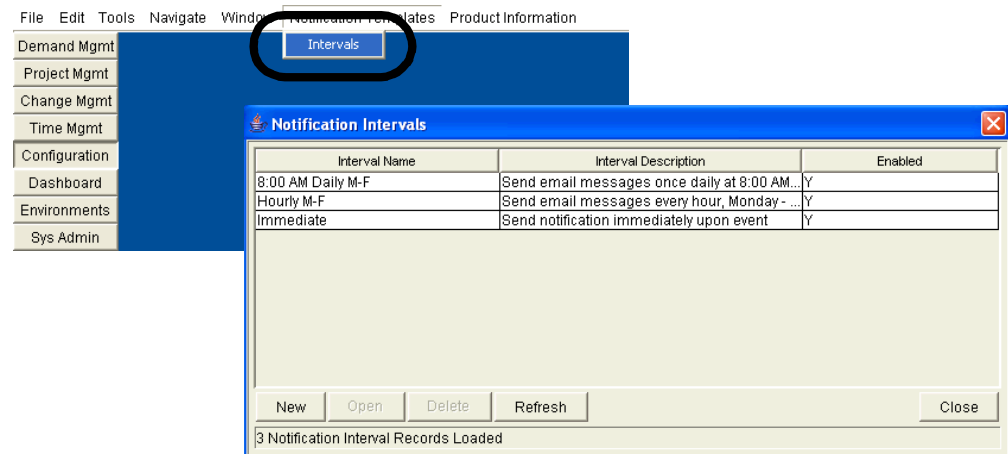
To create a new notification template:

1. Open the Notification Template Workbench.

To open the Notification Template Workbench, see [Opening the Notification Templates Workbench on page 273](#). The Notification Template Workbench window opens.

2. From the menu, select **Notification Templates > Intervals**.

The Notification Intervals window opens.



3. In the Notification Intervals window, click **New**.

The the Notification Intervals window opens.

4. Complete the fields in the **Interval** tab as specified in the following table:

Field Name	Description
Interval Name	This is the name assigned to the interval.
Description	Free form description of this 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	Number of hours to wait after the Start Time or the last batch sent, before sending out the next batch of notifications.
Days	Used to select which days this interval should execute on.
Enabled	If Yes is set, this interval is selectable. If No is set, this interval is unavailable.

5. In the **Interval** tab, click **OK**.

The Notification Interval window closes. The new interval is added to the system.

6. In the Notification Intervals window, click **Close**.

The Notification Intervals window closes. The new notification interval can now be used in any workflow step notification.

When notifications are sent with 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 individual notification appears at the top of the email message in a Summary section.

Checking the Usage of Notification Templates

To check the usage of a notification template:

1. Open the Notification Template Workbench.

To open the Notification Template Workbench, see [Opening the Notification Templates Workbench on page 273](#). The Notification Template Workbench window opens.

2. Open notification template.

A Notification Template window opens.

Chapter 10 Configuring User Data

In This Chapter:

- *Overview of User Data*
 - *Referring to User Data*
 - *Migrating User Data*
 - *Overview of Configuring User Data*
- *Opening the User Data Workbench*
- *Configuring General Information for User Data Types*
- *Creating User Data Fields*
 - *Copying a Field's Definition*
 - *Editing User Data Fields*
 - *Configuring User Data Field Dependencies*
 - *Removing Fields*
- *Configuring User Data Layouts*
 - *Changing Column Widths*
 - *Moving Fields*
 - *Swapping Positions of Two Fields*
 - *Previewing the Layout*
- *Configuring Project and Task User Data Roll-Ups*
 - *Example Using Project and Task User Data Roll-Up*
 - *Overview of Configuring User Data Roll-Ups*
 - *Configuring Task User Data for User Data Roll-Ups*
 - *Configuring Project User Data for User Data Roll-Ups*
 - *Configuring User Data Roll-Ups*
 - *Editing User Data Roll-Ups*
 - *Deleting User Data Roll-Ups*

Overview of 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. For example, you might want to include an additional field on every package. To accomplish this, you would open **Validation Value User Data** and define the extra field. Once defined, the field would appear on a validation’s **User Data** tab.

User data types are configured in the User Data Workbench in the User Data Context window. *Figure 10-1* illustrates a partial list of the available user data types.

The screenshot shows the 'User Data Workbench' window with a table of user data types. The table has columns for 'User Data Type', 'Scope', 'Context Field', 'Context Value', and 'Enabled'. The 'Validation Value User Data' row is highlighted in blue.

User Data Type	Scope	Context Field	Context Value	Enabled
Resource Pool User Data	Global			Y
Security Group User Data	Global			Y
Skill User Data	Global			Y
Staff Prof Line User Data	Global			Y
Staffing Profile User Data	Global			Y
Task User Data	Global			Y
User User Data	Global			Y
Validation Value User Data	Global	Validation Name		Y
Validation Value User Data	Context	Validation Name	CONNECTION_PR...	Y
Validation Value User Data	Context	Validation Name	DATA_MASK	Y
Validation Value User Data	Context	Validation Name	TRANSFER_PROT...	Y
Workflow Step User Data	Global			Y
Workflow User Data	Global			Y

Buttons: New, Open, Copy, Delete, Refresh

32 User Data Context Records are loaded.

Figure 10-1. User data types

Each user data type consists of four components. All four of these components are required to fully identify a user data type. The following lists these components:

- User Data Type.** The User Data Type field lists the name of the user data type. All available user data types are created by Mercury IT Governance Center. You can only define fields for a user data type. You cannot create a new user data type.

- **Scope.** The scope refers to the category of the user data type. There are two available scopes for user data types:
 - **Global.** The standard user data type scope. When Scope is defined as **Global**, every designated entity has the defined field added to the **User Data** tab.
 - **Context.** A context sensitive user data type. When Scope is defined as **Context**, only those entities with the correct Context Field definition and Context Value definition receive the defined user data field.
- **Context Field.** The Context Field is the name of the context sensitive field. The Context Field is only applicable to user data types with a scope of **Context**. There is only one Context Field value available for each user data type. As a result, Context Fields are filled in automatically.
- **Context Value.** The Context Value is the value (context) of the context sensitive field. The **Context Value** is only applicable to user data types with a scope of **Context**. There are multiple, pre-defined values for Context Value. You cannot create a new Context Value, you can only assign an available Context Value.

Mercury IT Governance Center can contain up to twenty user data fields that can be defined. These fields are displayed in the **User Data** tab of the defined entity. The major attributes of each of these fields, such as their graphical presentation, the validation method, and whether or not they are required can be configured.

Referring to User Data

Once a user data field has been created, it is possible to refer to it from other parts of the product by its token name, preceded by the entity abbreviation and the UD qualifier.

Migrating User Data

For any configuration entity with user data fields the data in the user data fields is migrated along with the entity.

- If two instances have identical user data configurations, then the user data will be migrated correctly.
- If two instances do not have identical user data configurations, then the user data will be mapped into the data model according to the storage configuration in the source instance. For this reason, the two instances should be configured with the same user data fields, or the user data should be corrected after migration.
- If the user data is context sensitive, then a corresponding context sensitive configuration must exist in the destination instance, or the migration will fail.
- User data fields that have different hidden and visible values may be problematic. When the hidden value of a user data field refers to a primary key such as, Security Group ID, that can be different in the source and destination instances, then the migrator does not correct the hidden value. The user data should be corrected manually after migration.

Overview of Configuring User Data

The following is a list of the main components of User Data Context window:

- **General information.** General information includes basic information concerning the user data, such as the user data type and the user data context value. See [Configuring General Information for User Data Types on page 292](#).
- **Fields.** The **Fields** tab is used to create additional fields for a user data type. See [Creating User Data Fields on page 295](#).
- **Layout.** Once all of the fields are created for a user data type, the layout of those fields can be configured using the **Layout** tab. See [Configuring User Data Layouts on page 306](#).

Opening the User Data Workbench

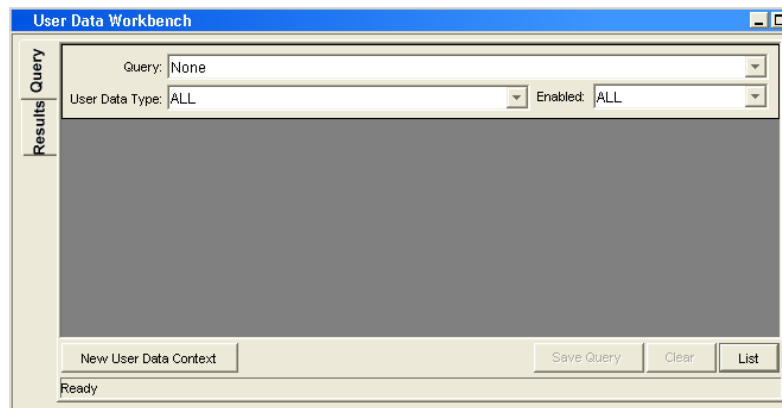
To open the User Data Workbench:

1. Log on to the Mercury IT Governance Center.
2. From the menu bar, select **Administration > Open Workbench**.
3. A Workbench status window opens. A few minutes later, a Warning - Security window opens.
4. In the Warning - Security window, select **Yes**.

The Workbench opens.

5. From the shortcut bar, select **Configuration > User Data**.

The User Data Workbench window opens.



For More Information

For information on how to search and select an existing user data, copy user data, and delete user data, see *Getting Started*.

Configuring General Information for User Data Types

To configure the general information for a user data type:

1. Open the User Data Workbench.

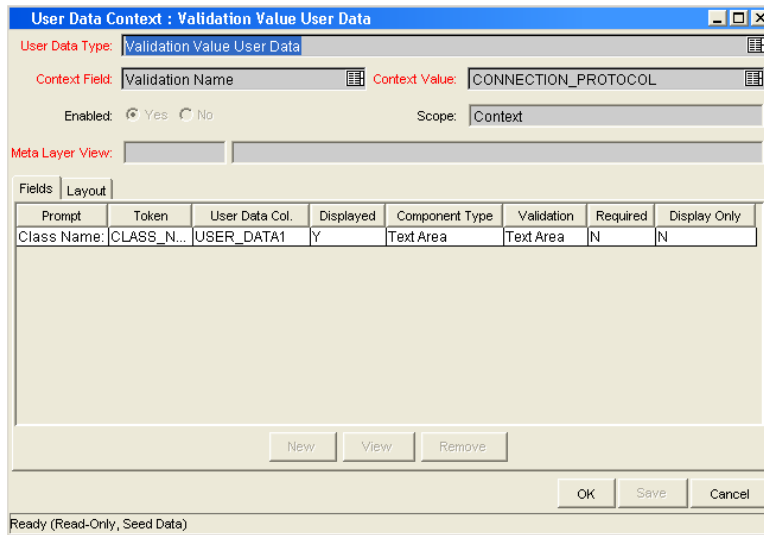
To open the User Data Workbench, see [Opening the User Data Workbench on page 291](#). The User Data Workbench window opens.

2. Select an existing user data type or create a new user data type.

When configuring a global user data type, you must open an existing user data type. For example, Skill User Data is already created by Mercury IT Governance Center.

When configuring a context sensitive user data type, you can select an existing context sensitive user data type or click **New** to create a context sensitive user data type.

The User Data Context window opens.



3. In the User Data Context window, complete the fields in the User Data Context window as specified in the following table:

Field	Description
User Data Type	<p>Selects the name of the user data type.</p> <p>For global user data types, this field is automatically populated.</p> <p>For context sensitive user data types, select the context sensitive user data type from the drop-down list. You can choose one of the following context sensitive user data types:</p> <ul style="list-style-type: none"> • Package User Data • Validation Value User Data
Context Field	<p>The name of the context sensitive field. This field is disabled for user data types where Scope = Global. This field is automatically filled in for context sensitive user data. The following lists the User Data Types and the Context Field:</p> <ul style="list-style-type: none"> • Package User Data - Priority • Validation Value User Data - Validation Name
Context Value	<p>Selects the value for the Context Field. This field is disabled for user data types where Scope = Global. For context sensitive user data types, select the context value from the drop-down list. Only one Context Value can be defined at a time. For example, you cannot have two context sensitive user data types with the same Context Field and Context Value (such as Priority = Critical).</p>
Enable	<p>Indicates whether or not the user data type is available to Mercury IT Governance Center.</p>

Field	Description
Scope	<p>The category of user data type. This field is automatically filled in based on the user data type. The possible scopes for a user data type are:</p> <ul style="list-style-type: none"> • Global. The standard user data type scope. When Scope is defined as Global, every designated entity has the defined field added to the User Data tab. • Context. A context sensitive user data type. When Scope is defined as Context, only those entities with the correct Context Field definition and Context Value definition receive the defined user data field.
Meta Layer View	<p>Meta layer views relate information specific Mercury IT Governance Center. 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.</p>

4. Save the changes to the user data type.

Click **OK** to save the changes and close the User Data Context window. Click **Save** to save the changes and leave the User Data Context window open. Click **Cancel** to lose the changes and close the User Data Context window.

Creating User Data Fields

To create a new user data field:

1. Open the User Data Workbench.

To open the User Data Workbench, see [Opening the User Data Workbench on page 291](#). The User Data Workbench window opens.

2. Select an existing user data type or create a new user data type.

The User Data Context window opens. The **Fields** tab is displayed.

3. In the **Fields** tab, click **New**.

The Field window opens.

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

Field	Description
Field Prompt	The prompt visible for the user data field in the request.
Token	An uppercase text string used to identify this field. The token name must be unique for the specific user data. An example of a token name is ASSIGNED_TO_USER_ID.
Description	A description of the user data field.
Enabled	Indicates whether or not the field is turned on for this user data.

Field	Description
Validation	Indicates the validation logic to determine the valid values for this field. This could be a list of user-defined values, a rule that the result has to be a number, and so on.
Component Type	Defines the visual characteristics of the field (drop-down list, free form text field, and so on.). This is derived from the validation chosen. This field cannot be edited.
Multiselect	Indicates whether or not the field allows users to select more than one entry. Only valid for fields with an autocomplete component for the validation.

5. In the Field window, click the **Attributes** tab.

6. Complete the fields in the **Attributes** tab as specified in the following table:

Field	Description
User Data Col	Indicates the internal column that the field value will be stored in. These values will 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, therefore allowing up to 20 fields. No two fields in user data can use the same column.
Display Only	Indicates whether the field is only displayed and cannot be updated. Select Use Dependency Rules to use the logic defined in the Dependencies tab.
Display	Indicates if the user sees this field in the User Data tab.
Required	Indicates if the user is required to specify a value for this field. Select Use Dependency Rules to use the logic defined in the Dependencies tab.

7. In the Field window, click the **Defaults** tab.

8. Complete the fields in the **Defaults** tab as specified in the following table:

Field	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 entered 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 entered or updated in the source field, it will automatically be entered or updated in this destination field.

9. In the Field window, click the **Dependencies** tab.

10. Complete the fields in the **Dependencies** tab as specified in the following table:

Field	Description
Clear When _ __ 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 drop-down list containing 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 drop-down list containing logical qualifiers, and a text field. To use this functionality, select Use Dependency Rules in the Attributes tab.

11. In the Field window, click the **Security** tab to define which users can view or update this field.

Enter the information as follows:

- a. In the **Security** tab, click **Edit**.
- b. Complete the fields in the Edit Field Security window as specified in the following table:

Field / Button	Description
Visible to all users	<p>Checking this checkbox allows all users to see the field. If this checkbox 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 checkbox is not checked, the Select User/ Security Group that can view this field area is activated.</p> <p>Deselecting the Visible to all users or Editable by all users checkboxes enables the Select Users/Security Groups that can view this field area of the Edit Field Security window.</p>
Editable by all users	<p>Checking this checkbox allows all users to edit the field. If this checkbox is not checked, you can set who can edit the field. The default is for all user to be able to edit a field.</p> <p>De-selecting the Visible to all users or Editable by all users checkboxes enables the Select Users/Security Groups that can view this field area of the Edit Field Security window.</p>
Enter a Security Group (drop-down list)	<p>To select the format for specifying users to grant visibility and edit permission, use the Enter a Security Group drop-down list. The drop-down lists the formats to choose users. The drop-down list dynamically updates the Security Group Validate autocomplete window list.</p> <p>The choices are:</p> <ul style="list-style-type: none"> ● Enter a Username. Select a specific user a to see and/or edit the field. The user must have an email address. ● Enter a Security Group. Select a specific security group to see and/or edit the field. ● Enter a Standard Token. Select a standard token to see and/or edit the field. ● Enter a User Defined Token. Select a user defined token to see and/or edit the field. Selecting the Enter a User Defined Token format enables the Tokens button. <p>Selecting an item from the Enter a Security Group drop-down list dynamically updates the Security Group field.</p>
Security Group	<p>Provides a field for specifying the recipient. If the Enter a Security Group drop-down list is:</p> <ul style="list-style-type: none"> ● 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.

12. In the Edit Field Security window, click **OK**.

The Edit Field Security window closes. The new field appears in the Field window.

13. In the Field window, click **OK**.

The changes to the User Data Context field are saved.

Copying a Field's Definition

The **Copy From** functionality can also be utilized to streamline the process of adding fields by copying the definition of existing fields.

To copy a field's definition:

1. Open the User Data Workbench.

To open the User Data Workbench, see [Opening the User Data Workbench on page 291](#). The User Data Workbench window opens.

2. Select an existing user data type or create a new user data type.

The User Data Context window opens. The **Fields** tab is displayed.

3. In the **Fields** tab, click **New**.

The Field window opens.

4. In the **Fields** tab, click **New**.

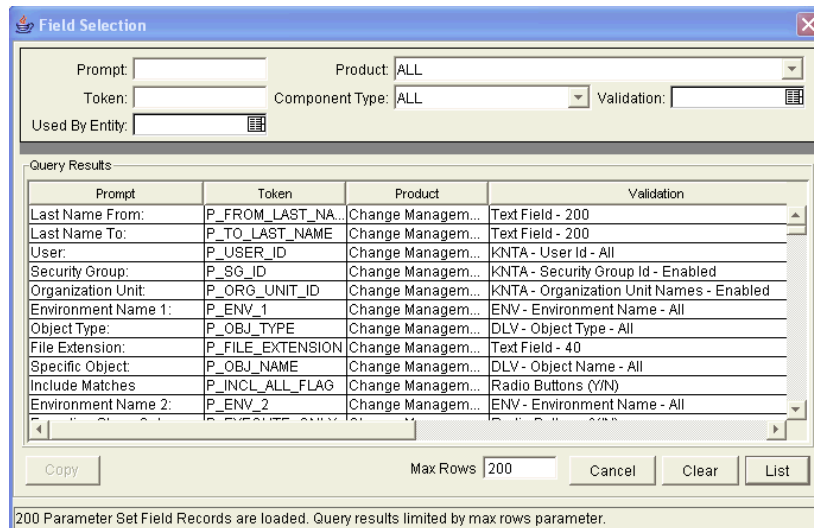
The Field window opens.

5. Click **Copy From**.

The Field Selection window opens.

6. In the Field Selection window, complete the fields and click **Copy From**.

The Field Selection window refreshes with fields matching the search criteria.



7. In the Field Selection window, select a field to copy.

Query fields by a number of criteria, such as the token name or field prompt. It is also possible to perform more complex queries such as listing all fields that reference a certain validation or are used by a certain entity.

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

This closes the window and copies the definition of the selected field into the Field window.

9. In the Field window, make any necessary modifications and click **OK**.

The changes to the user data type are saved.

Editing User Data Fields

To edit an existing field:

1. Open the User Data Workbench.

To open the User Data Workbench, see [Opening the User Data Workbench on page 291](#). The User Data Workbench window opens.

2. Select an existing user data type or create a new user data type.

The User Data Context window opens. The **Fields** tab is displayed.

- In the User Data Context window, select the field and click **Edit**.

The Field window opens.

The screenshot shows the 'Field: New' dialog box. It has a title bar with 'Field: New' and a close button. The main area contains several sections: 'Field Prompt:' and 'Token:' text boxes; a 'Description:' text box; 'Enabled:' radio buttons for 'Yes' (selected) and 'No'; a 'Validation' section with a text box and 'New'/'Open' buttons; a 'Component Type:' dropdown menu set to 'None'; 'Multiselect:' radio buttons for 'Yes' and 'No' (selected); a header region with tabs for 'Attributes', 'Default', and 'Dependencies'; 'User Data Col:' dropdown set to 'USER_DATA1'; 'Display Only:' dropdown set to 'Never'; 'Display:' radio buttons for 'Yes' (selected) and 'No'; 'Required:' dropdown set to 'Never'; a 'Copy From...' button; and 'OK', 'Add', and 'Cancel' buttons. The status bar at the bottom says 'Ready'.

- In the Field window, make the desired changes in the header region, **Attributes** tab, **Default** tab, and **Dependencies** tab.

For information concerning the **Attributes** tab, **Default** tab, and **Dependencies** tab, see [Creating User Data Fields on page 295](#).

- In the Field window, click **OK**.

The Field window closes. The User Data Context field opens.

- In the User Data Context window, click **OK**.

The changes to the user data type are saved.

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**.

A field can be configured to:

- Clear when another field changes.
- Become read only when another field meets a logical condition, defined in [Table 10-1](#).
- Become required when another field meets a logical condition, defined in [Table 10-1](#).

Table 10-1. Field dependencies

Logical qualifier	Description
like	A like condition looks for close matches of the value to the contents of the field chosen.
not like	A not like condition looks for contents in the selected field that are not close matches to the Value field.
is equal to	An is equal to condition looks for an exact match of the Value to the contents of the Field chosen.
is not equal to	An is not equal to condition is true when there are no results exactly matching the value of the field contents.
is null	An is null condition is true when the field selected is blank.
is not null	An is not null condition is true when the field selected is not blank.
is greater than	An is greater than condition looks for a numerical value larger than the value entered in the Value field.
is less than	An is less than condition looks for a numerical value below the value entered in the Value field.
is less than equal to	An is less than equal to condition looks for a numerical value below or the same as the value entered in the Value field.
is greater than equal to	An is greater than equal to condition looks for a numerical value larger than or the same as the value entered in the Value field.

To configure a user data field dependency:

1. Open the User Data Workbench.

To open the User Data Workbench, see [Opening the User Data Workbench on page 291](#). The User Data Workbench window opens.

2. Select an existing user data type or create a new user data type.

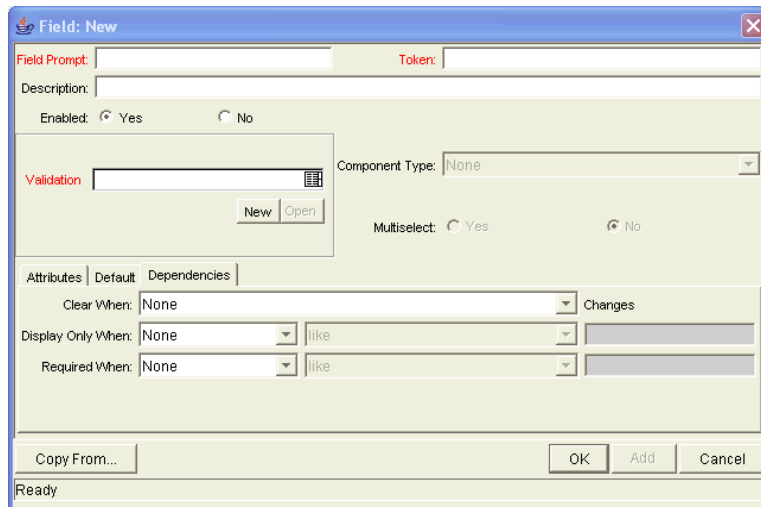
The User Data Context window opens. The **Fields** tab is displayed.

3. In the User Data Context window, select the field and click **Edit**.

The Field window opens.

4. In the Field window, click the **Dependencies** tab.

The **Dependencies** tab opens.



The screenshot shows the 'Field: New' dialog box with the 'Dependencies' tab selected. The dialog has a title bar with a close button. The main area contains several fields and controls:

- Field Prompt:** A text input field.
- Token:** A text input field.
- Description:** A text input field.
- Enabled:** Radio buttons for 'Yes' (selected) and 'No'.
- Validation:** A text input field with a list icon and 'New' and 'Open' buttons below it.
- Component Type:** A dropdown menu set to 'None'.
- Multiselect:** Radio buttons for 'Yes' and 'No'.
- Attributes | Default | Dependencies:** A tabbed interface with 'Dependencies' selected.
- Clear When:** A dropdown menu set to 'None'.
- Changes:** A dropdown menu set to 'like'.
- Display Only When:** A dropdown menu set to 'None'.
- Required When:** A dropdown menu set to 'None'.
- Buttons:** 'Copy From...', 'OK', 'Add', and 'Cancel'.
- Status:** 'Ready'.

5. In the **Dependencies** tab, set the field dependencies. It is possible to:
 - Select a field name from the Clear When drop-down list to indicate that the current field should be cleared when the selected field changes.
 - Select a field name from the Display Only When drop-down list to indicate that the current field should for display only (for example, not editable) when certain logical criteria are satisfied. This field functions with two adjacent fields. These are a drop-down list containing logical qualifier and another field which dynamically changes to a date field, drop-down list, or text field, depending on the selected field's validation.
 - Select a field name from the Required When drop-down list to indicate that the current field should be required when certain logical criteria are satisfied. This field functions with two adjacent fields. These are a drop-down list containing logical qualifier and another field which dynamically changes to a date field, drop-down list, or text field, depending on the selected field's validation.

6. In the **Dependencies** tab, click **OK**.

The **Dependencies** tab closes. The Field window opens.

7. In the Field window, click **OK**.

The Field window closes. The User Data Context window opens.

8. In the User Data Context window, click **OK**.

The changes to the user data type are saved.

Removing Fields

To remove a field permanently from a user data type:

1. Open the User Data Workbench.

To open the User Data Workbench, see [Opening the User Data Workbench on page 291](#). The User Data Workbench window opens.

2. Select an existing user data type or create a new user data type.

The User Data Context window opens. The **Fields** tab is displayed.

3. In the User Data Context window, select the field and click **Remove**.

The row is removed.

4. In the User Data Context window, click **OK**.

The changes to the user data type are saved.

Configuring User Data Layouts

The layout of user data fields can be changed in the **Layout** tab of the User Data Context window.

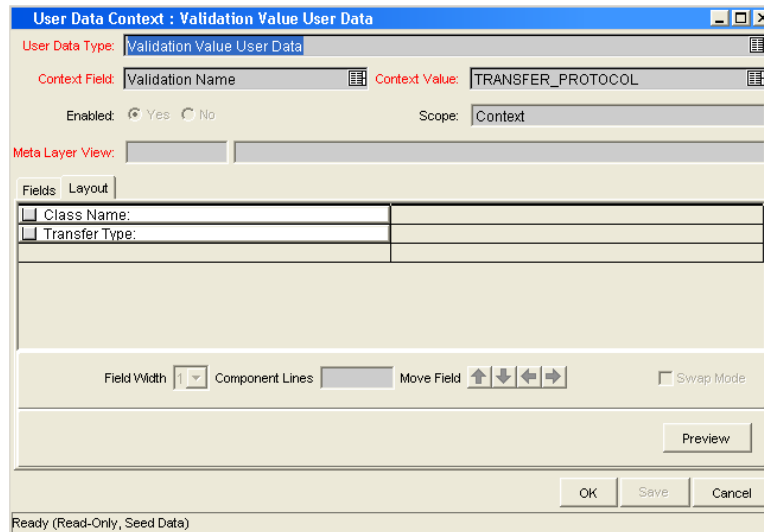


Figure 10-2. User Data window Layout tab

Changing Column Widths

To change the column width of a field:

1. Open the User Data Workbench.

To open the User Data Workbench, see [Opening the User Data Workbench on page 291](#). The User Data Workbench window opens.

2. Select an existing user data type or create a new user data type.

The User Data Context window opens. The **Fields** tab is displayed.

3. In the User Data Context window, click the **Layout** tab.

The **Layout** tab opens.

4. In the **Layout** tab, select the field.

5. In the **Layout** tab, in Field Width, select either **1** or **2** inches.

The Layout editor will not allow changes to be made if it conflicts with another field in the layout (for example, a field's width cannot be changed from one to two if another field exists in column two on the same row).

Additionally, for fields of component type **Text Area**, it is possible to 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 will be blank and non-updateable.

6. In the **Layout** tab, click **OK**.

The changes to the user data type are saved.

Moving Fields

To move a field or a set of fields:

1. Open the User Data Workbench.

To open the User Data Workbench, see [Opening the User Data Workbench on page 291](#). The User Data Workbench window opens.

2. Select an existing user data type or create a new user data type.

The User Data Context window opens. The **Fields** tab is displayed.

3. In the User Data Context window, click the **Layout** tab.

The **Layout** tab opens.

4. In the **Layout** tab, select the field.

To select more than one field, press the Shift key while selecting the last field in a set. It is only possible to select a continuous set of fields.

A field, or a set of fields, cannot be moved to an area where other fields already exist. Those other fields must be moved out of the way first.

5. At the bottom of the **Layout** tab, use the **Arrow** icons to move the fields to the desired location in the layout builder.

6. In the **Layout** tab, click **OK**.

The changes to the user data type are saved.

Swapping Positions of Two Fields

To swap the positions of two fields:

1. Open the User Data Workbench.

To open the User Data Workbench, see [Opening the User Data Workbench on page 291](#). The User Data Workbench window opens.

2. Select an existing user data type or create a new user data type.

The User Data Context window opens. The **Fields** tab is displayed.

3. In the User Data Context window, click the **Layout** tab.

The **Layout** tab opens.

4. In the **Layout** tab, select the field.

5. In the **Layout** tab, select the Swap Mode check box.

This causes an **S** to appear in the check box area of the selected field.

6. Once the **S** appears, double-click on the field to be swapped with.

This causes the two fields to change positions. Following the swap, the swap mode is turned off.

7. In the **Layout** tab, click **OK**.

The changes to the user data type are saved.

Previewing the Layout

You can check to see what the layout will look like in actual use. In the User Data Content window, in the **Layout** tab click **Preview**. This opens a small window that shows the fields as they will appear in the window, shown in *Figure 10-3*.

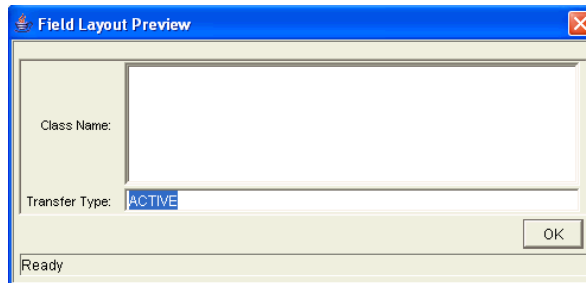


Figure 10-3. Preview mode

If all fields have a width of one column, all displayed columns will automatically span the entire available area when an entity of the given user data is being viewed or generated.

Non-displayed fields do not affect the layout. The layout engine considers them the same as a blank field.

Configuring Project and Task User Data Roll-Ups



Note

The scope of this section is limited to Project Management.

Values from Task User Data fields can be configured to roll-up (combine and process values in a meaningful way) into parent Project User Data fields. The following types of task user data can roll up into project user data:

- Numeric fields (Text field component type with numeric data mask)
- Date fields

For each project, a Project User Data field can show a roll-up of Task User Data values using one of the following methods:

- **Average.** Shows the average of all values of a specified Task User Data field for every task under the project (numeric fields).
- **Maximum.** Shows the largest of all values of a specified Task User Data field for every task under the project (numeric and date fields).
- **Minimum.** Shows the smallest of all values of a specified Task User Data field for every task under the project (numeric and date fields).
- **Sum.** Shows the summation of all values of a specified Task User Data field for every task under the project (numeric fields).

Project and task user data roll-up can be used to capture various important aspects of a project. For example:

- Using the **Average** roll-up method, the average cost of all a project's tasks can be easily determined and automatically recalculated each time a task is updated.
- Using the **Maximum** roll-up method, the latest date out of a project's tasks can be captured.
- Using the **Minimum** roll-up method, the earliest date out of a project's tasks can be captured.
- Using the **Sum** roll-up method, the total cost of a project's tasks can be easily determined and automatically recalculated each time a Task is updated.

Example Using Project and Task User Data Roll-Up

A company needs to capture the total cost for a testing project. Total project cost in this case is to be calculated by adding the costs of individual tasks. User data fields for task cost and project total cost are each defined. The relationship is illustrated in *Figure 10-4*.

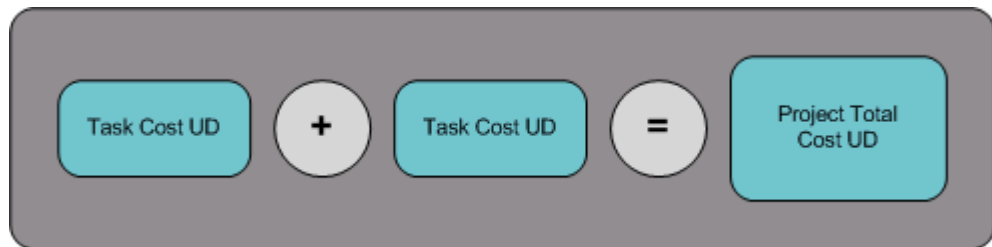


Figure 10-4. Project and task example

Each task has its own Cost User Data field (Task Cost). The values for each Task Cost User Data field are rolled up using the **Sum** roll-up method into the Project Total Cost User Data field (Total Cost). *Figure 10-4* illustrates the project's **User Data** tab and two of the project's task **User Data** tabs.

The screenshot illustrates the roll-up process. It shows three windows from a software application. The top window is titled 'Task Information: Phase One Testing' and shows a 'Task Cost' field with the value '18'. The middle window is titled 'Task Information: Phase Two Testing' and shows a 'Task Cost' field with the value '12'. A plus sign '+' is placed to the left of the middle window. Below these two windows is a thick horizontal line. The bottom window is titled 'Project Information: Rollup Project' and shows a 'Total Cost' field with the value '30'. An equals sign '=' is placed to the left of the bottom window. The 'Task Cost' and 'Total Cost' fields in all three windows are circled in black.

Overview of Configuring User Data Roll-Ups

User Data must be configured for the project and tasks before specifying user data roll-up methods. The following lists the main steps required to configure user data roll-ups:

To configure user data roll-ups:

1. Configure the Task User Data field.
2. Configure the Project User Data field.
3. Configure the User Data Roll-Up Method.

Configuring Task User Data for User Data Roll-Ups

Only two User Data fields of the same type can be selected for user data roll-up. For example, a Numeric text field cannot roll up into a Date field.

While a Task User Data field can have multiple user data roll-up relationships associated with it, a Project User Data field can have only one user data roll-up relationship defined.

To configure task user data for user data roll-ups:

1. Open the User Data Workbench.

To open the User Data Workbench, see [Opening the User Data Workbench on page 291](#). The User Data Workbench window opens.

2. From the User Data Workbench, open Task User Data.

The User Data Context window opens. The **Fields** tab is displayed.

3. In the **Fields** tab, select **New**.

The **Fields** window opens.

User Data Context : Validation Value User Data

User Data Type: Validation Value User Data

Context Field: Validation Name Context Value: CONNECTION_PROTOCOL

Enabled: Yes No Scope: Context

Meta Layer View:

Prompt	Token	User Data Col.	Displayed	Component Type	Validation	Required	Display Only
		USER_DATA1	Y	Text Area	Text Area	N	N

New View Remove

OK Save Cancel

Ready (Read-Only, Seed Data)

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

Field	Description
Field Prompt	The prompt visible for the user data field in the request. For example: Task Cost.
Token	An uppercase text string used to identify this field. The token name must be unique for the specific user data. An example of a token name is USER_DATA_TASK_COST.
Description	A description of the user data field.
Enabled	Select Yes .
Validation	Selects the validation. The validation must be the same as the validation for the Project User Data field. The validation must be one of the following: <ul style="list-style-type: none"> Numeric fields (Text field component type with numeric data mask) Date fields
Component Type	Automatically set by the validation type.
Multiselect	Automatically set by the validation type.

5. In the Field window, select **OK**.

The Field window closes. The new field is added to the User Data Context window.

6. In the User Data Context window, click **OK**.

The changes to the user data type are saved.

Configuring Project User Data for User Data Roll-Ups

Only two User Data fields of the same type can be selected for user data roll-up. For example, a Numeric text field cannot roll up into a Date field.

While a Task User Data field can have multiple user data roll-up relationships associated with it, a Project User Data field can have only one user data roll-up relationship defined.

To configure project user data for user data roll-ups:

1. Open the User Data Workbench.

To open the User Data Workbench, see [Opening the User Data Workbench on page 291](#). The User Data Workbench window opens.

2. From the User Data Workbench, open Project User Data.

The User Data Context window opens. The **Fields** tab is displayed.

3. In the **Fields** tab, select **New**.

The **Fields** window opens.

Prompt	Token	User Data Col.	Displayed	Component Type	Validation	Required	Display Only
CLASS_N...	USER_DATA1	Y	Text Area	Text Area	N	N	

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

Field	Description
Field Prompt	The prompt visible for the user data field in the request. For example: Total Cost.
Token	An uppercase text string used to identify this field. The token name must be unique for the specific user data. An example of a token name is USER_DATA_PROJECT_TOTAL_COST.
Description	A description of the user data field.
Enabled	Select Yes .
Validation	Selects the validation. The validation must be the same as the validation for the Task User Data field. The validation must be one of the following: <ul style="list-style-type: none"> Numeric fields (Text field component type with numeric data mask) Date fields
Component Type	Automatically set by the validation type.
Multiselect	Automatically set by the validation type.

5. In the Field window, select **OK**.

The Field window closes. The new field is added to the User Data Context window.

6. In the User Data Context window, click **OK**.

The changes to the user data type are saved.

Configuring User Data Roll-Ups

Only two User Data fields of the same type can be selected for user data roll-up. For example, a Numeric text field cannot roll up into a Date field.

While a Task User Data field can have multiple user data roll-up relationships associated with it, a Project User Data field can have only one user data roll-up relationship defined.

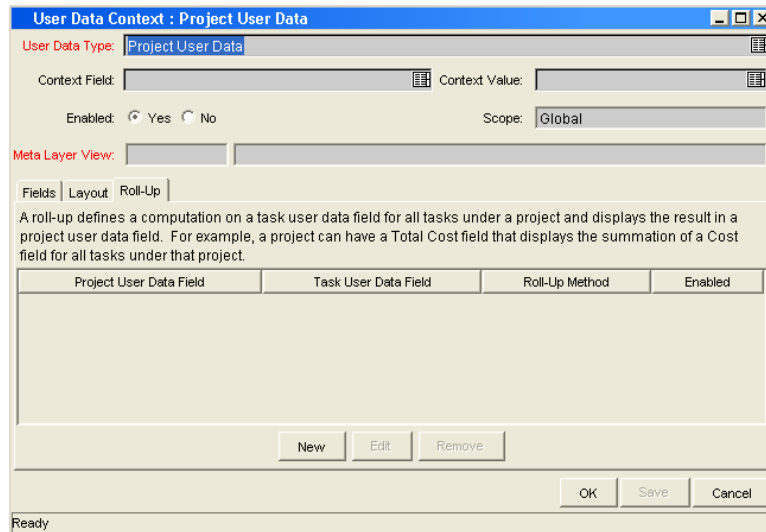
To configure user data roll-ups:

1. Open the User Data Workbench.

To open the User Data Workbench, see [Opening the User Data Workbench on page 291](#). The User Data Workbench window opens.

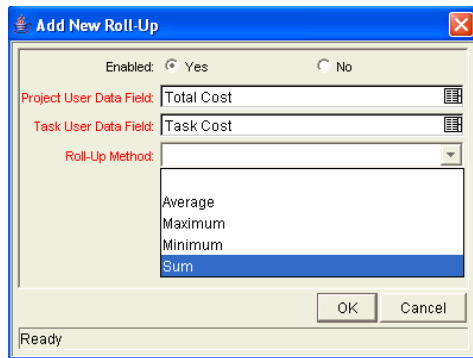
2. From the User Data Workbench, open Project User Data.
3. In the Project User Data window, select the **Roll-Up** tab.

The **Roll-Up** tab opens.



4. In the **Roll-Up** tab, click **New**.

The Add New Roll-Up window opens.



5. Complete the fields in the Add New Roll-Up window as specified in the following table:

Field	Description
Enabled	Makes the user data roll-up method available to the system. Yes makes the user data roll-up method available to the system.
Project User Data Field	<p>Selects from the available Project User Data fields. The Project User Data field must already exist. The Project User Data field validation type must be the same as the Task User Data field validation type. For example,</p> <ul style="list-style-type: none"> • Numeric Task User Data fields cannot roll-up to a date Project User Data field. • Date Task User Data fields cannot roll-up to a numeric Project User Data field.
Task User Data Field	<p>Selects from the available Task User Data fields. The Task User Data field must already exist. The Task User Data field validation type must be the same as the Project User Data field validation type. For example,</p> <ul style="list-style-type: none"> • Numeric Task User Data fields cannot roll-up to a date Project User Data field. • Date Task User Data fields cannot roll-up to a numeric Project User Data field.
Roll-Up Method	<p>Selects the method of the user data roll-up. The following lists the types of user data roll-up:</p> <ul style="list-style-type: none"> • Average. Shows the average of all values of a specified Task User Data field for every task under the project (numeric fields). The output is displayed in the specified Project User Data field. • Maximum. Shows the largest of all values of a specified Task User Data field for every task under the project (numeric and date fields). The output is displayed in the specified Project User Data field. • Minimum. Shows the smallest of all values of a specified Task User Data field for every task under the project (numeric and date fields). The output is displayed in the specified Project User Data field. • Sum. Shows the summation of all values of a specified Task User Data field for every task under the project (numeric fields). The output is displayed in the specified Project User Data field.

6. In the Add New Roll-Up window, click **OK**.

The Add New Roll-Up window closes. The user data roll-up relationship is added to the **Roll-Up** tab.

7. In the **Roll-Up** tab, click **Save**.

The changes to the user data type are saved.

Editing User Data Roll-Ups

To edit an existing user data roll-up:

1. Open the User Data Workbench.

To open the User Data Workbench, see [Opening the User Data Workbench on page 291](#). The User Data Workbench window opens.

2. From the User Data Workbench, open Project User Data.

3. In the Project User Data window, select the **Roll-Up** tab.

The **Roll-Up** tab opens.

4. In the **Roll-Up** tab, select the user data roll-up and click **Edit**.

The Edit New Roll-Up window opens.

5. In the Edit New Roll-Up window, edit the user data roll-up.

For details on the fields of the Edit New Roll-up window, see [Configuring User Data Roll-Ups on page 316](#).

6. In the Add New Roll-Up window, click **OK**.

The Add New Roll-Up window closes.

7. In the **Roll-Up** tab, click **Save**.

The changes to the user data type are saved.

Deleting User Data Roll-Ups

To delete an existing user data roll-up:

1. Open the User Data Workbench.

To open the User Data Workbench, see [Opening the User Data Workbench on page 291](#). The User Data Workbench window opens.

2. From the User Data Workbench, open Project User Data.
3. In the Project User Data window, select the **Roll-Up** tab.

The **Roll-Up** tab opens.

4. In the **Roll-Up** tab, select the user data roll-up and click **Remove**.

The user data roll-up method is removed.

5. In the **Roll-Up** tab, click **Save**.

The changes to the user data type are saved.

Chapter

11

Rolling Out Your Deployment Process

In This Chapter:

- *General Deployment System Configuration Checklist*
 - *Workflow Checklist*
 - *Object Type Checklist*
 - *Environments Checklist*
 - *Security and User Access Checklist*
 - *Dashboard and Portlet Checklist*
 - *Cross-Entity Checklist*
 - *Migrating Configuration Data into Production*
 - *Enabling Entities and User Access*
 - *Educating Your Users*
-

General Deployment System Configuration Checklist

The following items have to be configured to enable the deployment system. For additional details/instructions on configuring each of the entities, see the referenced sections in the Notes column.

Table 11-1. General configuration checklist

Done	Entity Defined?	Notes
	Workflow	<p>One or more workflows that will be used to process the packages (deploy your objects) must be available. See the following for details on workflow construction:</p> <ul style="list-style-type: none"> • Configuring Workflows on page 57 • Configuring Workflow Components on page 127
	Object Types	<p>Define an object type for each type of object to be deployed. This includes creating fields that describe the object and commands required to process it during deployment. See the following for details on object type and command construction:</p> <ul style="list-style-type: none"> • Configuring Object Types on page 167 • Commands and Tokens Guide and Reference
	Environments	<p>Define the source and destination environments for the objects being deployed. See the following for details on environment definition:</p> <ul style="list-style-type: none"> • Configuring Environments on page 229
	Security Groups/User Access	<p>Define the security groups used to control different aspects of the deployment process: package creation, package processing, and deployment system configuration. See the following for details on security group and user participant definition:</p> <ul style="list-style-type: none"> • Security Model Guide and Reference <p>Work with the application administrator to configure user and security group definitions.</p>
	Dashboard/Portlets	<p>Decide which portlets can be added to the Dashboard. If none of the default system portlets suit the business needs, construct custom portlets. See the following for details on portlet construction and Default Dashboard creation:</p> <ul style="list-style-type: none"> • Configuring the Standard Interface

Workflow Checklist

Table 11-2. Workflow configuration checklist

Done	Workflow Check Item	Notes
	Business process is modeled on the Workflow	Execution, decision and condition steps have been added to the Layout tab on the workflow window. See the following for details: <ul style="list-style-type: none"> • Configuring Workflows on page 57
	Command execution points are set	The points at which commands will run have been determined. If they are object-specific commands, or commands that need to run for each package line, then you should execute the object type commands. If they are other, non-object-specific commands, consider setting them in the workflow step source.
	Decision steps set	See the following for details: <ul style="list-style-type: none"> • Configuring Workflows on page 57
	Timeouts are set	Timeout values have been placed on how long workflow steps can remain in a single state, and have added timeouts to command executions. This ensures that the deployment process is not delayed from lack of user action or complications during executions. See the following for details: <ul style="list-style-type: none"> • Configuring Workflows on page 57
	Automatic transitions are properly set	Ensure that the package will not become “stuck” in a step. This can happen when the results of an execution or query yield a result that is not linked to a transition out of the step. See the following for details: <ul style="list-style-type: none"> • Configuring Workflows on page 57
	Manual transitions are set	Ensure that the step has a transition path for each available decision result. See the following for details: <ul style="list-style-type: none"> • Configuring Workflows on page 57
	Deployment steps specify a source and destination Environment	Execution steps (and the included commands) need to recognize which environments to connect to for machine connections and object transfers. See the following for details: <ul style="list-style-type: none"> • Configuring Workflows on page 57
	Notifications are set on appropriate Workflow Steps	You need to configure notifications to be sent at specific points in the process. See the following for details: <ul style="list-style-type: none"> • Configuring Workflows on page 57

Table 11-2. Workflow configuration checklist [continued]

Done	Workflow Check Item	Notes
	Includes a Close step	The process should conclude with a “Closed” package at all exit points. See the following for details: <ul style="list-style-type: none"> • Configuring Workflows on page 57
	Verify the Workflow	Use the workflow’s Verify tool to avoid any serious configuration errors. The workflow verification tool checks for the possible configuration errors described in the following table (Table 11-3). See the following for details: <ul style="list-style-type: none"> • Configuring Workflows on page 57

Table 11-3. Workflow logical guidelines

Guideline	Returns	Reason
Workflow should have at least one step.	Error	No processing can be done if the workflow has no steps.
Workflow should have at least one Close step.	Error	The package line cannot be closed without a Close step in the workflow.
Each enabled workflow step should have at least one incoming transition	Error	It is not possible to flow to a workflow step without an incoming transition.
Each decision step should have at least one security group, user or token defined in the Security tab.	Error	No one is authorized to act on the step without a security group.
Each manual execution step should have at least one security group, user or token defined in the Security tab.	Error	No one is authorized to act on the step without a security group.
First workflow step should not be a condition.	Error	Workflow processing may not be correct if the first step is a condition.
A condition step should not have a transition to itself.	Error	A condition with a transition to itself could cause the workflow to run indefinitely.
Transition value is not a valid validation value (error).	Error	The validation value has changed since the transition has been made.
Close steps should not have a transition on ‘Success’ or ‘Failure.’ Return steps should have no outgoing transitions.	Error	The package or request will not close if a transition exists on ‘Success.’
An immediate execution step should not have a transition to itself on ‘Success’ or ‘Failure.’	Error	The workflow could loop indefinitely.

Table 11-3. Workflow logical guidelines [continued]

Guideline	Returns	Reason
'Other Values' and 'All Values' transitions should not exist at the same step.	Warning	'Other Values' transition is always ignored if an 'All Values' transition exists.
Each workflow step should have at least one outbound transition.	Warning	The branch of the workflow stops indefinitely without closing the package line or request.
Each value from a list-validated validation should have an outbound transition.	Warning	There are validation values that do not have transitions defined.
Step with text or numeric validation should have an 'Other Values' or 'All Values' transition.	Warning	Since text and numeric Validations are not limited, an 'Other Values' or 'All Values' transition should be defined.
All steps should be enabled.	Warning	Disabled steps cannot be used by a package line or request.
AND or OR condition step should have at least two incoming transitions.	Warning	An AND or OR condition with only one incoming transition will always immediately be true and have no effect.
Subworkflow should have at least one Return step.	Error	Should include a Return step.
Notifications with reminders should not be set on results that have transitions.	Error	Transition into the Return Step does not match the validation.
Close step in subworkflow will close entire package line or request.	Warning	Has a Close step.
Top-level workflow should not have a Return step.	Error	Only subworkflows have a Return step.

Object Type Checklist

Table 11-4. Object type configuration checklist

Done	Object Type Check Item	Notes
	Fields defined	<p>Fields are required to define the object. Ensure that the correct parameters describe the object to be deployed. See the following for details:</p> <ul style="list-style-type: none"> • Configuring Object Types on page 167 • <i>Commands, Tokens, and Validations Guide and Reference</i>
	Commands defined	<p>All commands needed to process and deploy the object have been constructed. See the following for details:</p> <ul style="list-style-type: none"> • Configuring Object Types on page 167 • <i>Commands, Tokens, and Validations Guide and Reference</i>
	Conditions set in commands	<p>Conditions to steps within the command that dictate when the specific command steps run have been added. See the following for details:</p> <ul style="list-style-type: none"> • <i>Commands, Tokens, and Validations Guide and Reference</i>

Environments Checklist

Table 11-5. Environment definition checklist

Done	Environment Check Item	Notes
	Define the “source” Environment	See the following for details: <ul style="list-style-type: none"> • Configuring Environments on page 229
	Define the “destination” Environment	See the following for details: <ul style="list-style-type: none"> • Configuring Environments on page 229
	Select the appropriate connection protocol	See the following for details: <ul style="list-style-type: none"> • Configuring Environments on page 229
	Select the appropriate transfer protocols	See the following for details: <ul style="list-style-type: none"> • Configuring Environments on page 229
	Define Environment Groups	See the following for details: <ul style="list-style-type: none"> • Configuring Environment Groups on page 259
	Verify the Environment Definitions	See the following for details: <ul style="list-style-type: none"> • Configuring Environments on page 229

Security and User Access Checklist

Table 11-6. Security/user access configuration checklist

Done	Security/User Access Check Item	Notes
	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. See the following for details: <ul style="list-style-type: none"> • <i>Security Model Guide and Reference</i>
	Created security groups (for association with Workflow Steps)	security groups to allow users to act on a specific workflow step have been created. See the following for details: <ul style="list-style-type: none"> • <i>Security Model Guide and Reference</i>
	Set security on Package Creation	All available options for restricting who can create and submit packages have been set. See the following for details: <ul style="list-style-type: none"> • <i>Security Model Guide and Reference</i>
	Set security on Package processing	All available options for restricting who can process packages have been set. See the following for details: <ul style="list-style-type: none"> • <i>Security Model Guide and Reference</i>
	Set security on deployment system configuration	You have specified who can modify the deployment process. This includes editing the workflow, object type, environment, security groups, and so on. See the following for details: <ul style="list-style-type: none"> • <i>Security Model Guide and Reference</i>

Dashboard and Portlet Checklist

Table 11-7. Dashboard/portlet configuration checklist

Done	Dashboard Check Item	Notes
	Created custom Portlets to display desired data	Advanced users with a knowledge of SQL programming can create their own Dashboard. See the following for details: <ul style="list-style-type: none"> • <i>Configuring the Standard Interface</i>
	Enable Portlets for use on the Dashboard	See the following for details: <ul style="list-style-type: none"> • <i>Configuring the Standard Interface</i> • <i>Security Model Guide and Reference</i>
	Specify which users can add use certain Portlets	See the following for details: <ul style="list-style-type: none"> • <i>Configuring the Standard Interface</i> • <i>Security Model Guide and Reference</i>
	Create a default Dashboard or distribute a Dashboard to your users.	See the following for details: <ul style="list-style-type: none"> • <i>Configuring the Standard Interface</i>

Cross-Entity Checklist

Table 11-8. Cross entity configuration checklist

Done	Entities	Configuration Considerations
	Workflow and Object Type	<p>The following items should be coordinated between the workflow and object type:</p> <ul style="list-style-type: none"> • Decide which workflow steps will execute the object type commands. • Decide which object type commands will 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 - Change Management Settings tab).
	Workflow and Environments	<p>The following items should be coordinated between the workflow and environments:</p> <ul style="list-style-type: none"> • Specify the source and destination environments (or environment groups) on the appropriate workflow execution steps.
	Workflow and Security Groups	<p>The following items should be coordinated between the workflow and security groups:</p> <ul style="list-style-type: none"> • Associate security groups with workflow steps. Users in the included groups can act on the step. • Set workflow and workflow step ownership.
	Object Types and Environments	<p>The following items should be coordinated between the object types and environments:</p> <ul style="list-style-type: none"> • Specify any environment overrides in the object type commands.
	Security Groups and other entities (Object Types, Environments, and so on)	<p>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.</p>

Migrating Configuration Data into Production

After all entities have been validated in a Development or Testing Environment, perform the final migration into production. The following entities can be transferred using the migrators:

- Workflows
- Object Types
- Validations
- User Data Context
- Special Commands
- Report Types
- Request Types
- Request Header Types
- Project Template
- Portlet Migrator



Note

When migrating the above entities, related configuration information is also migrated. For example, when migrating the object type the following information is also migrated: validations referenced by the object type fields, environments referenced by the validations, and special commands referenced by commands or validations.

It is possible to process all of the migrations in a single package. Each individual entity (such as Workflow A, Workflow B, Object Type 1, or Object Type 2) is added as a separate package line.

The entity will inherit the **Enabled** or **Disabled** status. For example, if the object type is disabled in the test instance, then it will be disabled in the production instance upon migration. Ensure that each entity has the desired enabled or disabled status in the production instance.

Enabling Entities and User Access

Each entity used in the deployment process includes an Enabled parameter. This parameter needs to be set to **Yes** in order to provide general access. Ensure that the following entities are enabled in the system:

- Workflows (including subworkflows)
- Object Types
- Environments
- Environment groups
- Security groups
- Users
- Portlets

Educating Your Users

The final step in rolling out a deployment system is training your users. This includes educating the users on the following activities:

- **Basic Product Use.** Ensure that each user understands how to create, process, and report on packages.
- **Process-specific training.** Ensure that each user understands the deployment process. Consider holding a formal rollout meeting or publishing documents on the configurations and processes.
- **User Responsibilities.** Ensure that each user understands their individual role in the deployment process. For example, the QA team may be restricted to only approve the testing phase of the deployment. Also, take advantage of the product's email notification functionality. The notifications can be very specific, instructing individual users with their required deployment tasks.

Appendix **A** Worksheets

In This Appendix:

- *Configuration Workflow Worksheets*
 - *Execution Workflow Step Worksheets*
 - *Decision Workflow Step Worksheets*
 - *Subworkflow Workflow Step Worksheets*
 - *Object Type Configuration Sheets*
-

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				
16				
17				
18				
19				
20				
* Type = Workflow Step Type: Decision (D), Execution (E), Condition (C), Subworkflow (S)				

Execution Workflow Step Worksheets

Table A-2. Workflow step [execution], step number _____

	Value
Step Name	
Goal/Result of Step	
Validation*	
Execution Type**	
Processing Type	
Timeout (Days)	
Source Environment (Group)	
Dest Environment (Group)	
Security (who can act on step): <ul style="list-style-type: none"> • User Name • Standard Token • User Defined Token 	
Include Notification (Yes/No)	
Notification Event	
Notification Recipient: <ul style="list-style-type: none"> • 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, 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: <ul style="list-style-type: none"> • Execute Commands • Close • Jump/Receive • Ready for Release • Return from Subworkflow 	
PL/SQL Function	
Token	
SQL Statement	
Workflow step commands	

Decision Workflow Step Worksheets

Table A-5. Workflow step [decision], step number _____

	Value
Step Name	
Goal/Result of Step	
Validation*	
Decisions Required (Vote on Step's outcome?)	<ul style="list-style-type: none"> • One • At Least One • All
Timeout (Days)	
Security (who can act on step):	
<ul style="list-style-type: none"> • Security Group • User Name • Standard Token • User Defined Token 	
Include Notification (Yes/No)	
Notification Event	
Notification Recipient:	
<ul style="list-style-type: none"> • 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: <i>(text field, auto-complete, drop-down list, and so on)</i>	
Validation Definition (list of values or SQL)	

Subworkflow Workflow Step Worksheets

Table A-7. Workflow step [subworkflow], step number ____

	Value
Step Name	
Goal/Result of Step	
Validation*	
Vote on Step's outcome?	
Timeout (Days)	
Source Environment (Group)	
Dest Environment (Group)	
Security (who can act on step): <ul style="list-style-type: none"> • Security Group • User Name • Standard Token • User Defined Token 	
Include Notification (Yes/No)	
Notification Event	
Notification Recipient: <ul style="list-style-type: none"> • 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?	
Validation Type: <i>(text field, auto-complete, drop-down list, and so on)</i>	
Validation Definition (list of values or SQL)	

Object Type Configuration Sheets

Table A-9. Object type information

	Value
Object Type Name	
Description	

Table A-10. Object type fields

#	Field Names	Description
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		

Table A-11. Object type commands

Goal of Commands	
Command Steps	
Conditions (When to execute)	

Table A-12. Object type field information

Field Name	
Validation*	
Field Behavior:	
Attributes (select one):	<ul style="list-style-type: none"> • Display • Editable • Display Only • Required
Default Value	
Dependencies:	
Clear field when	
Display only when	
Required when	

Table A-13. Field validation information

Existing Validation?	
New Validation?	
Validation Type: (<i>text field, auto-complete, drop-down list, and so on.</i>)	
Validation Definition (list of values or SQL)	

Table A-14. Object type field information

Field Name	
Validation*	
Field Behavior:	
Attributes (select one):	<ul style="list-style-type: none"> • Display • Editable • Display Only • Required
Default Value	
Dependencies:	
Clear field when	
Display only when	
Required when	

Table A-15. Field validation information

Existing Validation?	
New Validation?	
Validation Type: (<i>text field, auto-complete, drop-down list, and so on.</i>)	
Validation Definition (list of values or SQL)	

A

- access grants 25
- adding
 - notification intervals to notification templates 282
 - ownerships to environment groups 266
 - ownerships to environments 248
 - packages to releases using Package window 216
 - packages to releases using Release window 214
 - participants to environments 251
 - requests to releases 219
 - requests to releases with Release window 219
 - requests to releases with Requests window 221
 - transitions back to the same step 105
- AND condition workflow steps 68
- application codes
 - copying from other environments 246
 - environments 243

B

- base paths 239, 240
 - mass updates 255
- business flows defined 30

C

- closing
 - packages as failed 146
 - packages as success 145
 - workflow steps 71
- commands
 - changing field values 198
 - conditions 194
 - conditions examples 194
 - requirements 46
- communication requirements 55
 - example 56
- condition workflow steps 68
- configuration-level restrictions 25
- configuring
 - dynamic security for workflow steps 82
 - environment groups information 263
 - environments information 237
 - environments overview 234
 - field dependencies 178
 - field widths in object types 184
 - first workflow step 72
 - follow up notifications 92
 - intervals for notifications 91
 - marking packages ready for release 147
 - moving object type fields 185
 - notification intervals on notification

- templates 282
 - notification message for workflow steps 95
 - notification setup for workflow steps 86
 - notification templates 274
 - notification templates creating
 - notification templates 274
 - notifications for workflow steps 84
 - object type field dependencies 178
 - object type fields 182
 - object type names 187
 - ownership for environments 248
 - ownership of notification templates 279
 - ownership of workflow step sources 133
 - recipients for notifications 93
 - releases 206
 - releases overview 209
 - security for workflow steps 79
 - sending notifications at specific times 91
 - sending notifications on specific errors 89
 - sending notifications on specific results 87
 - sending notifications when workflow step eligible 86
 - swapping object type fields 186
 - timeouts for workflow steps 98
 - transitions back to step 105
 - transitions based on all but one specific value 103
 - transitions based on all results 104
 - transitions based on data 103
 - transitions based on errors 104
 - transitions based on specific results 101
 - transitions based on workflow results 108
 - transitions for subworkflows 110
 - transitions for workflow steps 100
 - transitions not based on specific results 103
 - user access 332
 - user access for environment groups 269
 - user data column widths 306
 - user data field dependencies 302
 - user data field widths 306
 - user data fields 295, 300, 307
 - user data general information 292
 - user data layouts 306
 - user data overview 290
 - user data roll-up fields 310
 - validations and execution types 113
 - validations for workflow steps 111
 - workflow general information 64
 - workflow step sequences 72
 - workflow step source restrictions 129
 - workflow steps 75, 77
 - workflows and performance considerations 162
- connection protocols for environments 230
 - copying
 - application codes from other environments 246
 - object type fields 180
 - user data fields 299
 - workflows for trial versions 163
 - creating
 - decision workflow step sources 135
 - distributions 223
 - environment groups 264
 - environments 238
 - execution workflow steps 139
 - notification templates 274
 - releases 207
 - subworkflow workflow step sources 153
 - user data fields 295
 - workflow parameters 155
 - workflow step sources 131
 - workflow step sources overview 128
 - workflows 64
 - cross-entity checklist 330
- ## D
- decision workflow step sources 135
 - decision workflow steps 68
 - worksheets 337
 - deleting
 - notification templates 273
 - object type fields 183
 - ownerships from environment groups 268
 - ownerships from environments 250

-
- ownerships from notification templates 281
 - participants from environment groups 270
 - participants from environments 252
 - user data fields 305
- dependencies and run groups 204
- deployments
- business flow 30
 - checklists 322
 - command requirements 46
 - communication requirements 55
 - environment maintenance 253
 - environment requirements 47
 - migrating finished 331
 - object requirements 42
 - object revision 188
 - participants and security 50
 - process requirements 28
 - release management 41
 - step information requirements 36
 - subworkflows 39
 - technical flow requirements 33
- distributions
- creating 223
 - disabling package lines 225
 - enabling package lines 225
 - processing 226
 - running through a workflow 226
 - workflow 201
- dynamic security for workflow steps 82
- E**
- editing
- user data fields 300
- enabling workflows 73
- entity-level restrictions 25
- environment groups 20
- adding ownership 266
 - configuring 263
 - creating 264
 - deleting ownerships 268
 - deleting participants 270
 - overview 260
 - setting ownership 266
 - setting the execution order 265
 - setting user access 269
- Environment Groups Workbench 262
- environments 20
- adding ownerships 248
 - adding participants 251
 - checklist 327
 - choosing based on application code 116
 - configuring general information 237
 - configuring overview 234
 - connection protocols 230
 - copying application codes 246
 - creating 238
 - deleting environments 250
 - deleting participants 252
 - integrating with workflows 115
 - maintenance 253
 - mass update of base paths 255
 - opening the Workbench 236
 - overview 230
 - password management 256
 - requirements 47
 - selecting FTP protocols 231
 - setting ownership 248
 - testing setup 253
 - transfer protocols 231
 - transfer protocols notes 231
 - using application code environments 243
- executing
- multiple system level commands for packages 152
 - transitions for packages based PL/SQL function results 147
 - transitions for packages based SQL function results 148
 - transitions for packages based token results 150
- execution order 263
- in environment groups 265
- execution step source
- creating 139
 - defining executions 143
-

- execute object type commands 144
- execution workflow steps 70
 - set up rules 143
 - worksheets 335
- executions
 - configuring workflow steps with validations 113
 - defining 143
 - execute object type commands 144
 - types in workflows 141

F

- field-level restrictions 25
- fields
 - changing widths in object types 184
 - configured in object types 182
 - configuring
 - dependencies for object types 178
 - configuring user data dependencies 302
 - configuring user data fields 295, 300
 - copying in object types 180
 - copying user data 299
 - copying user data fields 299
 - creating for user data 295
 - deleting from user data 305
 - deleting in object types 183
 - deleting user data fields 305
 - moving in object types 185
 - moving user data fields 307
 - selecting validations for object types 174
 - swapping in object types 186
 - user data 300
 - user data dependencies 302
- FIRST LINE condition workflow steps 69
- FTP protocols for environments 231

I

- integrating
 - environments and workflows 115
 - object type commands and workflows 114
 - object types and workflows 114
 - requests and packages 117

J

- jump step generation 121
- jump/receive
 - step labels 118
 - workflow steps 124

L

- LAST LINE condition workflow steps 69
- licenses 25
- loop counter example 157

M

- mapping workflows to processes 60
- mass updates or base paths 255
- migrating user data 290
- modifying
 - active workflows 161
 - production workflows 164

N

- notification templates
 - adding notification intervals 282
 - checking usage 285
 - configuring ownership 279
 - creating 274
 - deleting 273
 - deleting notifications 281
 - opening Workbench 273
 - overview 272
- notifications
 - configuring 84
 - configuring message 95
 - configuring messages 86
 - sending at specific times 91
 - sending follow ups 92
 - sending on specific errors 89
 - sending on specific results 87
 - sending to recipients 93
 - sending with step eligible 86
 - smart URL tokens 98

specifying intervals 91
using smart URLs 97
using tokens 97

O

object requirements 42
object revision 188
object types 21, 178
 changing field widths 184
 checklist 326
 command requirements 46
 configuring fields 182
 copying fields 180
 deleting fields 183
 fields changing values with commands 198
 fields selecting validation 174
 fields text area 185
 integrating commands with workflows 114
 integrating with workflows 114
 moving fields 185
 naming 187
 previewing 187
 setting revisions 188
 swapping fields 186
 worksheet 341
opening
 environment groups Workbench 262
 Environments Workbench 236
 Notification Templates Workbench 273
 releases 206
 User Data Workbench 291
 Workflow Workbench 63, 130
OR condition workflow steps 68

P

package level subworkflow 202
package lines 21
packages 21
 adding to releases using Package window 216
 adding to releases using Release window

214
 closing as failed 146
 closing as success 145
 disabling package lines in distribution 225
 enabling package lines in distribution 225
 integrating with requests 117
 marking ready for release 147
 moving out of workflow steps 165
 package level subworkflows 202
 processing package lines 227
 ready to release workflow steps 217
 workflow 200

parameters in workflows 155
participants requirements 50
password maintenance for environments 256
previewing object types 187
process requirements 28
 business flow 30
 release management 41
 step information 36
 subworkflow considerations 39
 technical flow 33
 workflow considerations 28
processing
 distribution steps 226
 package lines 227
 releases 206, 208

R

receive steps 123
release management 41
releases
 adding packages using Package window 216
 adding packages using Release window 214
 adding requests 219
 adding requests using Release window 219
 adding requests using Requests window 221
 configuring overview 209
 creating 207

- dependencies 204
 - distribution workflows 201
 - open releases 206
 - overview 209
 - package workflows 200
 - pre-configuration 206
 - processing 208
 - processing overview 206
 - submitting releases 206
 - verifying 222
- requests
- adding to releases 219
 - adding to releases using Release window 219
 - adding to releases using Requests window 221
 - integrating with packages 117
 - moving out of workflow steps 165
- requirements for deployment processes 28
- rollout migration deployment system 331
- run groups 204
- S**
- security
- access grants 25
 - checklists 328
 - configuration-level restrictions 25
 - configuring workflow steps 79
 - entity-level restrictions 25
 - field-level restrictions 25
 - licenses 25
 - requirements 50
- selecting
- FTP protocol for environments 231
 - validations for object types 174
- sending
- notification follow ups 92
 - notification recipients 93
 - notifications at specific times 91
 - notifications on specific errors 89
 - notifications on specific results 87
 - notifications when workflow steps become
 - eligible 86
 - setting execution workflow steps rules 143
- smart URL tokens 98
- step sources
- execution 139
 - overview 131
- submitting releases 206
- subworkflows 202
- configuring to and from workflow steps 110
 - consideration example 39
 - considerations 39
 - returning to Change Management workflows 153
 - workflow steps 71
 - worksheets 339
- SYNC condition workflow steps 69
- T**
- technical flow 33
- example 33
- text areas in object types 185
- timeouts in workflow steps 98
- tokens
- using in notifications 97
- transfer protocols
- configuration notes for environments 231
 - for environments 231
- transitions
- back to same step 105
 - based on workflow results 108
 - configuring for specific results 101
 - configuring for workflow steps 100
 - configuring for workflow steps based on all but one specific value 103
 - configuring for workflow steps based on all results 104
 - configuring for workflow steps based on data 103
 - configuring for workflow steps based on errors 104

configuring not based on specific results **103**
 executing multiple system level commands for packages **152**
 package transitions based on PL/SQL function results **147**
 package transitions based on SQL function results **148**
 package transitions based token results **150**
 to and from subworkflows **110**

U

user data

changing column widths **306**
 configuring field dependencies **302**
 configuring fields **300**
 configuring general information **292**
 configuring layouts **306**
 copying fields **299**
 creating fields **295**
 deleting fields **305**
 editing fields **300**
 field dependencies **302**
 migrating **290**
 moving fields **307**
 opening Workbench **291**
 overview **288, 290**
 previewing the layout **309**
 referring to **289**
 removing fields **305**
 roll-up fields **310**
 swapping field positions **308**

using

application codes environments **243**
 commands to change object type fields **198**
 smart URLs in notifications **97**
 tokens in notifications **97**
 workflow step source restrictions **129**

V

validations

configuring for workflow steps **111**

configuring workflow steps with execution types **113**

validations in jump/receive workflow steps **119**

verifying

releases **222**

verifying workflows **73**

visibility **55**

W

workflow considerations **28**

immediate or manual **29**

timeouts **29**

workflow steps

AND condition **68**

choosing **66**

closing **71**

condition **68**

configuring **75**

configuring first step **72**

configuring general information **77**

configuring notification messages **95**

configuring notification setup **86**

configuring notifications **84**

configuring security **79, 82**

configuring sequences **72**

configuring step source ownership **133**

configuring subworkflows **153**

configuring timeouts **98**

configuring to and from subworkflows **110**

configuring transitions **100**

configuring transitions based on all but one specific value **103**

configuring transitions based on all results **104**

configuring transitions based on data **103**

configuring transitions based on errors **104**

configuring transitions based on results **108**

configuring transitions based on specific results **101**

configuring transitions not based on specific results **103**

configuring validations **111**

- configuring validations and execution types 113
- creating decision sources 135
- decision 68
- disabling 164
- execution 70
- execution set up rules 143
- FIRST LINE condition 69
- LAST LINE condition 69
- moving packages out of steps 165
- moving requests out of steps 165
- OR condition 68
- ready to release 217
- restrictions 129
- sources overview 128
- subworkflow 71
- SYNC condition 69
- using smart URLs in notifications 97
- using tokens in notifications 97
- workflows 22
 - adjusting step sequence 72
 - AND condition workflow steps 68
 - checklist 323
 - choosing environments based on application code 116
 - choosing steps 66
 - closing 71
 - condition workflow steps 68
 - configuring notification messages for workflow steps 95
 - configuring notification setup for workflow steps 86
 - configuring notifications for workflow steps 84
 - configuring security for workflow steps 79
 - configuring workflow steps 75, 77
 - creating 64
 - creating parameters 155
 - creating step source 131
 - creating subworkflows 153
 - creating transitions based on PL/SQL results 147
 - creating transitions based on SQL results 148
 - creating transitions based on token results 150
 - decision workflow steps 68
 - disabling workflow steps 164
 - distribution 201
 - dragging and dropping 65
 - enabling 73
 - events 141
 - executing multiple system level commands for packages 152
 - execution step source 139
 - execution types 141
 - execution workflow steps 70
 - FIRST LINE condition workflow steps 69
 - integrating jump step source 121
 - integrating receive step source 123
 - integrating with environments 115
 - integrating with object type commands 114
 - integrating with object types 114
 - jump/receive step 118
 - jump/receive validations 119
 - jump/receive workflow steps 124
 - LAST LINE condition workflow steps 69
 - logical rules 324
 - loop counter example 157
 - mapping to process 60
 - modifying in production 164
 - modifying while in use 161
 - open the Workbench 130
 - opening Workbench 63
 - OR condition workflow steps 68
 - overview 58
 - package 200
 - packages ready to release 217
 - performance considerations 162
 - redirecting workflows 164
 - running distributions 226
 - scope 200
 - specifying first step 72
 - subworkflow workflow steps 71
 - subworkflows and Change Management 153

SYNC condition workflow steps **69**

trail versions **163**

using parameters **155**

verifying **73**

worksheets **334**

