Mercury™ IT Governance Center

Configuring a Deployment System (Change Management)

Version 5.5.0

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: U.S. Patent Nos. 5,701,139; 5,657,438; 5,511,185; 5,870,559; 5,958,008; 5,974,572; 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 and 6,694,288. Other patents pending. All rights reserved.

Mercury, Mercury Interactive, the Mercury Interactive logo, LoadRunner, LoadRunner TestCenter, QuickTest Professional, SiteScope, SiteSeer, TestDirector, Topaz and WinRunner are trademarks or registered trademarks of Mercury Interactive Corporation or its subsidiaries, in the United States and/or other countries. The absence of a trademark from this list does not constitute a waiver of Mercury Interactive's intellectual property rights concerning that trademark.

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

Mercury Interactive Corporation
1325 Borregas Avenue
Sunnyvale, CA 94089 USA
Tel: (408) 822-5200
Fax: (408) 822-5300

© 2004 Mercury Interactive Corporation. All rights reserved.

If you have any comments or suggestions regarding this document, please send them via e-mail to documentation@merc-int.com.

Publication Number: ConfigureDeploy-0304A

Table of Contents

Chapter 1 Introduction

Mercury Change Management allows companies to automate and manage the deployment of packaged applications, custom applications, legacy systems, and Web content. Change Management enforces deployment processes and performs all tasks required to install software changes correctly across Development, Test, Stage, and Production systems.

About This Document

This guide provides instructions for configuring a deployment system. This includes requirements gathering, modeling your processes in a Workflow, defining commands used by the execution engine, and rolling out this system to your users. Each chapter covers a particular topic:

Key Concepts	Defines the key concepts and definitions used when creating a deployment system.
Developing Your Configurations (Using Migrators)	Provides an overview of how to use multiple product instances (installations) to configure and deploy a system.
Configuring a Deployment System—Process Overview	Provides an overview of the process used to configure a deployment system and summarizes each phase of configuration.
Gathering Process Requirements and Specifications	Discusses the information that needs to be collected before configuring a deployment system.
Mapping your Process into a Workflow	Provides instructions for setting up a Workflow skeleton, including all Workflow steps, transitions and Validations included in the process.

Constructing the Object Type	Provides instructions on configuring Object Types that will be used to process objects (Package Lines) through the deployment Workflow. This includes configuring Object Type fields and Commands.
Defining your Environments	Provides an overview for defining the Environments that will be used in the deployment system.
Integrating Participants into Your Deployment System	Provides instructions for integrating users into the deployment process.
Setting Up Communication Paths	Provides an overview for different modes of communication that can be used in the deployment system. This includes configuring Email Notifications, the Dashboard, and reports.
Rolling Out Your Deployment Process	Discusses a number of topics to consider when rolling out a deployment process.
Validations	Provides instructions for creating and using Validations in a deployment system.
Tokens	Summarizes how Tokens are used in Change Management and provides a link to additional details.
User Data Creation and Processing	Provides instructions on creating and using User Data fields in a deployment process.
Configuration Worksheets	Provides worksheets that can be printed out and used to capture data required for configuring a deployment system.

Intended Audience

The intended audience for this document include:

- Business or technical users who configure and maintain a deployment system using Mercury Change Management
- Users responsible for Workflow configuration
- Managers responsible for reporting on software and application deployments
- Release Managers

Document Conventions

Table 1-1 lists the types of conventions used in this document.

Table 1-1. Document conventions

Convention	Description	Example
Button, menu, tabs	Names of interface components that can be clicked (such as buttons, menus, and tabs) are shown in bold.	**Apply** button
Fields, Windows, Pages	Names of windows, fields, and pages are shown as displayed.	New Request window
`Code`	Code input and output are shown as displayed.	`CauchoConfigFile C:/ITG_Home/conf/resin.conf`
Link	Linked URLs, filenames, and cross references are shown as blue italicized text.	*www.mercury.com*
Variable	Variables are shown as italicized text.	*ITG_Home*/bin directory
Note	Used to identify note boxes that contain additional information.	
Caution	Used to identify caution boxes that contain important information. Follow the instructions in all caution boxes, failure to do so may result in loss of data.	
Example	Used to identify example boxes that contain examples of related procedure.	

Additional Resources

Mercury Interactive provides the following additional resources to help you successfully configure the Mercury products:

- *Related Documentation*

- *Customer Support*
- *Education Services*

Related Documentation

The Library includes additional documents related to the topics discussed in this guide. Access the Library through the Mercury ITG Center online help.

Using the Dashboard	This document provides details for defining and configuring the Dashboard and custom Portlets.
Processing Requests (Demand Management)	This document explains how to process Requests using Demand Management.
Processing Packages (Change Management)	This document explains how to process Packages using Change Management.
Using the Workbench	This document explains how to navigate through the Workbench interface.
System Administration Guide	This document provides information necessary to implement, configure, and maintain Mercury ITG Servers.
Migrators Guide and Reference	This document provides details for configuring and using the Mercury ITG Migrator Object Types to migrate or archive data from Mercury ITG Center instances.
Commands and Tokens Guide and Reference	This document provides information on using commands and Tokens.
Security Model Guide and Reference	This document presents an overview of the data security model and provides instructions for controlling access to different entities.
Configuring a Request Resolution System	This document provides instructions for configuring a Request resolution system. This includes requirements gathering, modeling your processes in a Workflow, defining a Request Type to be integrated with the Workflow, and rolling out this system to your users.

Configuring a Release Management System	This document provides details for configuring, defining and processing Releases.
Configuring the Dashboard	This document provides instructions for configuring custom Portlets, maintaining standard and custom Portlets, and setting a Default Dashboard for all users.
Reports Guide and Reference	This document provides details for running reports.
Open Interface Guide and Reference	This document provides details for integrating third-party products with Mercury ITG Center entities.
Customizing the Standard Interface	This document provides details for customizing the standard interface, including the directory structure and methods of customization for changing the presentation of the standard interface.

Customer Support

Customer support and downloads for the Mercury ITG Center and additional product information can be accessed from the Mercury Interactive Support Web site at *http://support.mercuryinteractive.com*.

Education Services

Mercury Interactive provides a complete training curriculum to help you achieve optimal results using the Mercury IT Governance Center. For more information, visit the Education Services Web site at *http://www.merc-training.com/main/ITG*.

Chapter 2 Key Concepts

This chapter defines the key concepts and definitions related to configuring Mercury Change Management to deploy software and application changes.

This chapter covers the following topics:

Standard Interface and the Workbench

Mercury IT Governance (ITG) Center utilizes two interfaces: a standard HTML interface and the Workbench.

The standard interface uses HTML and Javascript to provide users with access to many key areas of functionality. This interface enables users of each product in the suite to perform common tasks without requiring a Power License.

The Workbench is a Java applet designed to help administrators, product configurers, and Power Users perform advanced configuring and processing tasks, such as creating Object Types and Workflows. The Workbench can also query detailed information on a specific entity, such as a particular Package.

When configuring a deployment system, the Workbench will be the primary interface. Also, end-user Package creation and the majority of Package processing will occur within the Workbench. For more information on standard Workbench navigation techniques, see *Using the Workbench*.

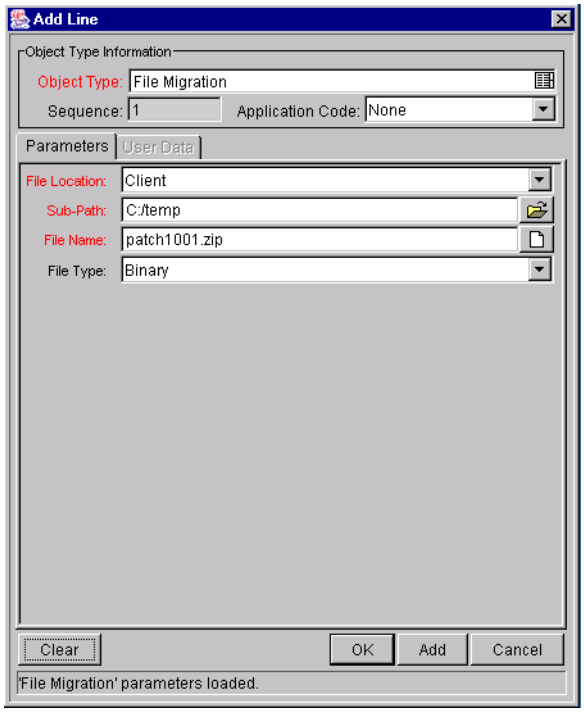
Deployment

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.

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.



Each Object Type contains:

- Object Type fields
- Object Type commands

Object Type fields describe the object—such as what it is, where it is, its name, what needs to be done to it, or if it needs to be compiled. It is possible to configure the field prompts, Tokens, behaviors and Validations uniquely for each Object Type. For example, to migrate a file, Mercury Change Management must know the file name. A field named File Name can be created to capture that information.

Object Type commands are instructions interpreted by the Mercury IT Governance Execution Engine and translated into operating system commands to be dynamically executed. Object Type commands are typically a blend between shell scripts and Mercury Change Management system Special Commands. Object Type Commands allow the automation of an entire

sequence of commands that would previously have been run manually. For example, these command sequences can automate source code compilation, check files into version control, or run a report.

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.

Definition

A Package:

- Is the fundamental work unit of Mercury Change Management.
- Represents a logical unit of objects that should be moved and tracked together.
- Contains all the information needed to process the Package, including the Package Lines, priority, and status.
- Specifies the Workflow to be used to deploy the change.
- Contains a list of all objects to be tracked and/or migrated as the Package moves through its Workflow.

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. *Figure 2-1* shows a sample Package.

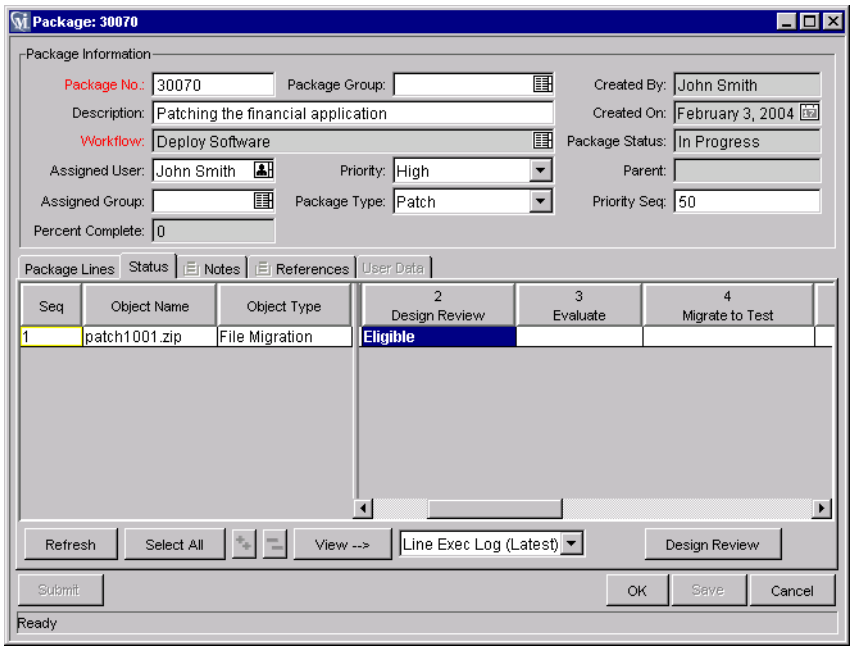


Figure 2-1 Sample Package

Package Lines

Packages can be used to deploy multiple objects. Each object is specified on a separate Package Line. *Figure 2-2* shows a Package Line in a Package.

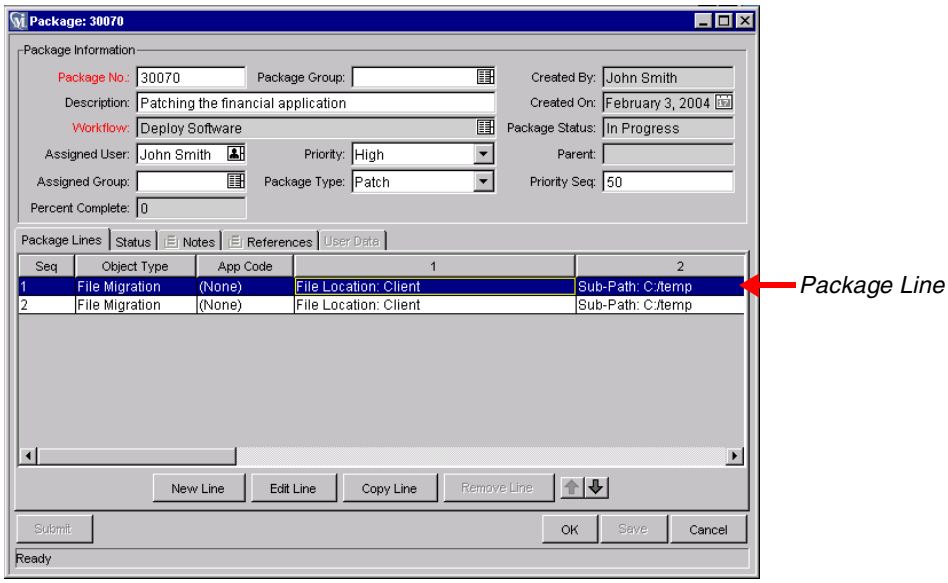


Figure 2-2 Sample Package Line

It is possible to configure your Workflow to process different types of objects (Package Lines) 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.

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.

Figure 2-3 shows a sample Workflow.

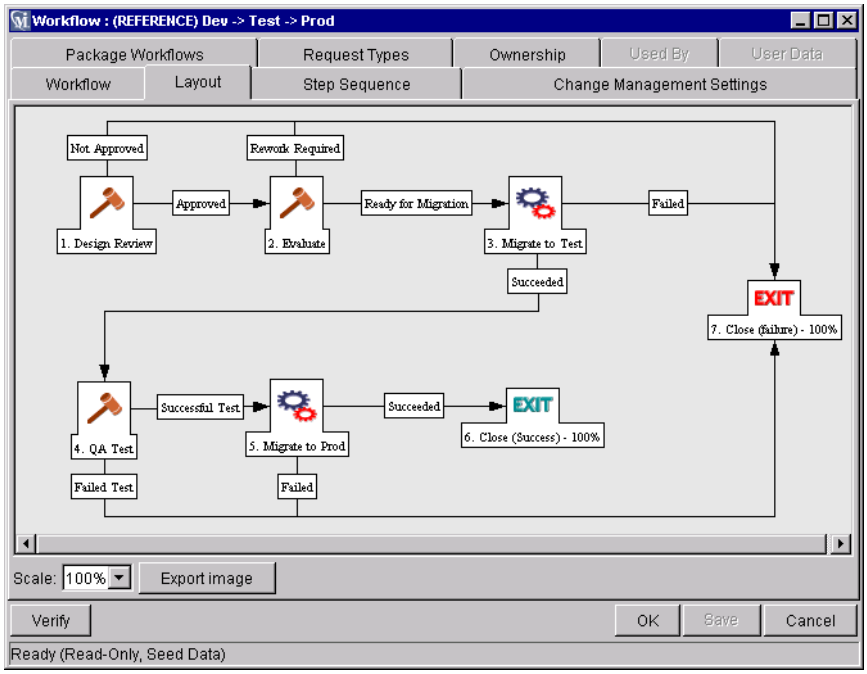


Figure 2-3 DEV -> TEST -> Prod Workflow

Object Type - Workflow Integration

Object Types are tightly integrated with the Workflow engine. It is possible to configure the Workflow to execute the commands contained in the Object Type at specific points in the process (Workflow Step). The Object Type commands are executed at Execution Workflow Steps.

Note the following configuration tips regarding the interaction between Object Type Commands and the Workflow:

- 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 (Package Line) 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.
- The Object Type can be configured to run only certain Commands at a particular Step. To do this, specify a Command Condition. For details, see *"Creating Object Type Commands"* on page 144.
- Each Object Type command can be configured so that only certain steps (within a command) are executed within a particular Workflow Step. This is done using conditional statements within the actual commands.

When a file is migrated from one location to another, it may be necessary to change directories (`cd`) on either the source or the destination machines. If the `cd` command is used to automatically change to a directory that does not exist, an error message appears and the migration will be cancelled. To avoid this, the following conditional statement can be used to ensure that the desired directory exists before the `cd` command is issued:

```
if [ ! -d [P.P_SUB_PATH] ]; then mkdir -p [P.P_SUB_PATH]; fi
cd [P.P_SUB_PATH]
```

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 (for example, named "Mercury ITG") 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.

(Any Windows NT computer that Mercury Change Management will access should have been configured as directed in *Installation Guide*).

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. Environment Groups are defined and edited using the Environment Group Workbench.

Commands

Commands are instructions interpreted by the Execution Engine and translated into operating system commands to be dynamically executed. Commands are typically a blend between shell scripts and Mercury Change Management system Special Commands. Commands allow the automation of an entire sequence of commands that would previously have been run manually. For example, these command sequences can automate source code compilation, checking files into version control, or running a report.

When configuring a deployment system, most of the Commands will be included in Object Type and Workflow Step definitions. Commands can be used with the following entities in the Mercury IT Governance Center:

- Object Types
- Workflow Steps
- Request Types
- Report Types
- Validations

For more instructions and examples on using Commands in Mercury Change Management, see *Commands and Tokens Guide and Reference*.

Special Commands

In order to simplify programming commands, Mercury Change Management provides a predefined set of Special Commands. These commands perform a variety of common functions, such as copying files between Environments and establishing connections to Environments for remote command execution. Mercury Change Management features two types of Special Commands:

- System Special Commands—These commands are shipped with Mercury Change Management. System Special Commands are read-only and have the naming convention "ksc_command_name." System Special Commands always begin with "ksc_."
- User Defined Special Commands—These commands are user-defined and have the naming convention "sc_command_name." User-defined Special Commands must begin with "sc_." User defined Special Commands can contain one or more of the System Special Commands.

Validations

Validations determine the acceptable input values for user-defined custom fields. Validations maintain data integrity by ensuring that the correct information is entered in a field before it is saved to the database. For example, Validations can be used to ensure that no textual information is entered into a numeric field, or that dates are entered into a field in the proper format. More complex Validations can be used to verify that only appropriate users are assigned to a task. The values in selection Validations (drop down lists and auto-complete lists) can be configured by either listing the values or performing a SQL query.

Validations are used in the following locations:

- Every custom field generated for an Object Type, Report Type, Request Type, or User Data.
- Every decision and execution Workflow Step.
- Every drop down list and auto-complete list in all Change Management windows are based upon a Validation. However, it is not always possible to change the Validations associated with predefined fields.

Tokens

While configuring certain features in Mercury Change Management, it is often necessary to reference information in variables that goes undefined until the product is actually used a particular context. Instead of generating objects that are valid only in those specific contexts, these variables can be used to facilitate the creation of general objects that can be applied to a variety of contexts. These variables are called **Tokens**.

There are two types of Tokens used within Mercury Change Management: standard Tokens and custom Tokens. Standard Tokens are provided with the product. Custom Tokens are generated to represent specific entity configurations such as Object Type fields, Request Type fields, and Workflow parameters. Each field of the following entities can be referenced as a Token:

- Object Types
- Request Types
- Report Types
- User Data

- Workflow Parameters

Tokens can be used in the following entity windows:

- Object Type commands
- Request Type commands
- Validation commands and SQL statements
- Report Type commands
- Executions and Notifications for a Workflow
- Workflow Step commands
- Notifications in a Report Submission
- Special Command commands
- Notifications for Tasks
- Workflow Step security

Security Groups

Security Groups are constructed to provide a set of users with specific access to product screens and functions. Each Security Group is configured with a set of Access Grants that enable specific access. Users are then associated with one or more Security Groups.

A user's Security Group memberships determine which windows a user can view or edit, which Workflows a user can use, and which Workflow Steps a user has authority to act on. Each user can be a member of multiple Security Groups. The collection of Security Groups to which a user belongs defines that user's role and access within Mercury Change Management.

Since users can be members of as many Security Groups as necessary, it is recommended that specific Security Groups are generated, each with a smaller range of responsibilities. Users can then be added to many different Security Groups to grant them their full range of access.

Security Groups control product access on the following levels:

- **Screen Security:**
  Each Security Group contains a list of Access Grants that determine a user's screen security. Access Grants are used to grant access to edit, view, manage or submit items within a specific screen. By controlling the set of Access Grants for each user, specific functional roles for the user community can be defined.
- **Workflow Step Security:**
  Each Workflow Step can be linked to a unique set of Security Groups. By adding or removing specific Security Groups from a Workflow Step in the Workflow window, it is possible to control which users can act on that Step. This security level provides an extremely detailed level of control over each user's actions.
- **Workflow Security:**
  (This security level applies to Mercury Change Management only.) Security Groups can also control which Workflows users can select to deploy their objects. When users generate a new Package, they must choose the Workflow that the requested changes will follow. The list of Workflows from which the user can choose is determined by that user's Security Group membership.
- **Application Code Security:**
  (This security level applies to Mercury Change Management only.) For complex Environments, information is often segmented in subsections called Environment Applications. Application Code security can be defined to further restrict a user's ability to cross functional boundaries and apply unwanted changes to applications that are managed by other divisions.

Only users with the Administrator licence can create, modify or delete Security Groups. This license is typically only given to a few individuals per company. This provides a centralized control over licenses and user roles within the Mercury IT Governance Center.

When configuring your deployment process, work with the Mercury IT Governance Administrator to define the Users and Security Groups needed for a process.

For details on configuring security and user access around your deployment system, see *Security Model Guide and Reference*.

Participants

Users who are involved in moving a Package through a Workflow are considered to be **Participants** in that Package. A Participant can be:

- The Assigned To user.
- A member of the Assigned Group.
- The creator of the Package.
- A member of a Security Group associated with any of the Workflow Steps contained in the Workflow.

It is possible to configure Mercury Change Management so that a Package is not visible to users who are not Participants. This means users will only see Packages relevant to their business role in their organization. Additionally, users running Reports will only see information for Packages for which they are considered to be Participants.

Integrating with Other Mercury ITG Products

This document focuses on configuring the deployment functions within Mercury Change Management. With additional Mercury IT Governance products and licenses, you can address the full set of challenges related to governing your IT department.

This section provides an overview of a few key integration points within Mercury IT Governance Center. For additional information on the complete Mercury IT Governance solution, see the Products section of the Mercury Interactive Web site at *http://www.mercuryinteractive.com/products/*.

This section covers the following topics:

Mercury IT Governance Dashboard

Intended for large and complex environments, the Mercury IT Governance Dashboard provides 360° visibility and control over technology-based initiatives and IT operational tasks. Configurable, role-based visual displays called "Portlets" provide relevant summary information and highlight exception conditions in your initiatives. Users can then drill down to any desired level of detail.

For example, a CIO may want to see the status of the major initiatives undertaken by the IT department. Instead of relying on weekly reports patched together from different sources and often compiled from out-of-date or incomplete information, he can go directly to Dashboard. The Dashboard displays the true status of the initiatives—based on current data captured automatically as part of actually performing the work. The Dashboard clearly identifies any initiative that is behind schedule, or in any other exception state, and displays the causes for the delay.

The Dashboard is beneficial to all participants throughout the Technology Chain. For example, developers can use the Dashboard to view all of their own action items, and end-users can consult their own Dashboards to see the status of all the Requests they have submitted.

Mercury Project Management

Mercury Project Management adds a critical dimension, automated execution, to complex project management in large IT organizations. Unlike static project management tools that simply schedule the tasks, dates, and resources, Mercury Project Management's automation proactively pushes project tasks to the assigned resource, links with Mercury Demand Management and Mercury Change Management to automatically perform issue resolution and deployment tasks, and automatically updates and reports project status as task are completed. Project managers guide projects from concept to completion from Mercury Project Management centralized environment.

Packages (used in your deployment process) can be added to the Mercury Project Management project plan. Dependencies can be set between Packages and Tasks on the project. This ensures that the technical aspects of the deployment process is respected by other resources on the project plan.

Mercury Demand Management

Mercury Demand Management accelerates the request resolution process from inception through implementation. Businesses can use Demand Management to model and enforce their best practice request management processes. As each new request is entered, email and pager notifications speed the request through the process, freeing users to perform required triage, approval and other resolution tasks. Higher priority requests receive appropriate treatment automatically. Resolution performance monitoring by activity, request, or even across the organization ensures SLAs are met.

Requests (in Mercury Demand Management) can be functionally linked to Packages (in Mercury Change Management). This enables you to utilize the request resolution features of Demand Management within your deployment system.

Mercury Change Management Extensions

Mercury Change Management Extensions simplify the complex activities required to maintain large enterprise applications like Oracle, PeopleSoft, SAP, and Siebel and Web applications built using Java, Oracle and others. These applications are constantly changing as new modules are added, customizations developed, configurations modified, patches applied, etc. The changes must be done precisely across the Development, Test, Stage and Production system landscape, usually by highly paid and hard-to-find specialists. Mercury Extensions automate these precise tasks using best practice processes designed specifically for each application.

Reports

Change Management reports output text that provides information on your specific entities or configurations. For a complete list of the reports used in commonly used in deployment systems, see *Reports Guide and Reference*.

Migrators

Migrators are used to move Mercury IT Governance configuration data such as Validations, Workflows, and Object Types between product instances (installations). Each Migrator is designed to migrate a specific Mercury IT

Governance entity, as well as all of its dependent objects. Migrators are delivered as system Object Types.

You can migrate a Request Type from your Testing instance to your Production instance. When migrating the Request Type, the following information related to the Request Type is also migrated: Validations referenced by the Request Type fields, and any Special Commands referenced by Request Type Commands or Validations.

Chapter 3
Developing Your Configurations (Using Migrators)

This chapter introduces the concept of using multiple instances, such as Development, Testing, and Production, when configuring the Mercury IT Governance Center. It then provides and overview on how to use Mercury IT Governance Migrators to manage the deployment of configurations between these instances. Finally, this chapter discusses how to use the Migrators to archive configuration data.

A Mercury Change Management Power license is required to use the Migrators.

This chapter represents a change management implementation recommendation. Using the concepts and procedures listed in this chapter can reduce the risk of down time when rolling-out Mercury ITG processes.

For detailed instructions on using the Migrators, see the *"Migrators Guide and Reference"*.

This chapter discusses the following topics:

Using Multiple Instances - Introduction

Mercury Change Management controls the deployment of objects to mission critical applications. Before rolling-out new or modified functionality in a deployment system, thoroughly test the changes in a Development or Testing instance. For example, before rolling out a new Web Update process to manage deployments to your company's web site, test the Workflow and Object Types used to perform the deployments.

Migrators are used to capture and move configuration data (such as Workflow or Object Type definitions). This enables the sharing of configuration data between multiple Mercury ITG instances. It is then possible to test configurations in a TEST instance, and then migrate the configurations to the PRODUCTION instance.

This chapter represents a change management implementation recommendation. Using the concepts and procedures listed in this chapter can reduce the risk of down time when rolling-out processes.

For additional details, see the following documents:

- *"System Administration Guide"* for instructions on setting up multiple Mercury ITG instances.
- *"Migrators Guide and Reference"* for detailed instructions on using Migrators to move configuration data.
- *"Installation Guide"* for instructions on installing new instances.

Implementing Multiple Instances - Overview

It is recommended multiple instances be used when configuring the Mercury IT Governance Center. The following sections discuss the simplest multi-instance configuration, consisting of two instances: DEV (development) and PROD (production) located on different machines. These basic migration principles can then be extended to support the number of instances used at your site.

There are two implementation scenarios for using multiple instances. The process for implementing these instances differs depending on the following scenarios:

- *Single Production Instance is Currently in Use*
  Requires you to clone the PRODUCTION instance (file system and database) to create the DEV instance.
- *New Implementation*
  Requires you to create multiple Mercury IT Governance instances by running the installation multiple times.

Single Production Instance is Currently in Use

To implement multiple instances when a single Production (PROD) instance is currently in use, clone the PROD instance. Each Mercury IT Governance instance consists of a file system and an Oracle database. These can exist on Unix or Windows machines. Contact the System Administrator for details about your site's configuration.

To move from a single active instance to multiple instances:

1. Clone the PROD instance.

   This includes the file system, database, and license information. Details for this procedure are included in the System Administration Guide. Work with the System Administrator to implement this configuration.

2. Configure any changes to Mercury product in the DEV instance.

   This includes creating or modifying Workflows, Object Types, Request Types, Validations, Security Groups, Environments, etc.

3. Configure a Package Workflow to migrate the configuration data from DEV to PROD.

   This process should be configured in the PROD instance.

4. Migrate data from the DEV instance into the PROD instance.

   Again, this activity is performed from the PROD instance. Therefore, it may be helpful to think of migrating the data as an "import" process.

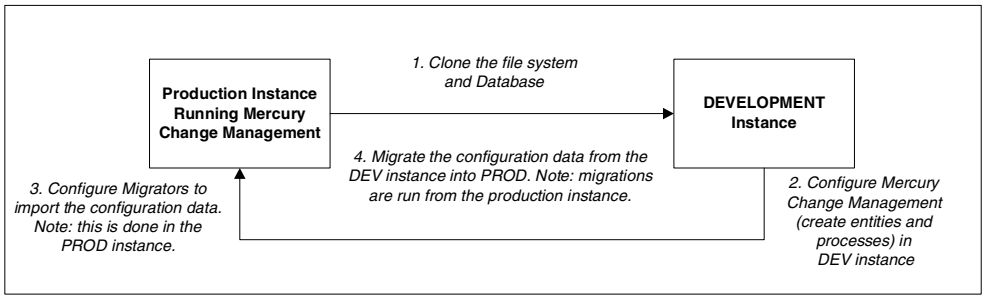


Figure 3-1 Cloning instance and configuring Migrators

New Implementation

When implementing the Mercury IT Governance for the first time, it is possible to immediately set up multiple instances. Configure one instance as the DEV instance, and the other as the PROD instance. By creating two blank instances up front, it is unnecessary to clone existing data from one instance into another. When this strategy is used, follow the instructions included in the *"Migrators Guide and Reference"* document.

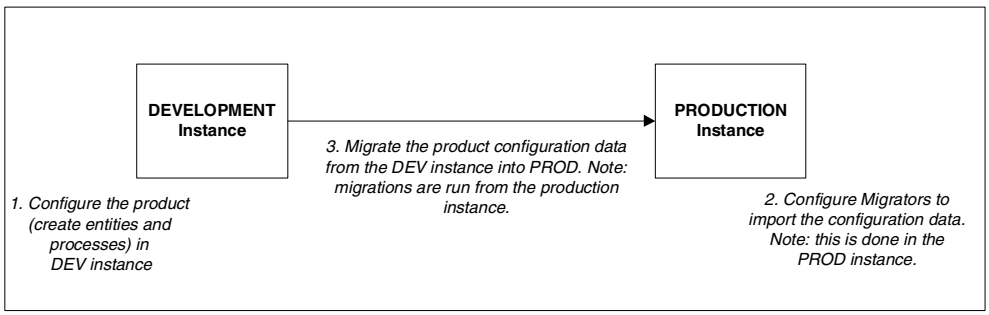


Figure 3-2 Migrating Kintana data between DEV and PROD

Migrating Configuration Data

This section provides an overview of the requirements and processes for using Migrators. This information is provided to help you communicate with the Mercury IT Governance Administrator (who maintains the Mercury ITG instances and license information) and the System Administrator (who maintains the Mercury IT Governance Server) when configuring the product.

This section covers the following topics:

How Migrators Work

Migrators are provided as Object Types in Mercury Change Management. These Migrator Object Types are run through a Workflow. Each supported entity type has its own Object Type. For example, to migrate a Workflow from one instance to another, use the Kintana Workflow Migrator Object Type.

Packages are used to process and audit the migration of configuration changes. When the Package (containing a Migrator object) enters the appropriate execution step in a Workflow, the Migrator's commands are executed. The commands extract the data that defines the entity into text (XML) files. These text files are then imported into the target instance. *Figure 3-3* represents this process.

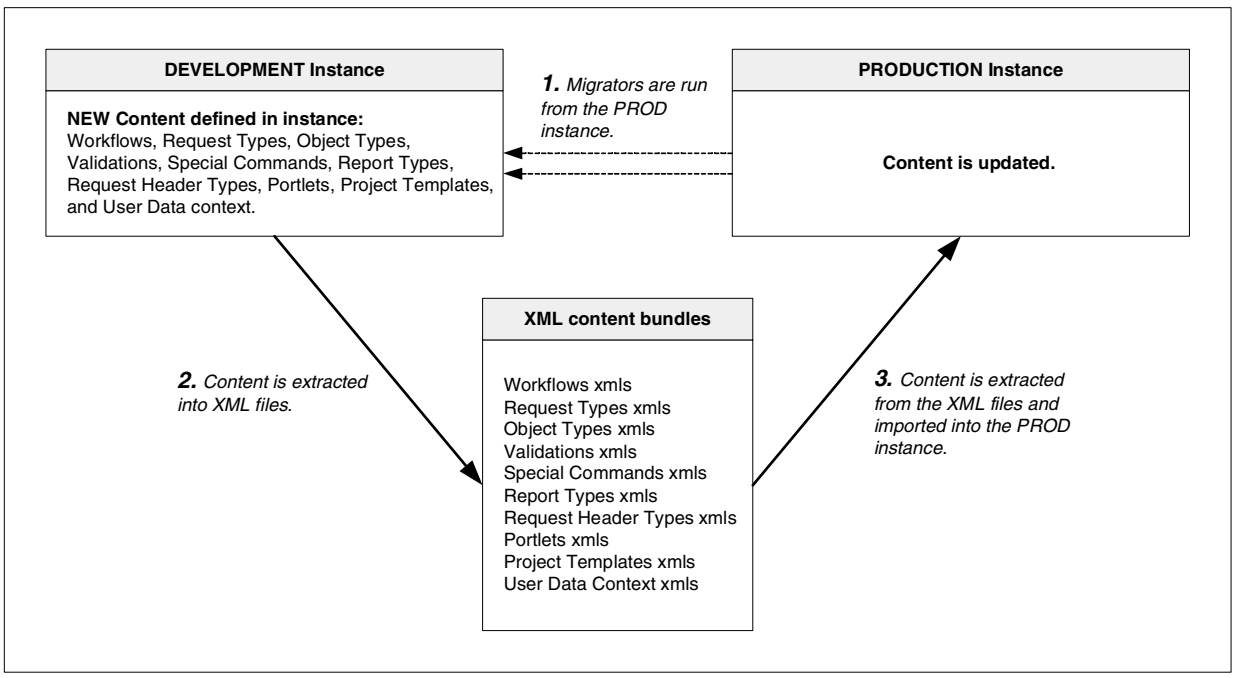


Figure 3-3 Migrator content extraction and import overview

Using the Migrators - Overview

To use the Migrators to migrate configuration information between instances:

1. Configure the appropriate Environments (DEV and PROD) to represent instances involved in the migration.
2. Define a Package Workflow to model the desired migration process.
3. Build a Package using this Workflow, using the appropriate Migrator Object Types to create the Package Lines.
4. Submit the Package and migrate the Package Lines. Use the execution log generated to evaluate the results of the migration.

Detailed instructions for using the Migrators are included in the *"Migrators Guide and Reference"* document.

Instance Requirements for Using Migrators

To use the Migrators:

- At least two working Mercury Change Management instances (DEV and PROD) must be available
- All instances must be accessible over a network

Requirements for PROD Instance

The following items must be configured in the PROD instance. This is the destination instance that will receive the configuration data from the DEV instance.

Requirements for a successful migration include:

- Environments (PROD and DEV) defined in the Environment screen in the Workbench.
- At least one Workflow (Workflow Scope = **Packages**) must be available to run the migration. This Workflow must contain at least one execution step with the source and destination Environments configured to DEV and PROD, respectively.

- Migrator Object Types must be enabled. There is a different Object Type for each of the following entities that can be migrated: Validation, User Data, Special Command, Workflow, Report Type, Object Type, Request Type, Request Header Type, Project Template, Portlet, User Data Contexts.
- The user creating, submitting, and processing the migrations must have a Change Management power license and proper screen access. For details, see the *"Security Model"* guide.

Archiving Configuration Data

Use Migrators to document processes by exporting configuration information to text files. These text files conform to the XML (eXtended Markup Language) specification and are suitable for storage in many archiving systems including source control systems. Source control check-in can be integrated into the Migrator Object Types, allowing organizations to maintain a detailed record of the specific changes made to the production configurations.

To archive configuration data:

1. Perform an Extract only using the appropriate Migrator.

   The content is extracted into .xml files and grouped (by entity) into .zip files.

2. Check the extracts (.zip files) into your source control.

   Each extract contains a file (Source_Descriptor.xml) that describes the Mercury ITG product version and date of extraction.

3. These files can then be imported back into a Mercury IT Governance instance (of the same version) at any point in the future.

Chapter 4

Configuring a Deployment System—Process Overview

This chapter provides an overview of the process used to configure a deployment system in Mercury Change Management. It summarizes each phase of configuration. Additional details for configuration are included in the referenced chapters.

This chapter covers the following topics:

Configuring your Deployment System

To configure a deployment system or process:

1. *Gathering Process Requirements and Specifications*

   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. *Mapping your Process into a Workflow*

   Using the information gathered in the *Gathering Process Requirements and Specifications* chapter, build the Workflow. This includes setting up required Workflow Step Sources, creating Validations to be used by the transitions, and adding Steps and transitions to your Workflow.

3. *Constructing the Object Type*

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

4. *Defining your 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.

5. *Integrating Participants into Your Deployment System*

   After the Workflow and Object Types are constructed, construct security around the process. This includes specifying such things as who can create and process Packages, who can act on a particular Workflow Step, and who can alter the process (Workflow, Object Types, Environments, etc.).

   Note that you will need to work with your Mercury ITG Administrator to configure User and Security Group definitions.

6. *Setting Up Communication Paths*

   Mercury Change Management includes a number of features that enable high visibility into Packages in the deployment process. Create Notifications for Workflow Steps, configure Portlets to provide additional real-time visibility, and use built-in reports.

7. *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.

Example: Configuring a Deployment System

This section provides a sample business case for configuring a deployment system in Mercury Change Management. The following example is used throughout this document to discuss configuration techniques.

Example: Deployment System for Financial Application system

The IT group at ACME Company receives hundred of requests every year for new and enhanced functionality for their FINANCIAL APPLICATION business system. The IT group is also responsible for applying regularly published application patches to this system. This system consists of over ten modules (billing, accounts payable, accounts receivable, fixed asset management, inventory, reporting, payroll, cash management, etc.).

The deployment of changes to the different modules can require unique processing items (destinations, Environment management, post deployment processing steps, etc.). The ACME 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 2:00am on Friday. Any additional system downtime could result in financial loss.

Chapter 5
Gathering Process Requirements and Specifications

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 ITG Dashboard, or reports.

This chapter covers the following topics:

- *Establish Communication Points and Visibility*

Define the Deployment Process

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.

The following sections discuss the specific information that needs to be gathered before configuring the Workflow:

- *Process (Workflow) Considerations*
- *Define the Business Flow*
- *Defining the Technical Flow*
- *Gather Information on Each Step in the Process*
- *Consider Using Subworkflows*
- *Consider Using Release Management*

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 *"Mapping your Process into a Workflow"* on page 65.

Decision versus Execution Steps

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).

Decision Steps are represented on the Workflow by the following default icon:

Execution Steps

Executions are 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 or Environment occurs at an Execution Step.

Execution Steps are represented on the Workflow by the following default icon:

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.

Define the Business Flow

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 (approved, not approved, rework, error, etc.)
- 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 2:00am 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.

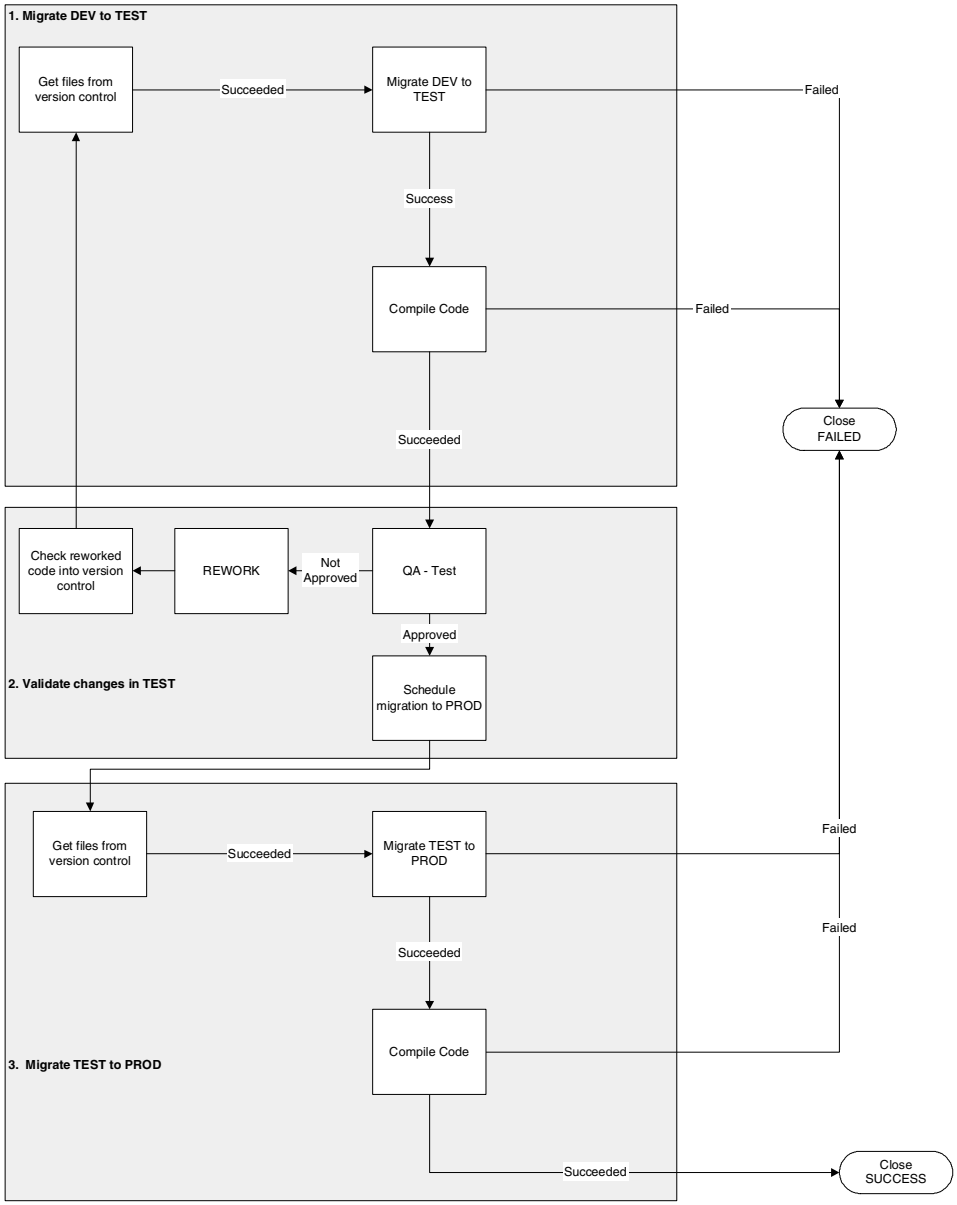


Figure 5-1 Deployment Process - High Level

Defining the Technical Flow

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.

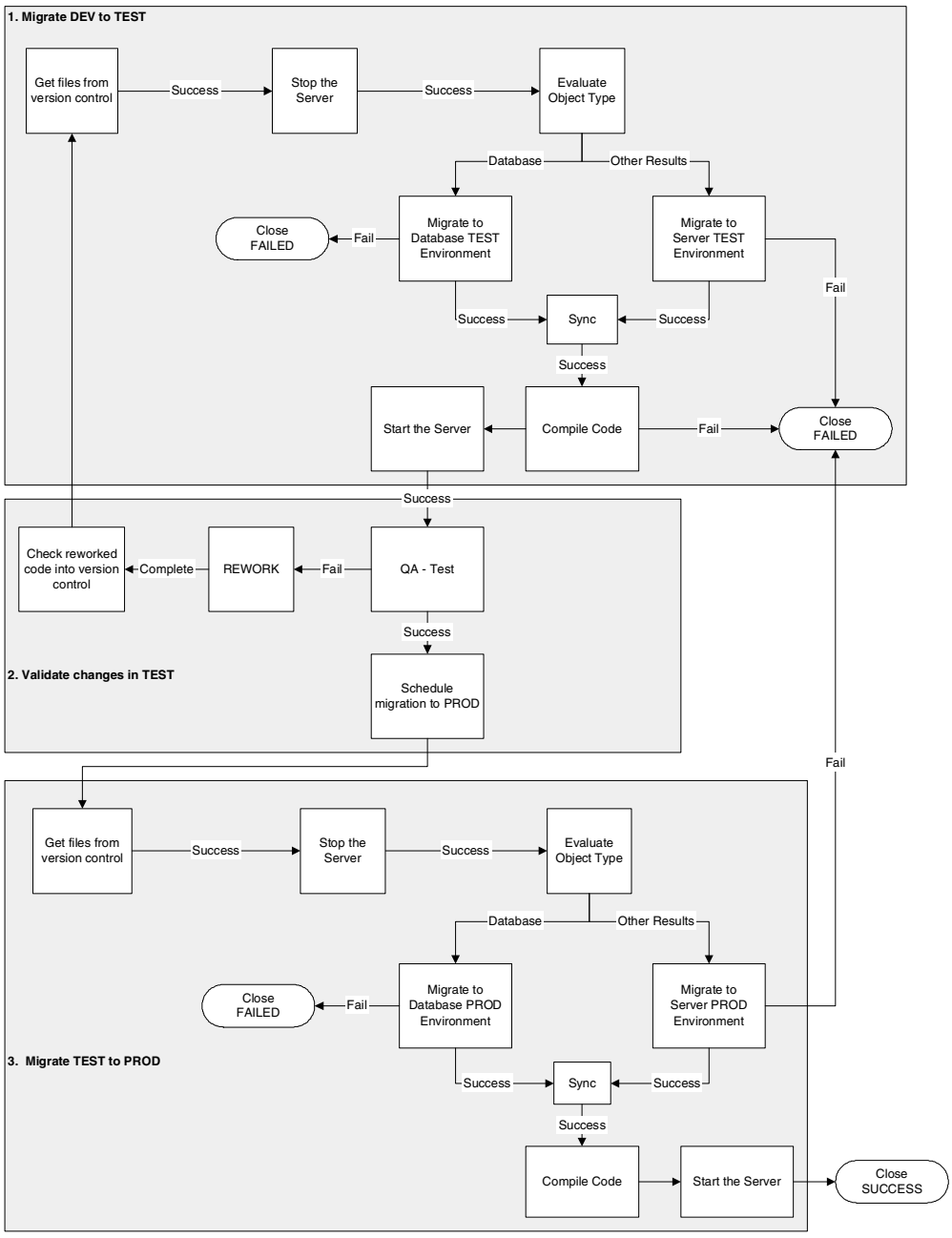


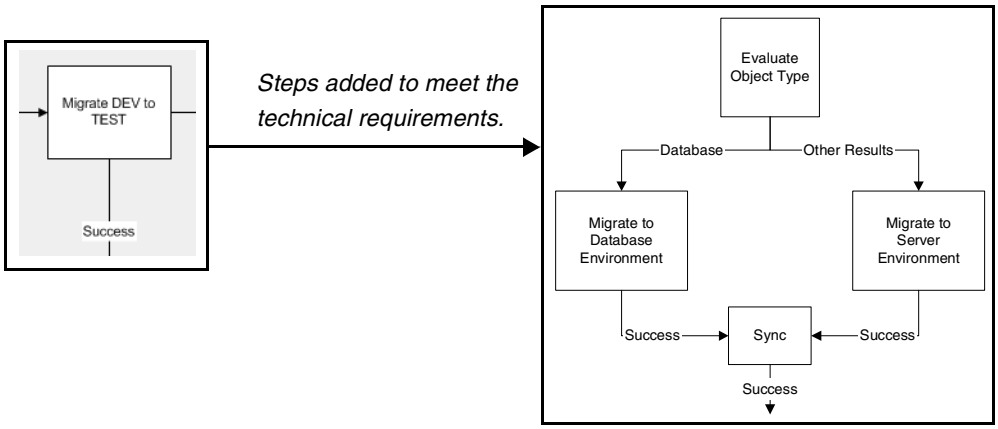
Figure 5-2 Deployment Process - Detailed Level

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

Process change 1:

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.



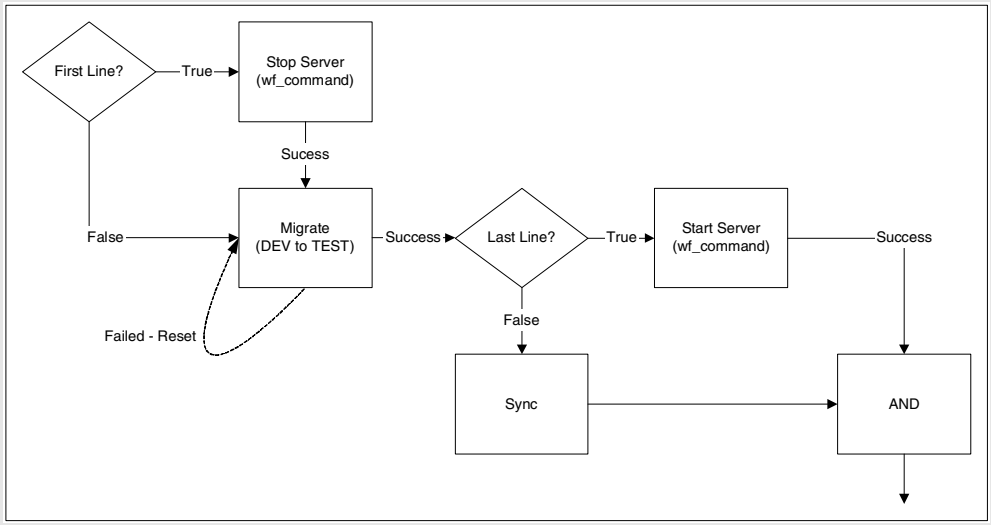
Process change 2:

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.

Tip

It is also possible to use the FIRST LINE and LAST LINE functionality to stop and start the server. The first Package Line (object) that is processed through the Workflow stops the server, and the last Package Line starts it. This is shown in the following diagram:



The Sync step is used to ensure that all Package Lines move in unison to the AND step. When the last line starts the server, it proceeds to the AND step and the Package can continue.

For additional details on this configuration, see *"Advanced Workflow Topics"* on page 255.

Process change 3:

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.

Gather Information on Each Step 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. *"Configuration Worksheets"* on page 409 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:
        - The 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.

For each step, information on Environments, Participants, and Notifications will also need to be collected. This is discussed in the following sections:

- *"Gather Information on Environments"* on page 54
- *"Identify Participants and Security"* on page 56
- *"Establish Communication Points and Visibility"* on page 61

The *"Configuration Worksheets"* on page 409 provide tools for capturing all of the Workflow Step information in one place.

Example: Gathering Workflow Step Information

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

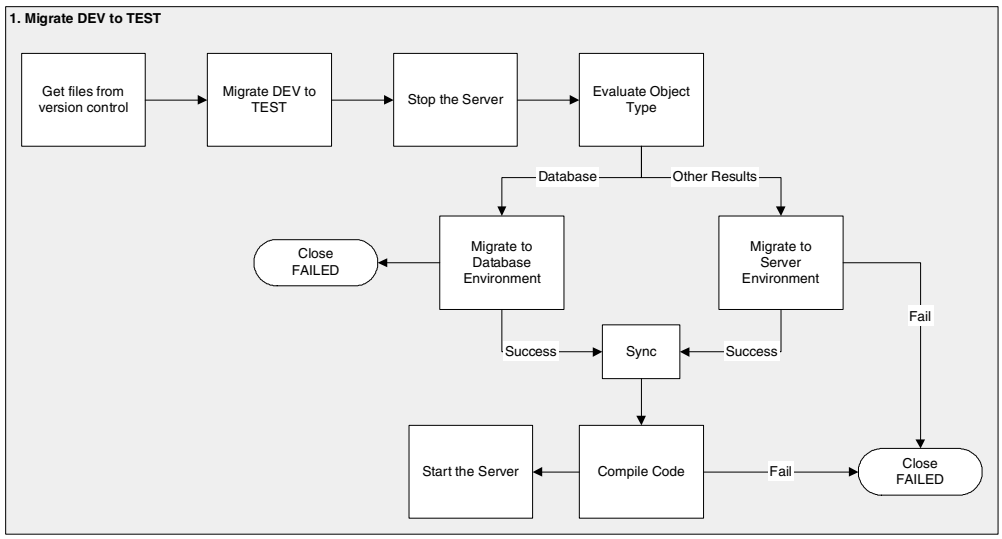


Table 5-1. ACME process Workflow step information

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.

Table 5-1. ACME process Workflow step information

Step Name	Type	Transition Values	Validation	Description
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.

Use the *"Configuration Worksheets"* on page 409 to gather the information related to Workflow security, Notifications, and Environments.

Consider Using 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 sub-processes.

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.

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 a Subworkflow

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 5-3* shows where ACME could implement a Subworkflow.

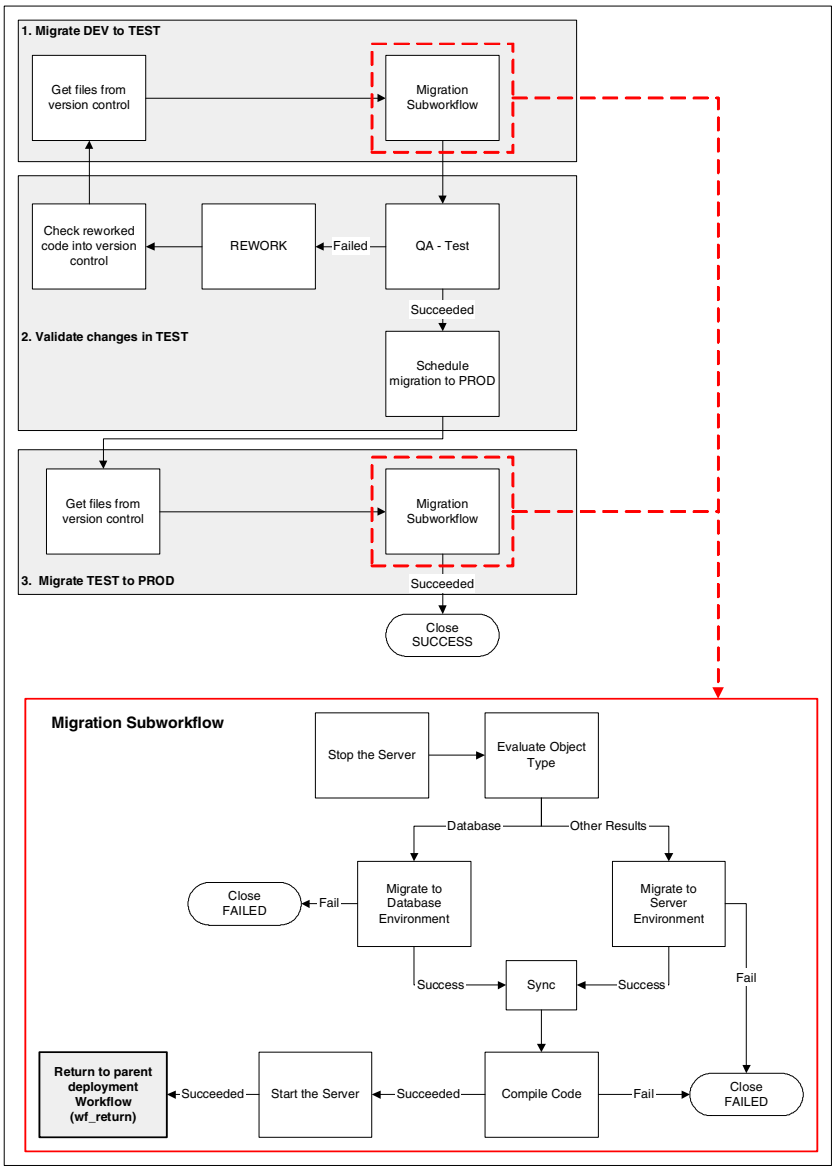


Figure 5-3 Using a Subworkflow in your deployment process (example)

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 ITG 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.

For more information on using Mercury Change Management's Release Management functionality, see *Configuring a Release Management System.*

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.

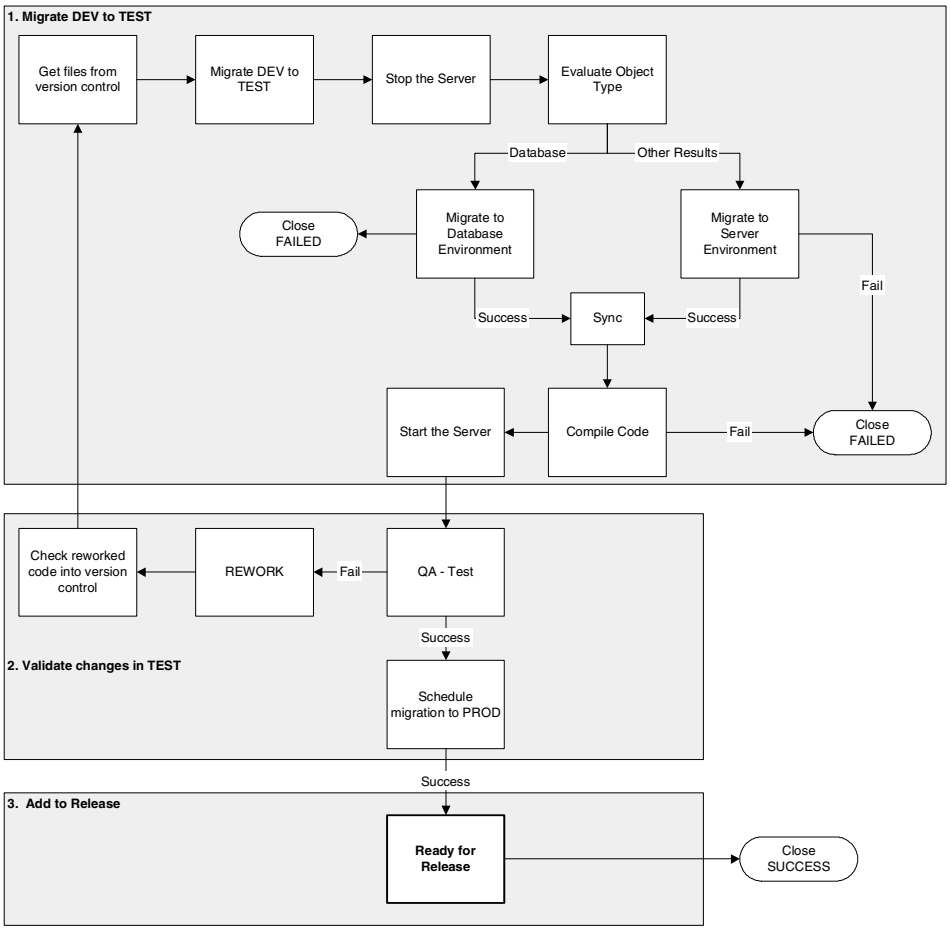


Figure 5-4 Using a Ready for Release step in the deployment process

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, etc. 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, etc.
- Commands contained in the object. Object Type commands often reference information stored in the parameters. These commands are executed at specific points (executions steps) in the Workflow. For additional information, see *"Determine Commands Needed for Objects"* on page 53.

For assistance in collecting the correct data, use the *"Configuration Worksheets"* on page 409.

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 *"Constructing the Object Type"* on page 123.

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 *Figure 5-5*.

Table 1. 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?	NA	NA
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		
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?	NA	NA
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?	NA	NA
Validation type	File 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 4		

Figure 5-5 ACME data collected to describe objects for deployment.

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 and Tokens Guide and Reference*.

The above information can be collected using *"Configuration Worksheets"* on page 409.

Commands		
Goal of Command		
Command Steps		
Conditions (when to run)		

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 *"Configuration Worksheets"* on page 409.

Each Environment must be carefully configured to ensure that passwords, communication protocols, and transfer protocols are configure properly. For instructions on configuring your Environments, see *"Defining your Environments"* on page 153. 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 App codes feature in the Environment definition. For additional details, see *"Using App Codes with Your Environment"* on page 160.

Example: ACME specifies the Environments

ACME determines that the following Workflow Steps require the Environment settings specified below:

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

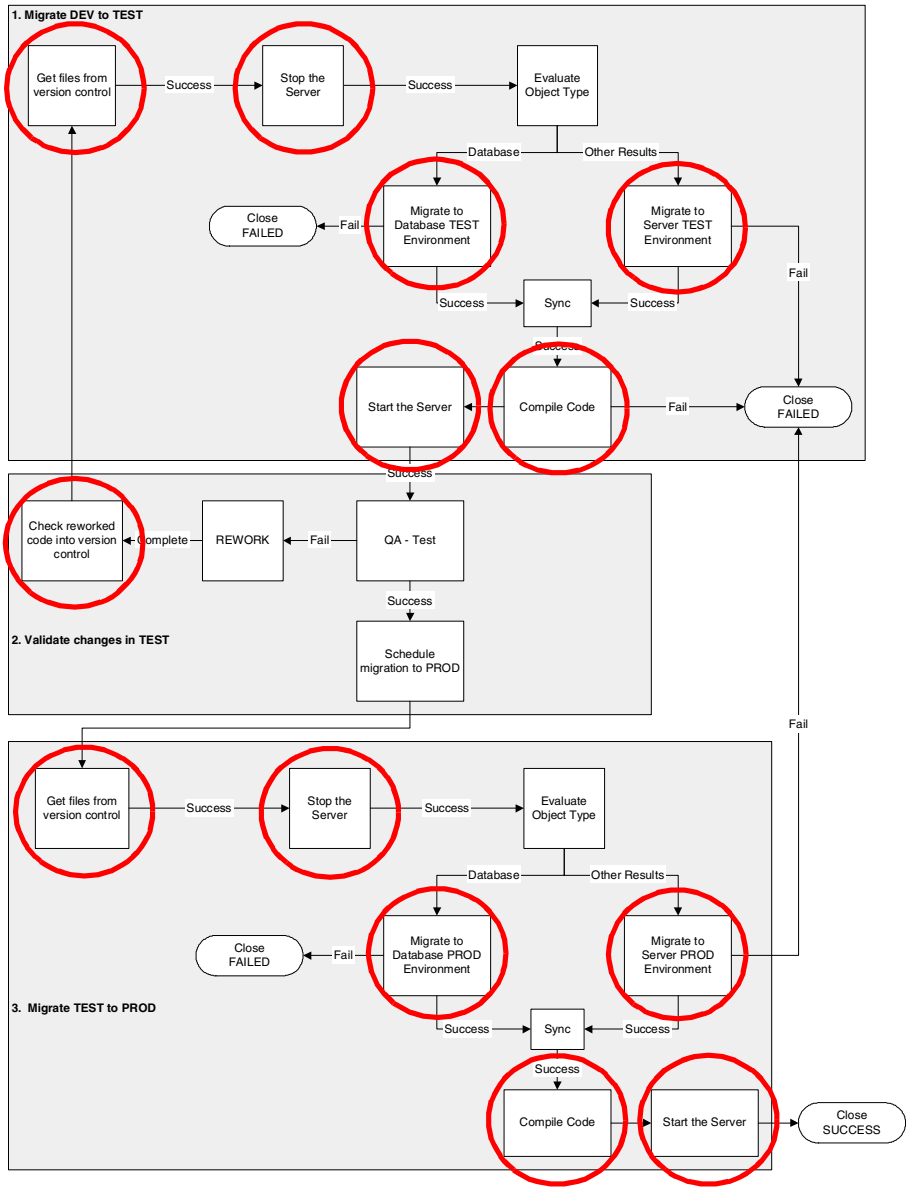


Figure 5-6 Workflow Steps that require Environment specification (circled)

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 *"Enabling Users to Act on a Specific Workflow Step"* on page 198.
  - 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 *"Configuration Worksheets"* on page 409. 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 ITG 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 5-2. 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.

Table 5-2. ACME's Security Groups

Security Group	Members	Responsibilities
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. The information is gathered in the following sections:

- Table 5-3, "ACME Package Creation Security," on page 59
- Table 5-4, "ACME Package Processing Security - Deployment Workflow," on page 60
- Table 5-4, "ACME Package Processing Security - Deployment Workflow," on page 60

Package Creation Security:

Table 5-3. 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 *"Restricting Package Processing to Participants"* on page 200. *Table 5-4* 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 sub-set of the Workflow Steps are included in the below table. To see the process referenced in this table, see *Figure 5-2 on page 41*.

Table 5-4. 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 5-5. 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

Table 5-5. ACME - Security around managing the Process [continued]

Action	Users allowed to perform action	Controlled by: (Users, Security Group, Token)
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

This section lists the information that should be gathered before defining Notifications. For more information on defining and using Portlets and Reports, see:

- *Configuring the Dashboard*
- *Reports Guide and Reference*

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 *"Configuration Worksheets"* on page 409:

Table 5-6. 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 5-7. 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 5-8. 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 6

Mapping your Process into a Workflow

This chapter provides an overview for how to set up a Mercury Change Management Workflow. This includes information on configuring all Workflow Steps, transitions and Validations included in a process.

This chapter discusses the following topics:

Building the Workflow Skeleton - Overview

This section provides a high level overview for building a Workflow.

To create a Workflow:

1. Enter the general Workflow information in a new Workflow window.

   This includes entering the Name and Workflow Scope in the **Workflow** tab.

2. Create any new Step Sources using the Workflow Step Sources window. This includes:

   - Creating Decision Steps.
   - Creating Execution Steps.

- Creating subworkflow steps.
- Creating any new Validations used by the above steps.

3. Add the Workflow Steps to the **Layout** tab.
4. Add transitions between the Workflow Steps.
5. Add Security to the Workflow.
6. Add Notifications to select Workflow Steps.
7. Enable the Workflow.

Required Workflow Settings for Deployment Process

In order to user a Workflow in your deployment process, the following requirements must be met:

- The Workflow Scope must be set to **Packages**
- The Workflow must be **Enabled**
- Add steps and transitions to the **Layout** of your Workflow
- Allow Object Types to be used with the Workflow (**Change Management Settings** tab)
- Enable Security for each Workflow Step. This allows users to act on the step.

- Workflows are created and configured using the Workbench.
- Users must have a Power License and have the proper Access Grants in order to create and edit a Workflow. See *Security Model Guide and Reference*for details.

Copying a Workflow

To copy an existing Workflow:

1. In the Workflow Workbench window, query for the Workflow that you would like to copy.

   The **Results** tab displays the results matching your search.

2. In the **Results** tab, select the Workflow.

3. Click **Copy**.

   The Copy Workflow window opens.

4. In the Workflow Name field, enter the name of the new Workflow.

5. Select the Workflow items that you would like to copy.

6. Click **OK**.

   A Question dialog opens. Click **Yes** if you would like to edit the Workflow.

7. Make any additional edits and click **Save**.

Specifying the First Step in a Process

For each Workflow, the first eligible step in the Workflow can be explicitly defined.

To specify the first step:

1. Open the Workflow window and click the **Workflow** tab.

2. From the First Step field, select the name of the step that you would like to serve as the first step in the process.

   This field contains all of the fields that have been defined for the Workflow.

3. Click **Save**.

Once Workflow Steps have been defined in the **Layout** tab, their sequential order can be set using the arrow buttons in the **Step Sequence** tab. The sequence relates to the graphical position of a step that a user will see when processing an entity, but it does not affect its actual processing. For example, Step A has a display sequence of 3 and Step B has a display sequence of 5. In this situation, Step A will not necessarily be eligible before Step B. Step B could, for example, be specified as the First Step in the process. Additionally, depending on the transition path of the steps, Step A may never be required.

Create the Required Step Source

Mercury Change Management includes a number of standard Workflow Step Sources that can be added to a Workflow. These sources are preconfigured with standard Validations (transition values), Workflow events, and Workflow scope. These available steps specify the following common attributes, which are expected to remain consistent across all Workflows which use that Step Source:

- 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 there is not a Step Source that meets the process requirements, one needs to be created.

If Mercury Change Management 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 Package Line based on the Object Type, copy and use the Check Object Type Workflow Step Source that is delivered with the product.

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:

- Unique Validation leaving the step (This is the step's transition.)
- Unique execution on the step: PL/SQL function, Token, SQL function, or Workflow Step Commands
- Different processing type: immediate vs. manual
- Specific Workflow Scope
- Unique combination of the above settings

The following sections discuss when and how to use specific settings in the Workflow Step Source:

- *Creating a Workflow Step Source - Overview*
- *Creating a Decision Type Step*
- *Create an Execution Type Step*

Creating a Workflow Step Source - Overview

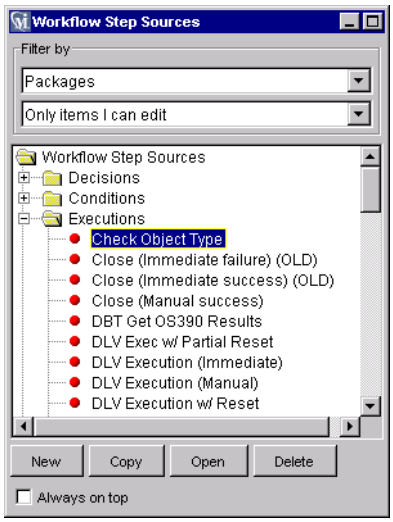
It is possible to create new Workflow Step Sources from the Workflow Step Sources window in the Workbench.

To create a new Workflow Step Source:

1. Click the **Configuration** screen group and click the **Workflows** icon.

   The Workflow Workbench and Workflow Step Sources windows open.

2. Select the Workflow Step Sources window.



3. Select to Filter by **Packages**.

4. Select the folder that corresponds to the type of Workflow Step Source that will be created.

   For example, to create an Execution Step, select the **Executions** folder.

5. Click **New**.

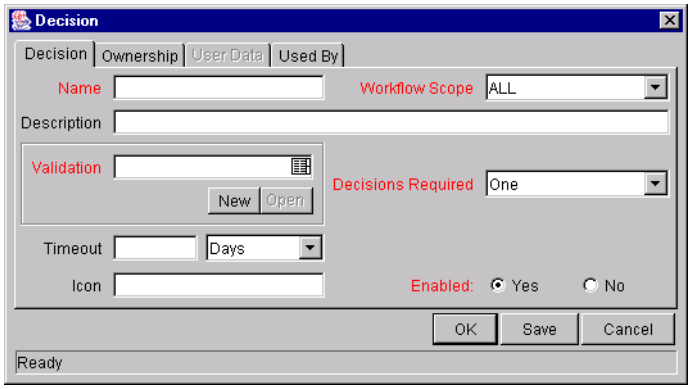
   This opens a window that corresponds to the selected Workflow Step Source type. The Decision and Execution windows are shown below. For information on configuring Workflow Step Sources, see *"Creating a Decision Type Step"* on page 73 and *"Create an Execution Type Step"* on page 77.

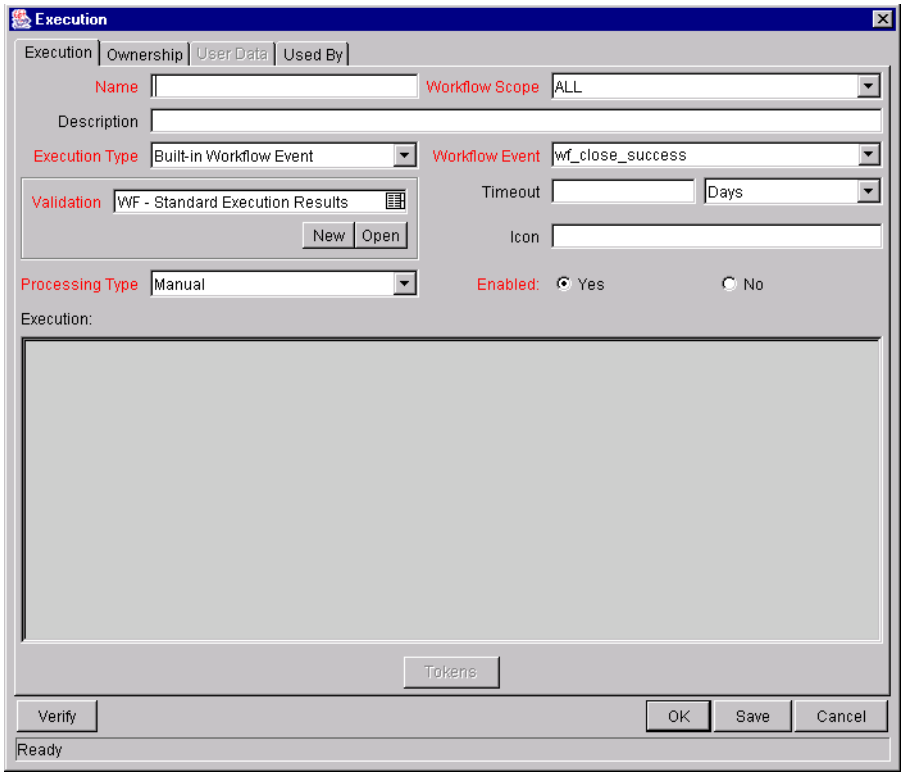
Note

Condition Step Sources cannot be added, deleted or modified. Mercury Change Management supports a set number of process Condition steps. These can be added to the Workflow layout just as any other Workflow Step Source.

If a Condition step is selected in the Workflow Step Sources window, the **New** button will not be enabled.

For more information, see *"Using Condition Steps"* on page 269.





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

   For detailed information about setting up the steps, see *"Creating a Decision Type Step"* on page 73 and *"Create an Execution Type Step"* on page 77.

7. To be able to use this step in a Workflow, select **Yes** in the Enabled radio button.

8. Click the **Ownership** tab.

   Select which Ownership Groups will have the ability to edit this Execution or Decision.

9. To save the changes and close the Execution or 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.

Related Topics:

- *"Creating a Decision Type Step"* on page 73
- *"Create an Execution Type Step"* on page 77
- *"Advanced Workflow Topics"* on page 255

Workflow Step Source Configuration and Usage 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 be changed, copy the Step Source and configure a new Validation.
- The Workflow Step Source must be Enabled to use them on a Workflow.
- Step sources can only be added to a Workflow when the Workflow has a matching Workflow Scope, or the Step Source has a scope of **All**.
- A step from a Workflow that has been used to process a Package can not be deleted. This would compromise data integrity. Instead of deleting the step, remove all transitions to and from the step and disable it. Select Display = **Never** in the Workflow Step window to hide the step from users who are processing Packages using that Workflow.

Creating a Decision Type Step

To create a Decision step:

1. *Enter the general information on the Decision Step Source* (name, scope, description).
2. *Select a Validation*.
3. *Specify the voting requirements on the step*.
4. *Specify the default timeout value*.

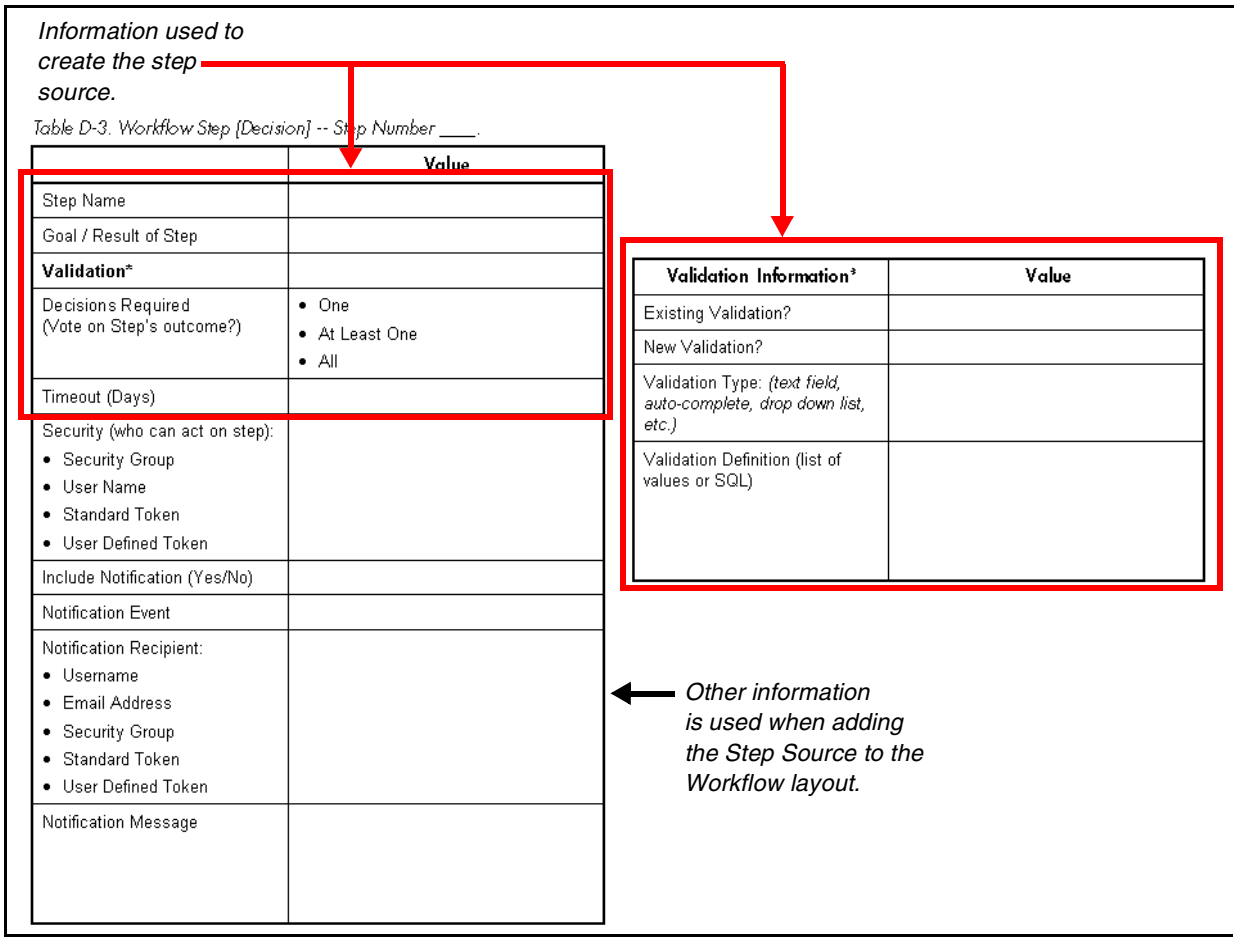


Figure 6-1 Worksheet - Information used to create the Decision Step source.

Enter the general information on the Decision Step Source

Enter the following information in the Decision window.

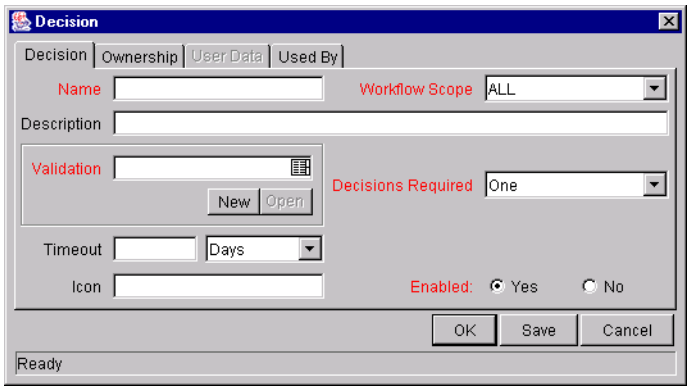


Table 6-1. Decision Step Source worksheet to window.

Field in Decision Window	Description
Name	This is the name that describes the 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. This should be set to **ALL** or **Packages** for deployment processes.
Description	Description of the Step Source.
Validation	Specifies the possible values that can exit the Workflow Step. See *"Configure the Step's Transition Values (Validation)"* on page 95.
Decisions Required	This specifies the number of people who need to approve a specific step. See *"Specify the voting requirements on the step"* on page 75.
Timeout	Specifies how long (in specific units) the Step can stay eligible for before either completing with a value of **Timeout** (if Decisions Required is **All** or **One**) or a result (if Decisions Required is **At Least One**). If this Workflow Step remains eligible for the value entered in the Timeout value, the Package can be configured to send an appropriate Notification, as well as escalate to other steps in the Workflow.

Table 6-1. Decision Step Source worksheet to window.

Field in Decision Window	Description
Icon	A different graphic can be specified to represent steps of this source for use on the Workflow layout tab. This graphic needs to exist in the `icons` subdirectory on the Mercury ITG server. All icons should be in .gif format.
Enabled	The Step Source must be enabled in order to add it to the Workflow layout.

Select a Validation

Select a Validation that has the transition values required for leaving the step. If a Validation that meets the requirements is not available, create a new one from the Workflow Step Source window.

For additional details, see *"Configure the Step's Transition Values (Validation)"* on page 95.

Specify the voting requirements on the step

When a Decision step is defined, the number of decisions required for that Workflow Step can be defined. *Figure 6-2* displays the available options for the Decisions Required field.

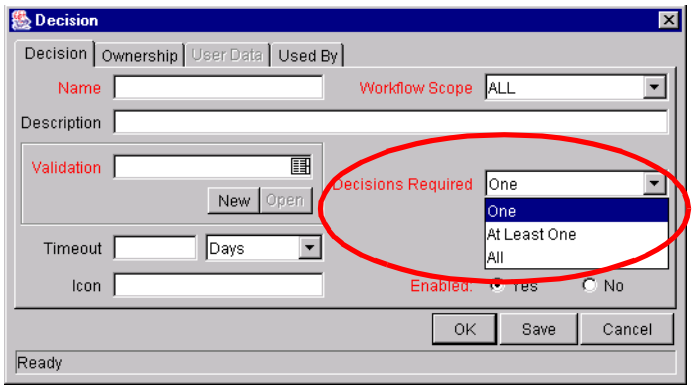


Figure 6-2 Decision Window - Decisions Required Drop Down List

The following choices are available for the Decisions Required:

- **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**

  A Timeout period must be defined to use this choice. When **At Least One** is selected for the Workflow Step, the 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.

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. For more information, see *"Adding Transitions Between Steps"* on page 109.

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**

  The All step is also commonly 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**.

Specify the default timeout value

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** or **One**) or completing with a result (if Decisions Required is **At Least One**). Timeouts can be by minute, hour, weekday or week. Timeout parameters for Executions and Decisions are a combination of a numerical timeout value and a timeout unit (such as weekdays).

If a Workflow Step remains eligible for the value entered in the Timeout value, the Package can send an appropriate Notification and proceed to other steps in the process.

Timeouts can be uniquely configured for each Workflow Step in the **Layout** tab. The timeout value specified in the Step Source acts as the default timeout value for the step. When adding a step to the Workflow using this Step Source, specify a different timeout value for the step.

Create an Execution Type Step

To create an Execution step:

1. *Enter the general information on the Execution Step Source* (name, scope, description).
2. *Define the Executions*.
3. *Select a Validation*.
4. *Specify the default timeout value*.

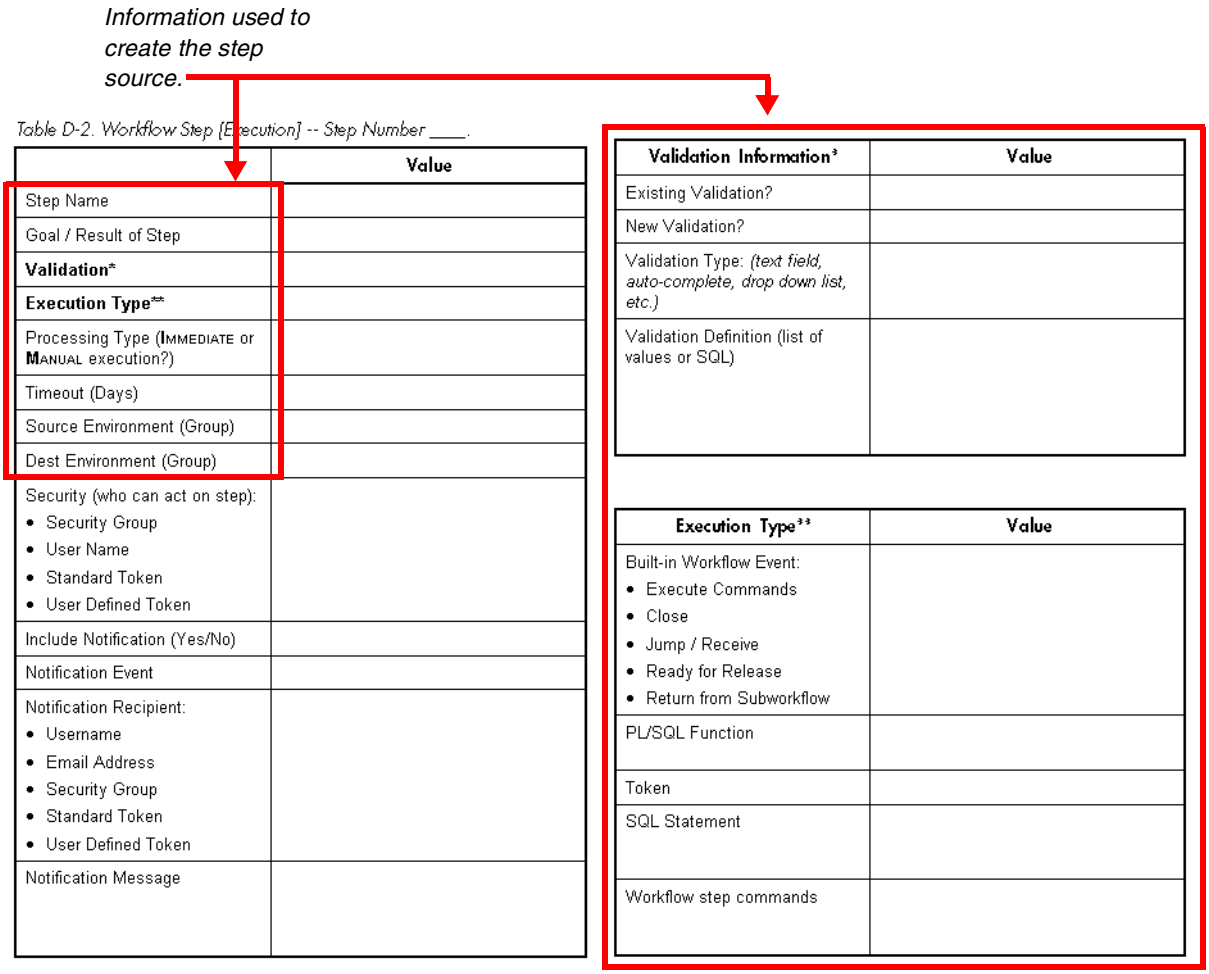


Figure 6-3 Worksheet - Information used to create the Execution Step source.

Enter the general information on the Execution Step Source

Enter the following information in the Execution window.

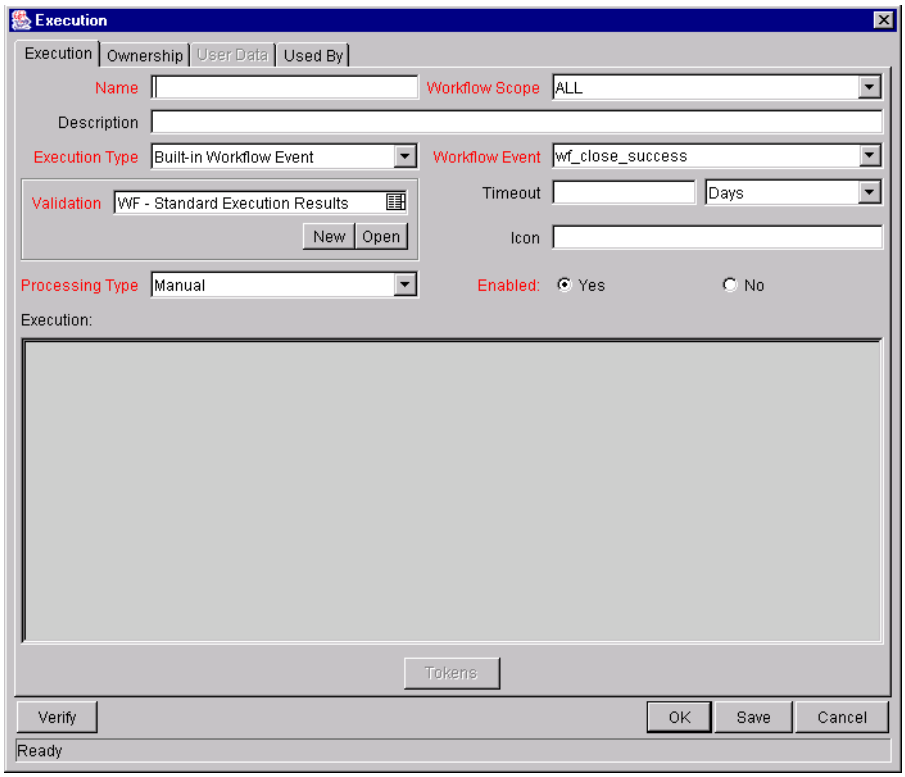


Table 6-2. Execution Step Source worksheet to window.

Field in Execution Window	Description
Name	This is the name that describes the 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. This should be set to **ALL** or **Packages** for deployment processes.
Description	Description of the Step Source. This should specify the execution that will occur.

Table 6-2. Execution Step Source worksheet to window.

Field in Execution Window	Description
EXECUTION TYPE	Used to select the type of execution to be performed. The choices include: • **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 Step. • **PL/SQL Function**: Runs a PL/SQL function and returns its result as the result for the Step. • **Token**: Calculates the value of a Token and returns its value as the result for the Step. • **Workflow Step Commands**: Executes a set of commands, independent of an Object, at a Workflow Step.
WORKFLOW EVENT - (Built-In Workflow Event)	For Executions of type **Built-in Workflow Event**, the specific event to perform must be selected. The available choices in the drop down list depend on which Workflow Scope has been selected. The choices include: • **execute_object_commands**: Executes the Object Type commands for a Package Line. • **rm_ready_for_release:** Sets a Package Line ready to be added to a Mercury Change Management Release. • **wf_close_success**: Sets the Package Line as closed with an end status of 'Success.' • **wf_close_failure**: Sets the Package Line as closed with an end status of 'Failed.' • **wf_jump:** (Mercury Change Management and Mercury Demand Management only) Instructs the Workflow to proceed to a corresponding Receive Workflow Step in another Workflow. • **wf_receive:** (Mercury Change Management and Mercury Demand Management only) 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 only) Used to route a Subworkflow process back to its parent Workflow.

Table 6-2. Execution Step Source worksheet to window.

Field in Execution Window	Description
PL/SQL FUNCTION	For Executions of type PL/SQL Function, the actual function to run. The results of the function will determine the outcome of the step. Note: The results of the function must be a subset of the Validation values for that step.
TOKEN	For Executions of type Token, the Token that will be resolved. The results of the Token resolution will determine the outcome of the step.
SQL STATEMENT	For Executions of type SQL Statement, the actual query to run. The results of the query will determine the outcome of the step. Note: The results of the query must be a subset of the Validation values for that step.
WORKFLOW STEP COMMANDS	For Executions of 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	Indicates whether the Execution is performed immediately (**Immediate**) when the Step becomes eligible or whether the Execution needs to be manually activated by a user (**Manual**).
Validation	Specifies the possible values that can exit the Workflow Step. See *"Configure the Step's Transition Values (Validation)"* on page 95.
Timeout	If this Workflow Step remains eligible for the value entered in the Timeout value, the Package can be configured to send an appropriate Notification, as well as escalate to other steps in the Workflow. See *"Specify the default timeout value"* on page 95.
Icon	You can select a different graphic to represent this steps of this Step Source. This graphic needs to exist in the `icons` subdirectory on the Mercury ITG server. All icons should be in .gif format.
Enabled	The Step Source must be enabled in order to add it to the Workflow layout.

Define the Executions

Execution steps are used to perform specific actions. Mercury Change Management provides a number of number of built in Workflow events for processing some common execution events (running Object Type commands, closing a Package, etc.). It is also possible to create custom executions based on SQL, PL/SQL, Token resolution, and custom commands.

This section discusses when to use specific types of executions and provides references for configuring these executions.

Execution steps can be created to perform the following actions:

- *Execute the Object Type Commands* and transition based on the success or failure of those commands.
- *Close the Package Line and mark it as a Success*:
- *Close the Package and mark it as Failed*
- *Transition (jump) to a Workflow that is Processing a Request*
- *Receive control from a Workflow that is Processing a Request*
- *Set a Package "Ready for Release" for use with Release Management Functionality*:
- *Return from a Subworkflow to the Parent Workflow*:
- *Execute a PL/SQL function and then transition based on the result*:
- *Execute a SQL statement and then transition based on the result*
- *Evaluate a Token and then transition based on the result*
- *Execute a number of system level commands and then transition based on the success or failure of those commands.*
    - Example: Start a server
    - Example: Stop a server

Execute the Object Type Commands

Different objects (stored on 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 a per-Object Type basis. The Workflow

can then be configured to execute the Object Type commands at a specific step in the process. Each object (Package Line) will run its own commands, ensuring the correct execution for that Object Type.

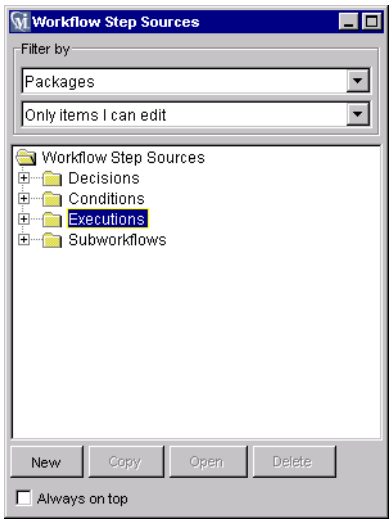
Note

Mercury Change Management includes a system Step Source that executes the Object Type commands. Use and modify (if necessary) a copy of this Step Source to process this action.

Step Source = **DLV Execution w/ Reset**

To create an Execution Step source that will execute the Object Type commands for each Package Line in your Package:

1. Open the Workflow Workbench.
2. Select the Workflow Step Sources window.
3. Select the Execution directory.

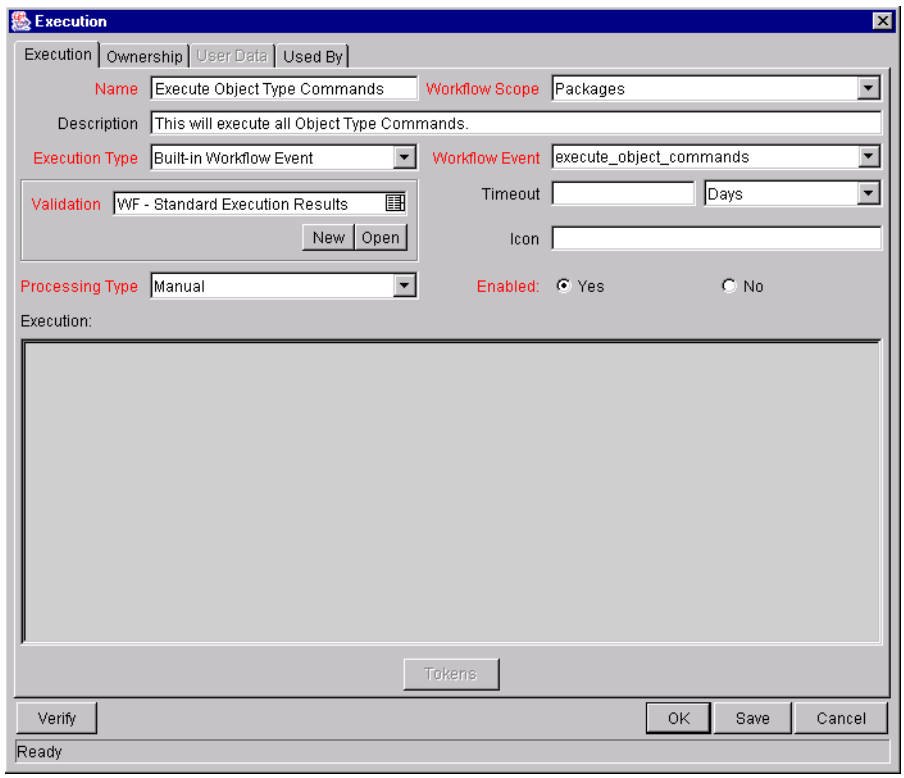


4. Click **New**.

   The Execution window opens.

5. Enter the following information:

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**



Close the Package Line and mark it as a Success

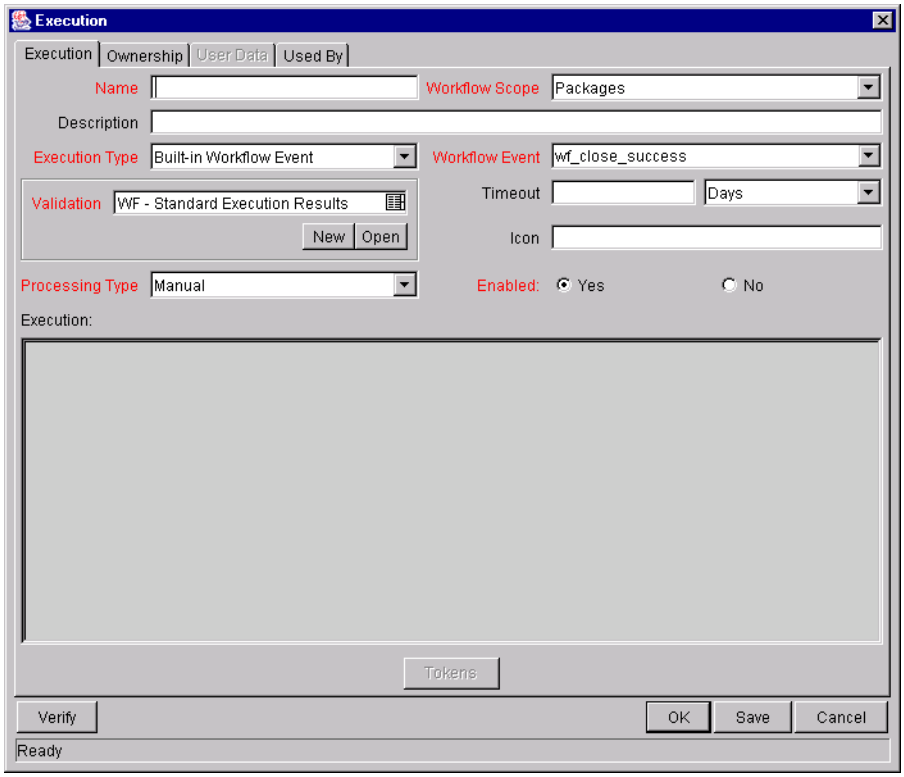
It is possible to create an Execution Step that closes a Package Line. When all Package Lines are closed, the Package will close. Each Package Workflow should resolve with a closed Package. Later, the Packages that were closed successfully can be reported on.

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.

To configure an Execution Step source to close a Package Line and mark it as a Success:

1. Create an Execution Step source with the following settings:

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	**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**



Tip

Mercury Change Management includes a system Step Source that performs this task. Use this Step Source unless it does not meet the required specifications (such as Validation or Processing Type).

Step Source = **Close (Immediate success)** or **Close (Manual success)**

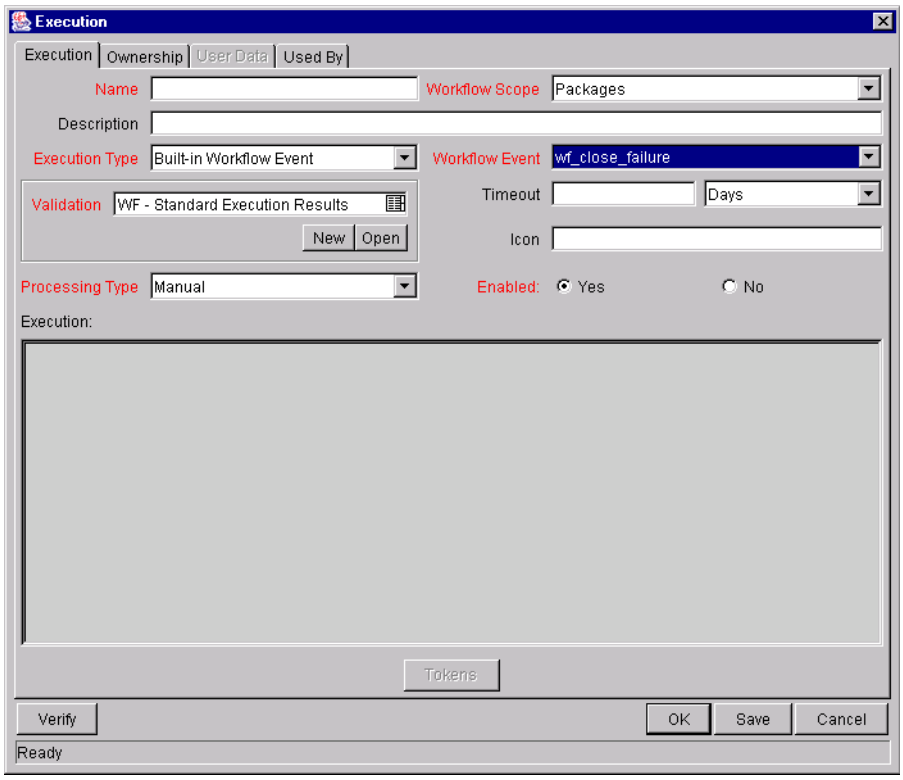
Close the Package and mark it as Failed

To configure an Execution Step source to close a Package and mark it as a Failed:

1. Set the following in the Execution window:

Field in Execution Window	Value
Name	Enter a descriptive name for the Step Source.

Field in Execution Window	Value
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**



Tip

Mercury Change Management includes a system Step Source that performs this task. Use this Step Source unless it does not meet the required specifications (such as Validation or Processing Type).

Step Source = **Close (Immediate failure)**

Transition (jump) to a Workflow that is Processing a Request

Package Workflows can communicate with Request Workflows at specific jump and receive points. To effectively utilize this functionality, you need to properly configure both the jump and receive execution Workflow Steps. See *"Package - Request Workflow Integration"* on page 260 for additional details.

Receive control from a Workflow that is Processing a Request

Package Workflows can communicate with Request Workflows at specific jump and receive points. To effectively utilize this functionality, properly configure both the jump and receive execution Workflow Steps. For additional details, see *"Package - Request Workflow Integration"* on page 260.

Set a Package "Ready for Release" for use with Release Management Functionality

To configure an Execution Step source to feed a Package into a Release, set the following in the Execution window:

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	**rm_ready_for_release**
PROCESSING TYPE	**Manual** or **Immediate**
Validation	**RM - Ready for Release**
Enabled	**Yes**

Mercury Change Management includes a system Step Source that performs this task. Use this Step Source unless it does not meet the required specifications (such as Validation or Processing Type).

Step Source = **Ready for Release**

For more detailed information on using the Release Management functionality, see *Configuring a Release Management System*.

Return from a Subworkflow to the Parent Workflow

Execution steps can be configured to automatically return from a subworkflow to its parent Workflow. Create an Execution Step (to be used on the Subworkflow) with the following configuration:

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	**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**

See *"Using Subworkflows"* on page 255 for additional details.

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. 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.

Mercury Change Management a system Step Source that performs this task. Use this Step Source unless it does not meet the required specifications (such as Validation or Processing Type).

Step Source = **Return from Subworkflow**

Execute a PL/SQL function and then transition based on the result

A PL/SQL function Execution Step runs a PL/SQL function and returns its results as the result of that Workflow Step. Include an Execution Step with the following source configuration:

Field in Execution Window	Value
Name	Enter a descriptive name for the Step Source.
Workflow Scope	**Packages**
EXECUTION TYPE	**PL/SQL Function**
PROCESSING TYPE	**Manual** or **Immediate**
Validation	Select or create a Validation that includes all of the possible values of the SQL query. Tip: 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**

Execute a SQL statement and then transition based on the result

SQL statement Execution steps are used when a Workflow needs to be routed based on the result of a query. A SQL statement Execution Step runs a SQL query and returns its results as the result of that Workflow Step.

Include an Execution Step with the following source configuration:

Field in Execution Window	Value
Name	Enter a descriptive name for the Step Source.
Workflow Scope	**Packages**
EXECUTION TYPE	**SQL Statement**
PROCESSING TYPE	**Manual** or **Immediate**
Validation	Select or create a Validation that includes all of the possible values of the SQL query. Tip: 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**

Configuration notes:

- Only use Select statements
- Tokens can be used within the WHERE clause
- Query must return only 1 value

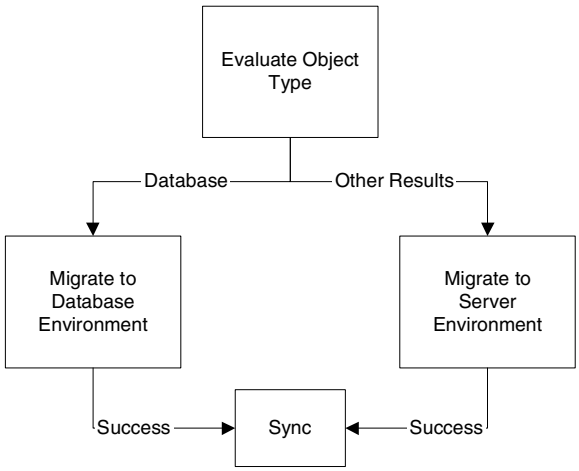
Evaluate a Token and then transition based on the result

Mercury Change Management includes Workflow Execution steps that may be used to set up data-dependent rules for the routing of Workflow processes. Token Execution steps enable a Workflow to be routed based on the value of any field within a particular entity. A Token Execution step references the value of a given Token and uses that value as the result of the Workflow Step.

A transition may be made based on the stored value stored by using Tokens in the Execution step. Include an Execution Step with the following source configuration:

Field in Execution Window	Value
Name	Enter a descriptive name for the Step Source.
Workflow Scope	**Packages**
EXECUTION TYPE	**Token**
PROCESSING TYPE	**Manual** or **Immediate**
Validation	Select or create a Validation that includes all of the possible values of the resolved Token. Tip: To create a Validation validated by SQL, include the same Token in the SQL.
Execution	Enter the Token for the value that the transition will be based on.
Enabled	**Yes**

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



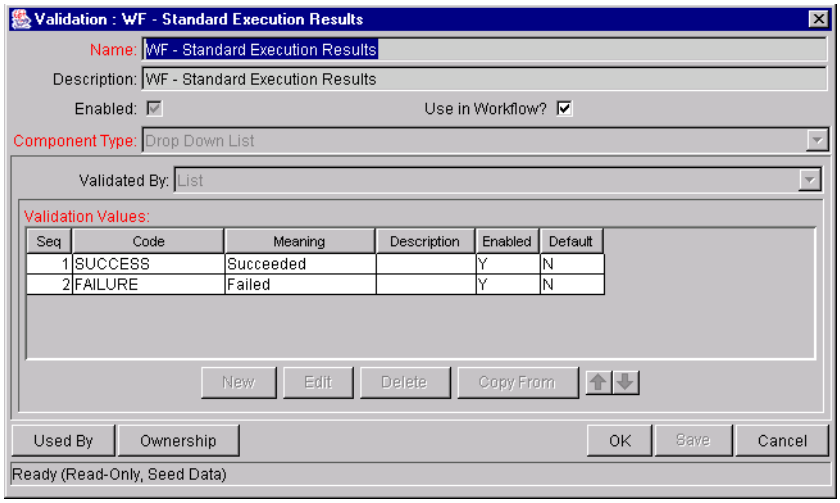
ACME decides to use an Execution step to automatically evaluate the type of object and route the Package line accordingly. To accomplish this, an Execution Step source (Evaluate Object Type) has been, configured with the following parameters.

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**

Execute a number of system level commands and then transition based on the success or failure of those commands.

System level commands can be run for Execution Steps of the following Execution Type: **Built-in Workflow Event** (**execute_object_commands**) and **Workflow Step Commands**. When either the Workflow or the Object Type commands execute at this step, the commands will either Succeed or Fail. To transition based on these results, the code for the Validation values must have the following values:

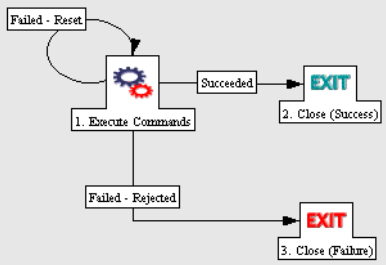
- SUCCESS
- FAILURE



Tip

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. To configure this:

1. Create an Execution Step source to execute the Workflow or Object Type commands.
2. Create a Validation with the following Validation Values.
   a. SUCCEEDED
   b. FAILED
   c. FAILED - RESET
   d. FAILED - REJECTED
3. Add the step to the Workflow **Layout** tab.
4. Add transitions based on the following Specific Results:
   a. SUCCEEDED
   b. FAILED - RESET -- set the transition to return back into the same step.
   c. FAILED - REJECTED



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 Package Line step and select Failed - Retry. The execution will re-run.

Select a Validation

Select a Validation that has the transition values required for leaving the step. If there is not already a Validation that meets the requirements, create a new one from the Workflow Step Source window.

For additional details, see *"Configure the Step's Transition Values (Validation)"* on page 95.

Specify the default timeout value

Timeouts in the Execution Steps can be set at two levels:

- Step level: the amount of time that a step is eligible before completing with an error. This is set in the Execution window.
- Command level: 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. It is set in the Command window.

Timeouts can be by minute, hour, weekday or week. Timeout parameters for Executions and Decisions are a combination of a numerical timeout value and a timeout unit (such as weekdays).

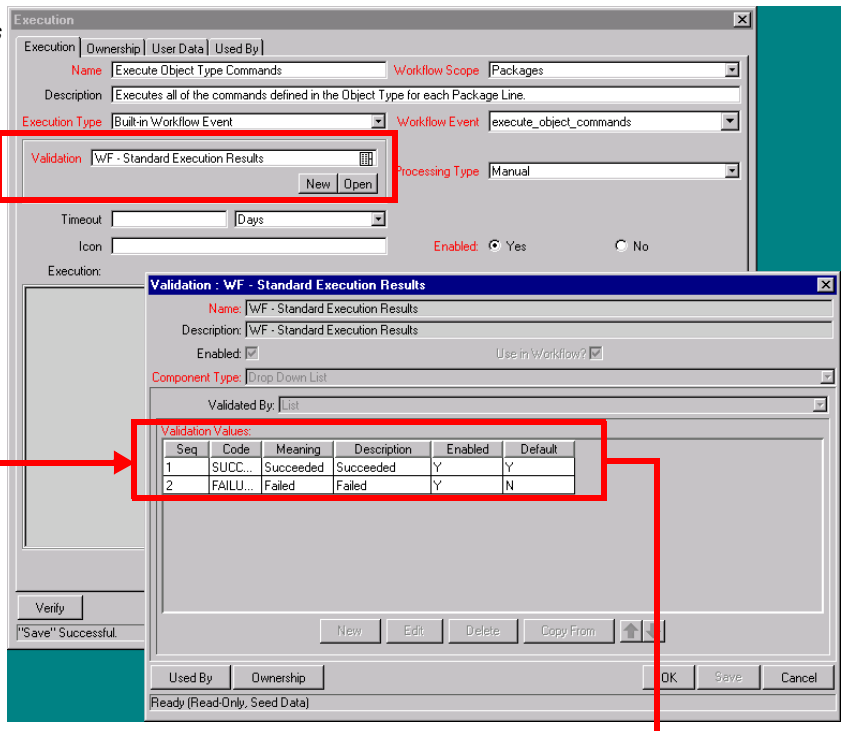
If this Workflow Step remains eligible for the value entered in the Timeout value, the Package can send an appropriate Notification and proceed to other steps in the Workflow.

Timeouts can be uniquely configured for each Workflow Step in the **Layout** tab. The timeout value specified in the Step Source acts as the default timeout value for the step. When adding a step to the Workflow using this Step Source, specify a different timeout value for the step.

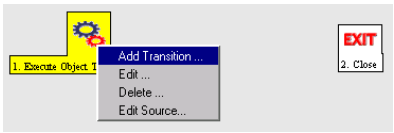
Configure the Step's Transition Values (Validation)

Workflows can be configured to transition based on values automatically returned from an execution or values selected by the user. For each Workflow Step, define all of the possible values for the step's transition. This is set in the Validation field on the Execution window or the Decision window. The Validation dictates the values in the Specific Result section on the Add Transition window.

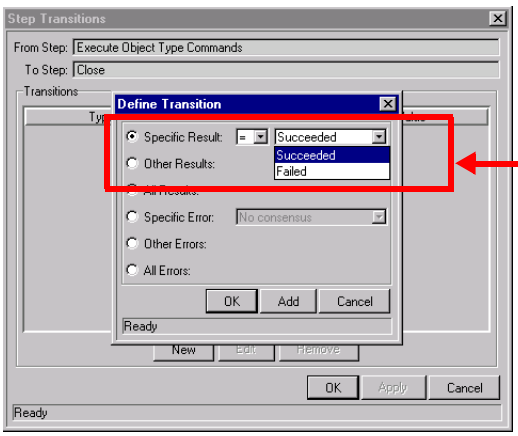
1. Validation specifies all possible results for the step.



2. Add a transition between two steps in the Workflow Layout tab.



3. Optionally base the transition on the values defined in the Validation.



When specifying the Validation for the Execution Step source, specify all possible transition values in the Validation. When using that Step Source on the Workflow (add it to the Layout tab), decide whether to transition on one of the specific results, or a number of other transition options:

- Other Results
- All Results
- Specific Error
- Other Errors
- All Errors

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 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.

The following table summarizes this relationship between Validations and Execution types.

Table 6-3. Relationship between Validation and Execution Type

Execution Types	Validation Notes
Built-in Workflow Event and Workflow Step Commands	Typically use a variation of the WF - Standard Execution Results Validation (Succeeded or Failed). A few of the Workflow Events have specific Validation Requirements: wf_return, wf_jump, wf_receive.
PL/SQL Function	Validation must contain all possible values returned by the function.
Token	Validation must contain all possible values for the Token.
SQL Statement	Validation must contain all possible values for the SQL query. Tip: You can use the same SQL in the Validation (drop down or auto-complete list) minus the WHERE clause.

Use the information captured in the *"Configuration Worksheets"* on page 409 to construct the Validation.

Consider copying existing Validations to save time and ensure that the SQL or other Validation technique is configured properly.

Add Steps and Transitions to the Workflow Layout

Build the process graphically by dragging and dropping Workflow Step Sources onto the Workflow window's **Layout** tab. When a Workflow Step Source is included in a Workflow, it is then referred to as a "Workflow Step." If a Step Source does not exist that meets the necessary requirements (such as decision, execution, correct transition Validation values, or processing type) one will need to be created.

When adding the Step Source to the **Layout** tab, supplemental information will need to be provided. The following sections discuss the configuration required when:

- *Adding Decision Steps*
- *Adding Execution Steps*
- *Adding a Subworkflow*
- *Adding Transitions Between Steps*

Adding Decision Steps

To add a Decision step to your Workflow:

1. Drag the Step Source onto the Workflow's **Layout** tab.
2. *Enter the general information on the Decision step*
3. *Specify the Security*
4. *Configure Notifications for the Workflow Step*
5. *Specify the Timeout value for the step*

Figure 6-4 shows the information used when creating a Decision Step.

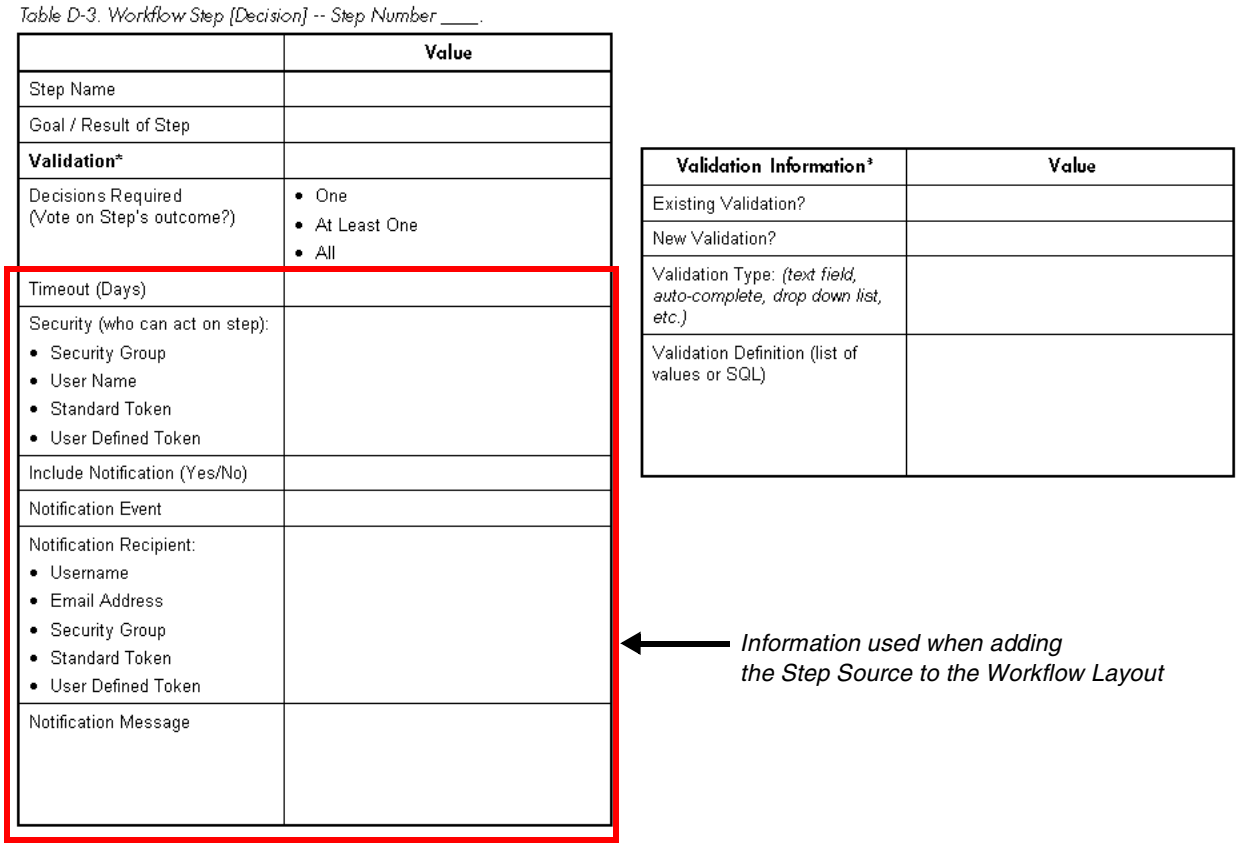


Table D-3. Workflow Step [Decision] -- Step Number ___.

	Value
Step Name	
Goal / Result of Step	
Validation*	
Decisions Required (Vote on Step's outcome?)	• One • At Least One • All
Timeout (Days)	
Security (who can act on step): • Security Group • User Name • Standard Token • User Defined Token	
Include Notification (Yes/No)	
Notification Event	
Notification Recipient: • Username • Email Address • Security Group • Standard Token • User Defined Token	
Notification Message	

Validation Information*	Value
Existing Validation?	
New Validation?	
Validation Type: *(text field, auto-complete, drop down list, etc.)*	
Validation Definition (list of values or SQL)	

Information used when adding the Step Source to the Workflow Layout

Figure 6-4 Information used to create the Decision Step.

Enter the general information on the Decision step

Enter the following information in the Workflow Step window.

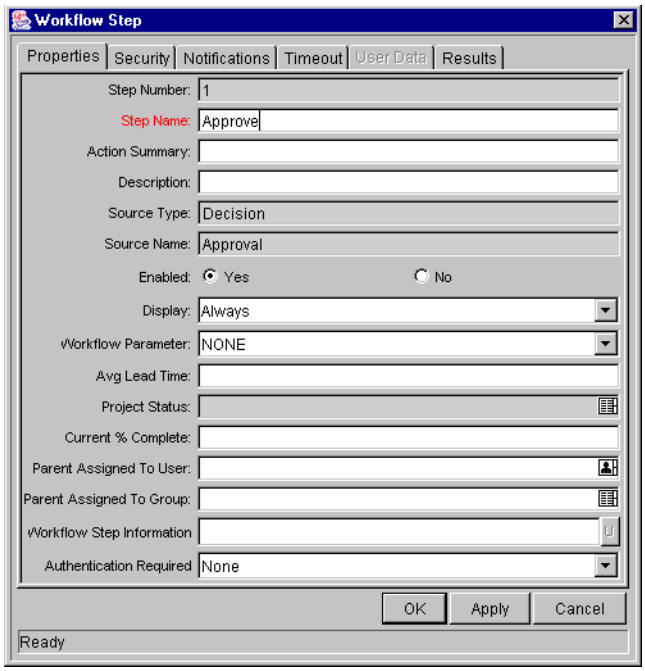


Table 6-4. Decision step worksheet to Workflow Step window

Field/Tab on Workflow Step Window	Description
Step Name	Name of the step that appears on the **Layout** tab.
ACTION SUMMARY	The text that appears on the action button in the Package status panel.
DESCRIPTION	A description of the step.
ENABLED	Whether or not the step is enabled.
DISPLAY	Whether or not to show the step on the Package status panel.
WORKFLOW PARAMETER	Used to save the results of a Workflow Step for later use in the Workflow processing.
AVG LEAD TIME	A user-specified metric for comparing actual performance to estimated goals. It does not affect any transactional logic.
PARENT ASSIGNED TO USER	If this field is not empty when the step becomes Eligible, the Assigned to User of the Package automatically changes to the user specified in the field.

Table 6-4. Decision step worksheet to Workflow Step window [continued]

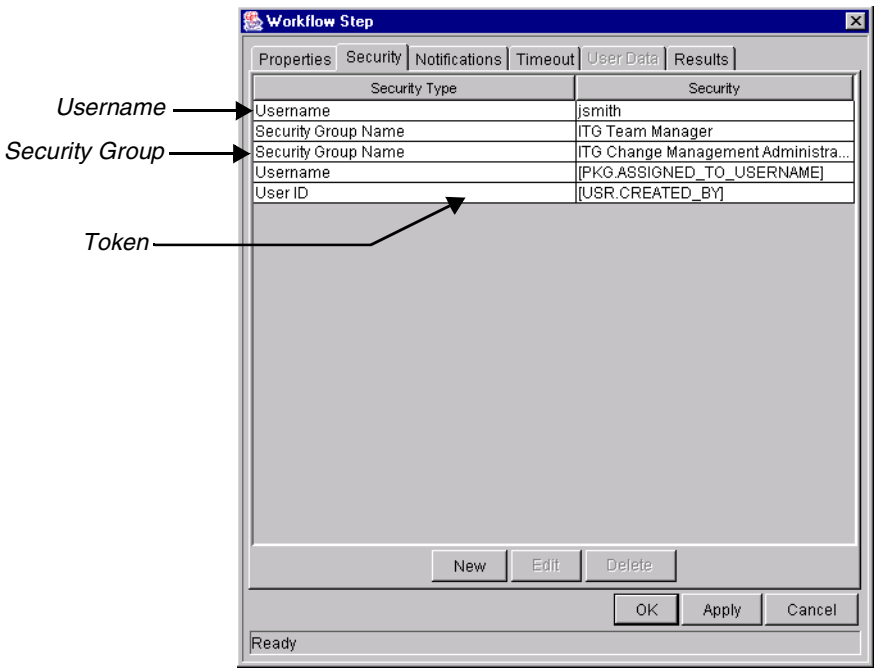
Field/Tab on Workflow Step Window	Description
PARENT ASSIGNED TO GROUP	If this field is not empty when the step becomes Eligible, the Assigned to Group of the Package automatically changes to the Security Group specified in the field.
WORKFLOW STEP INFORMATION	A text entry field in which any URL can be entered. This is where users can find documents, instructions or comments to aid them in processing the Workflow Step.
SECURITY TAB	See *"Setting Configuration Security"* on page 201 for configuration details.
Notifications Tab	Specify who will receive an email Notification when this step becomes eligible or has a specific result or error.
TIMEOUT TAB	Specify the Timeout value for this step. In the **Timeout** tab, select to use the Workflow Step Source timeout value or specify your own in the Specific Value section.

Specify the Security

"Integrating Participants into Your Deployment System" on page 187 provides information on setting up security for a deployment process. This includes such things as controlling who can create Packages and who can act on specific steps in the process. Security related directly to processing a Workflow Step is configured in the Workflow Step window.

Define who can act on the step by:

- Security Group
- Username
- Token (standard or user-defined)



Tip

Consider using Security Groups or dynamic access (Tokens) when defining the Workflow Step security. 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 an organizational reorganization), that list would need to be updated in many places on the Workflow. 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.

Configure Notifications for the Workflow Step

"Setting Up Communication Paths" on page 207 provides information on setting up Notifications for steps in a process. This includes such things as configuring the Notification's recipients and message.

For more details, see the following sections:

- *"Identify Participants and Security"* on page 56
- *"Setting Up Communication Paths"* on page 207

Specify the Timeout value for the step

Use the **Timeout** tab on the Workflow Step window to set a timeout for the Workflow Step. Timeout can be set according to either of the following options:

- **Use Workflow Step Source**—This is the default setting. The Workflow Step Source determines the Step's timeout.
- **Specific Value**—Enter a value for the Workflow Step's timeout according to:
    - **Constant** — Enter a numerical value and specify **Minutes**, **Hours**, **Days**, **Weeks**, or **Weekdays**.
    - **Token** — Enter a Token that resolves to a number and specify **Minutes**, **Hours**, **Days**, **Weeks**, or **Weekdays**.

Adding Execution Steps

To add an Execution step to your Workflow:

1. Drag the Step Source onto the **Layout** tab.
2. *Enter the general information on the Execution step*
3. *Specify the Security*
4. *Configure Notifications for the Workflow Step*
5. *Specify the Timeout value for the step*

Figure 6-5 shows the information used when creating an Execution Step.

Information used when adding the Step Source to the Workflow layout.

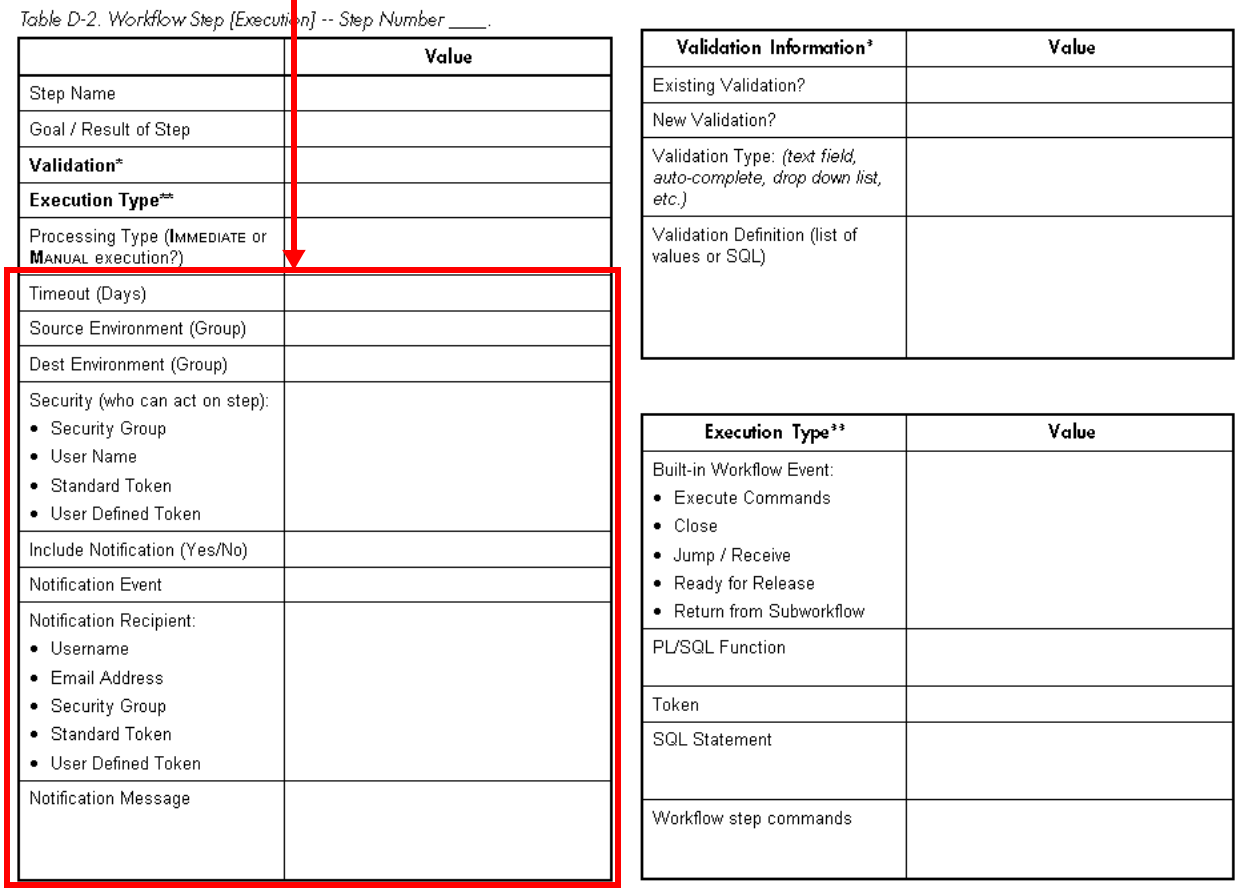


Table D-2. Workflow Step (Execution) -- Step Number ____.

	Value
Step Name	
Goal / Result of Step	
Validation*	
Execution Type**	
Processing Type (IMMEDIATE or MANUAL execution?)	
Timeout (Days)	
Source Environment (Group)	
Dest Environment (Group)	
Security (who can act on step): • Security Group • User Name • Standard Token • User Defined Token	
Include Notification (Yes/No)	
Notification Event	
Notification Recipient: • Username • Email Address • Security Group • Standard Token • User Defined Token	
Notification Message	

Validation Information*	Value
Existing Validation?	
New Validation?	
Validation Type: *(text field, auto-complete, drop down list, etc.)*	
Validation Definition (list of values or SQL)	

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 6-5 Information used to create the Execution Step.

Enter the general information on the Execution step

Enter the following information in the **Properties** tab in the Workflow Step window.

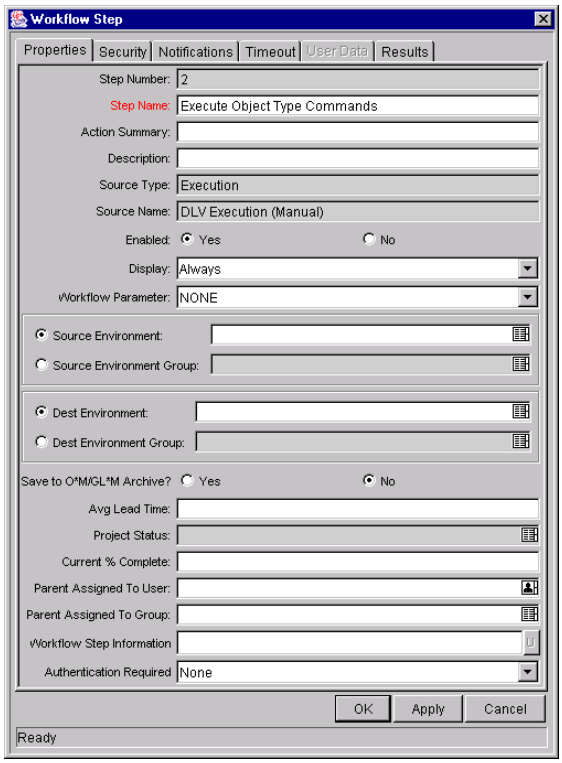


Table 6-5. Execution step worksheet to Workflow Step window mapping

Field on Workflow Step Window	Description
Step Name	Name of the step that appears on the **Layout** tab.
Action Summary	The text that appears on the action button in the Package status panel.
Description	A description of the step.
Enabled	Whether or not the step is enabled.
Display	Whether or not to show the step on the Package status panel.
Workflow Parameter	Used to save the results of a Workflow Step for later use in the Workflow processing.
Source Environment	Specifies the Source Environment where the software that is to be changed is located.

Table 6-5. Execution step worksheet to Workflow Step window mapping

Field on Workflow Step Window	Description
Source Environment Group	Specifies the Source Environment Group which contains the Environment from which the software change is obtained. The Source Environment Group can also be used in conjunction with the Environment Application Codes to provide a dynamic Source Environment selection.
Dest Environment	Specifies the Destination Environment to which the software change is deployed.
Dest Environment Group	Specifies the destination Environment Group. The destination consists of multiple Environments to which the software change is deployed.
Avg Lead Time	A user-specified metric for comparing actual performance to estimated goals. It does not affect any transactional logic.
Parent Assigned to User	If this field is not empty when the step becomes Eligible, the Assigned to User of the Package automatically changes to the user specified in the field.
Parent Assigned to Group	If this field is not empty when the step becomes Eligible, the Assigned to Group of the Package automatically changes to the Security Group specified in the field.
Workflow Step Information	A text entry field in which any URL can be entered. This is where users can find documents, instructions or comments to aid them in working the Workflow Step.

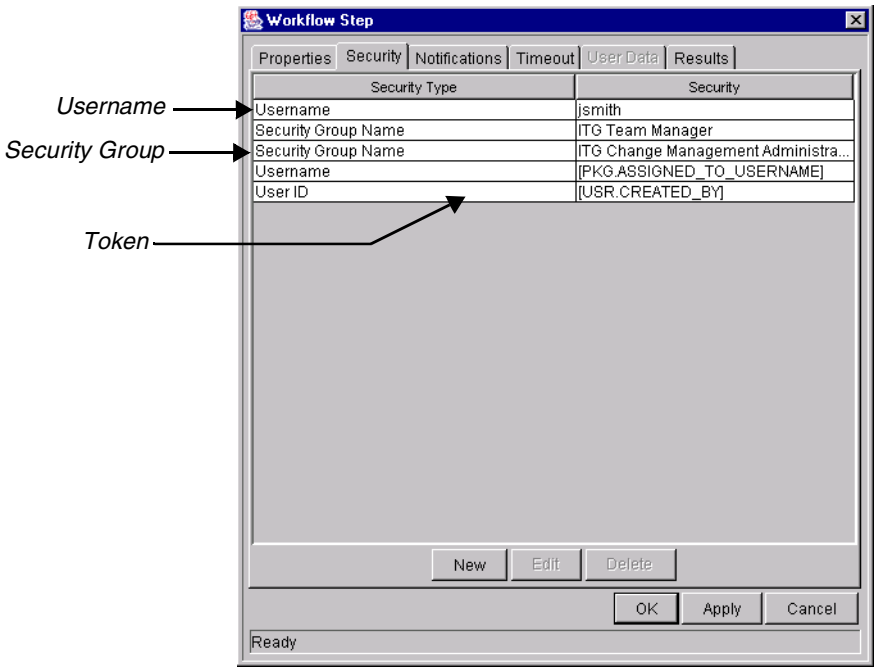
Specify the Security

"Integrating Participants into Your Deployment System" on page 187 provides information on setting up security for your deployment process. This includes such things as controlling who can create Packages and who can act on specific steps in the process. Security related directly to processing a Workflow Step is configured in the Workflow Step window.

Define who can act on the step by:

- Security Group

- Username
- Token (standard or user-defined)



Tip

Consider using Security Groups or dynamic access (Tokens) to define the Workflow Step security. 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 an organizational reorganization), that list would need to be updated in many places on the Workflow. 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.

Configure Notifications for the Workflow Step

"Setting Up Communication Paths" on page 207 provides information on setting up Notifications for steps in your process. This includes such things as configuring the Notification's recipients and message.

For more details, see the following sections:

- *"Identify Participants and Security"* on page 56
- *"Setting Up Communication Paths"* on page 207

Specify the Timeout value for the step

Use the **Timeout** tab on the Workflow Step window to set a timeout for the Workflow Step. Timeout can be set according to either of the following options:

- **Use Workflow Step Source** — This is the default setting. The Workflow Step Source determines the Step's timeout.
- **Specific Value** — You can enter a value for the Workflow Step's timeout according to:
    - **Constant** — Enter a numerical value and specify **Minutes**, **Hours**, **Days**, **Weeks**, or **Weekdays**.
    - **Token** — Enter a Token that resolves to a number and specify **Minutes**, **Hours**, **Days**, **Weeks**, or **Weekdays**.

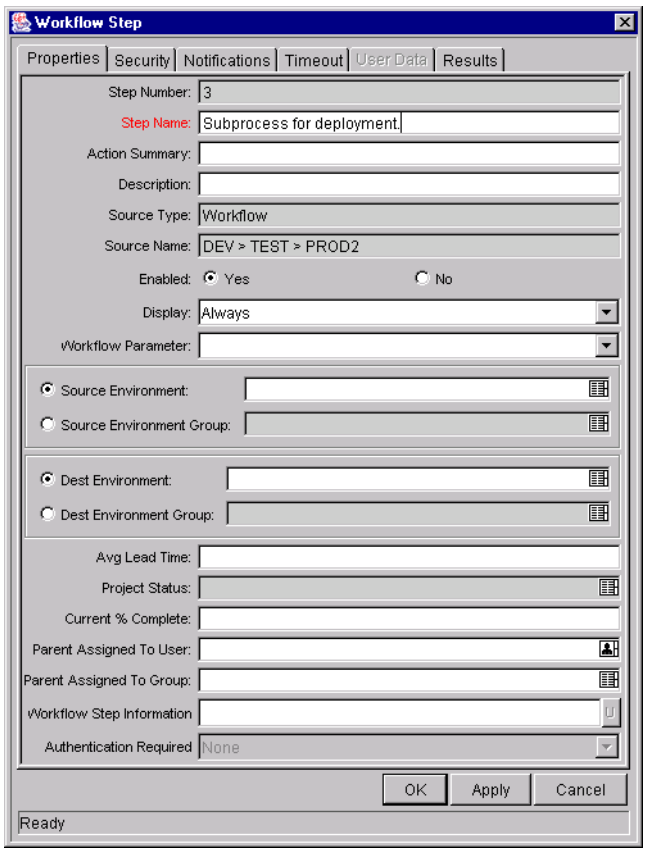
Adding a Subworkflow

A Subworkflow can be selected from the Workflow Step Sources window and dragged onto the **Layout** tab. When the Package, Request, or Release Distribution 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.

To add an enabled Subworkflow to another Workflow:

1. Select the desired Subworkflow and drag it to the **Layout** tab.

   The Workflow Step window opens. This window contains preconfigured information which is specific to the selected Workflow Step.



2. Configure this step as if configuring an execution or Decision Step.

3. Click **OK**.

For a detailed discussion of using Subworkflows, see *"Advanced Workflow Topics"* on page 255.

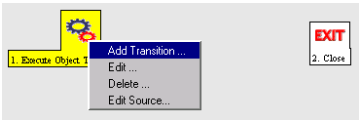
Adding Transitions Between Steps

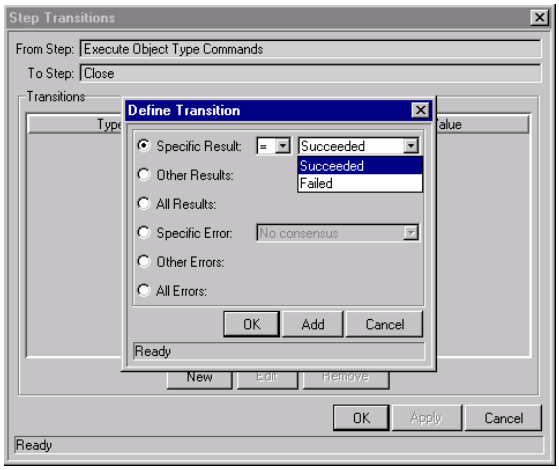
After adding the steps to the Workflow **Layout** tab, configure the transitions between them. Choose to transition between steps based on the following step results:

- **Specific Result** (based on the Validation configured in the Step Source)
- **Other Results**
- **All Results**

- **Specific Error**
- **Other Errors**
- **All Errors**

Select the proper transition between steps. The following section provides some example scenarios and transition configuration options:





Transition based on a specific result

Transitioning based on the result of a specific decision or execution allows the business process to branch based on anticipated results of a step in the Workflow.

To transition based on a specific Workflow Step result:

1. Add a transition between two steps.

   a. Right-click on a step and select **Add Transition.**

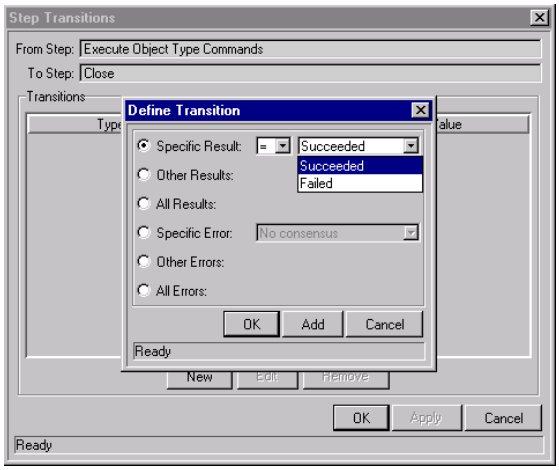
   b. Connect the arrow to the appropriate target step.

   The Define Transition window opens.

2. Select the Specific Result radio button.

3. Select the desired result from the drop down list.

The values in this list will vary depending on the Validation set in the Workflow Step Source for the transitioning step.



Transition based on a value in a field

It is possible to transition a Package based on the value in a particular field. This can be a general field in the Package definition (such as Priority, Assigned User, or Package Group) or a custom field specified in the Package Line (defined on the Object Type). For example, if the Package's Priority field is set to **Critical**, then you may want the Package to follow a different, more robust process. This is done by resolving a field Token in a Workflow Execution Step. The Workflow engine evaluates the field's value at a specific step and then can route the Package Line accordingly.

To transition based on the value in a field:

1. Add an immediate Execution Step source to the Workflow.

   It may be necessary to create a custom Workflow Step Source for this operation. The Step Source should be configured as shown in the following table:

Field in Execution Window	Value
Workflow Scope	**Packages**
EXECUTION TYPE	**Token**
PROCESSING TYPE	**Immediate**
Validation	Select or create a Validation that includes all of the possible values of the resolved Token. For example, if you plan on branching based on the Priority field, use the **[PKG.PRIORITY]** Token and the **DLV - Package Priority - Enabled** Validation. The Validation contains all possible values of the Token.
Execution	Enter the Token for the value that you would like to transition based on.
Enabled	**Yes**

2. Add transition between two steps.

   The Define Transition window opens.

3. Select the Specific Result radio button.

4. Select the desired result from the drop down list.

   The values in this list will vary depending on the Validation set in the Workflow Step Source for the transitioning step. For the previous Priority example, the possible values will be the values allowed in the Package's Priority field.

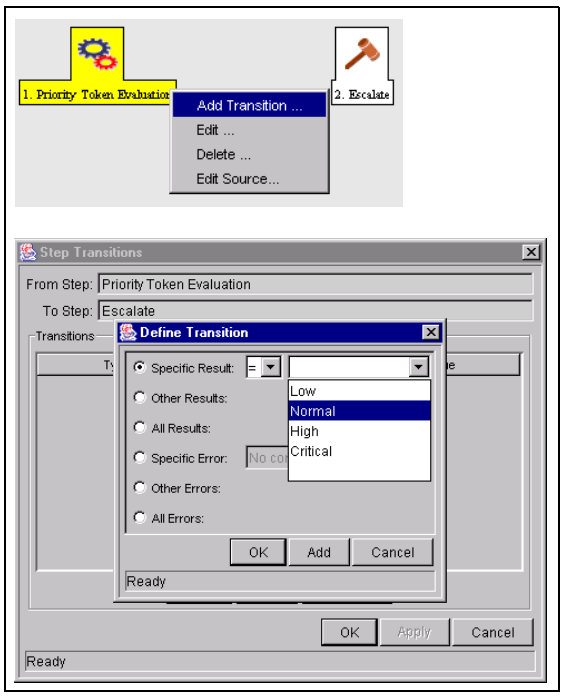


Figure 6-6 Example: Transitioning based on value in a field (Token)

Transition based on data in a table

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. For more information, see *"Execute a SQL statement and then transition based on the result"* on page 90.

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.

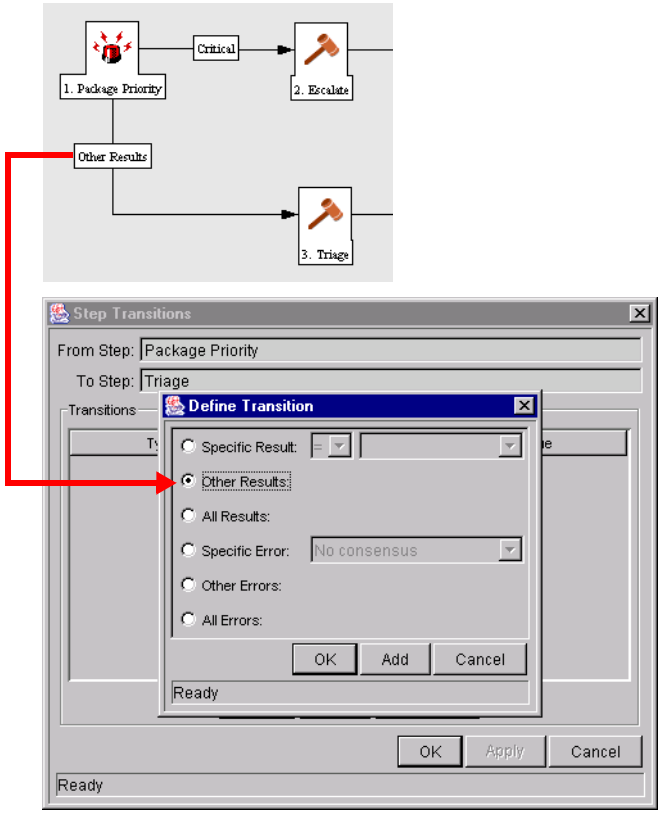
As with any Execution Step, configure this transition to be an immediate or manual step.

Transition based on all but one specific value

It is possible to transition based on all but one specified value. For example, you may want to transitional all "Critical" Packages one way and all other results another.

To transition based on all but one specific value:

1. Create a transition from a step based on a specific result.
2. Create another transition from the same step to another step and specify **Other Results**.



In the above example, only Packages with a “Critical” priority will follow the Escalate path. All other results are sent to Triage.

Tip

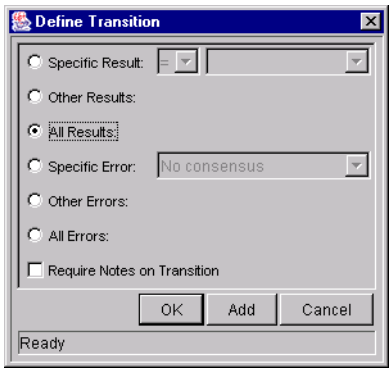
Use Other Results when multiple transitions are exiting a single step. Other Results will act as the transition if none of the other explicit transition conditions are satisfied.

Transition based on all results

It is possible to define a Package 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**. The Define Transitions window is shown below.



Tip

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**. The process can be brought together later using an AND step.

Transition based on error

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**.

To transition based on a specific Workflow Step error:

1. Add transition between two steps.

   The Define Transition window opens.

2. Select the Specific Error radio button.

3. Select the error from the drop down list.

   All values in this list are defined in *Table 6-6*.

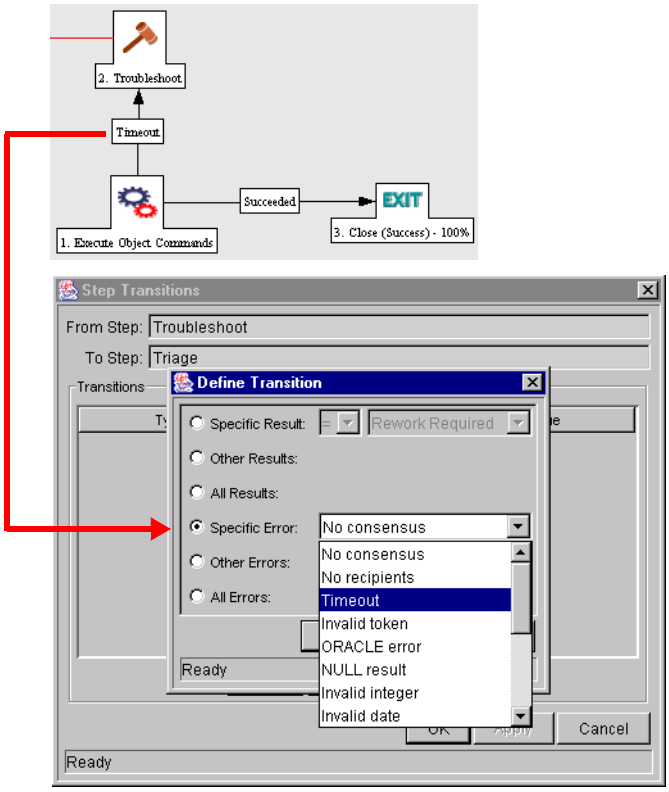


Table 6-6. Specific Error Step Transitions

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.
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.

Table 6-6. Specific Error Step Transitions

Transition Option	Meaning
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.

Transition back to the same step

It is possible 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.

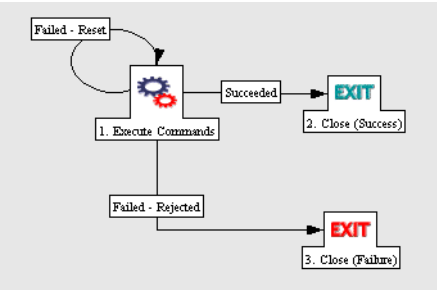
To transition back to the same Execution Step:

1. Create an Execution Step source to execute the Workflow or Object Type commands.
2. Create a Validation with the following Validation Values.
    a. SUCCEEDED
    b. FAILED
    c. FAILED - RESET
    d. FAILED - REJECTED
3. Add the step to the Workflow **Layout** tab.
4. Add transitions based on the following Specific Results:

a. SUCCEEDED

b. FAILED - RESET -- set the transition to return back into the same step.

c. FAILED - REJECTED



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 Package Line step and select **Failed - Retry**. The execution will re-run.

Note

Be careful when using an immediate Execution Step when the **FAILED** result is not feeding directly back into the Execution Step. This would result in a continual execution-failure loop.

Transition based on a previous Workflow Step result (parameters)

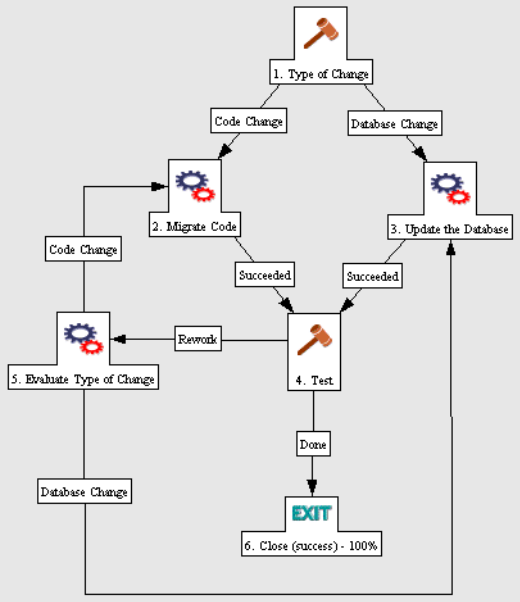
It is possible to use Workflow parameters to store the result of a Workflow Step. This value can then be used later to define a transition.

To create a transition based on a Workflow parameter:

1. Create a Workflow Parameter in the Workflow window.
2. Specify that Workflow Parameter in a Workflow Step on the **Layout** tab.
3. Create a Token Execution Step that will resolve the value in the Workflow parameter.

Example: Using a Workflow Parameter to Transition

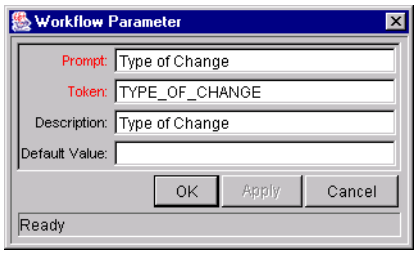
One step in the process requires the user to route the Package based on the type of change (code or database). The decision made at this step is then considered later in the process to correctly route rework of the specific type.



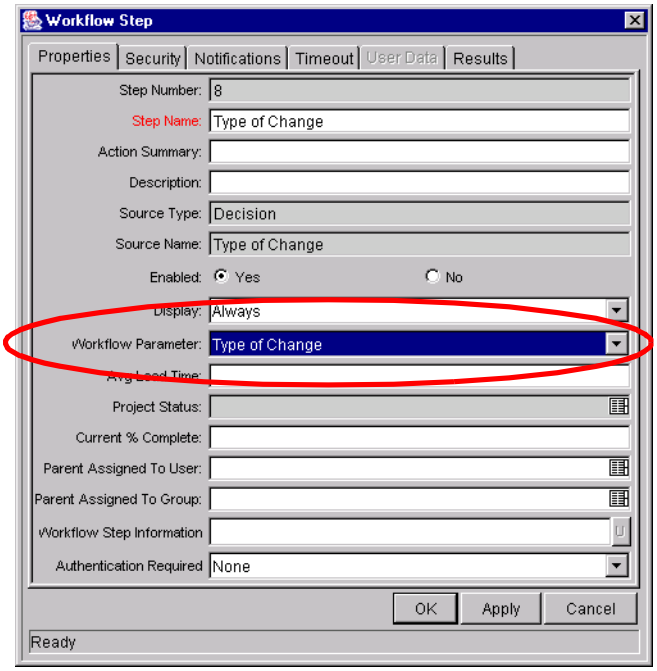
To enable this process, set the following in the Workflow:

1. Create a Workflow Parameter in the Workflow window.

   This is done by clicking **Add** in the **Workflow** tab on the Workflow window.

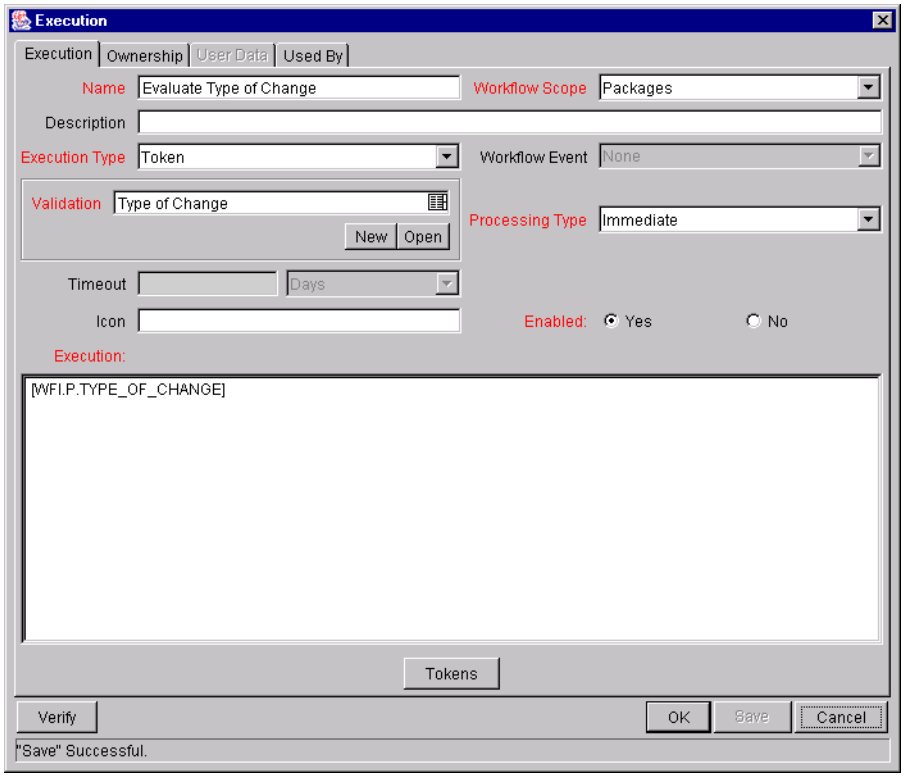


2. Select **Type of Change** for the Workflow Parameter in Type of Change Workflow Step on the **Layout** tab.

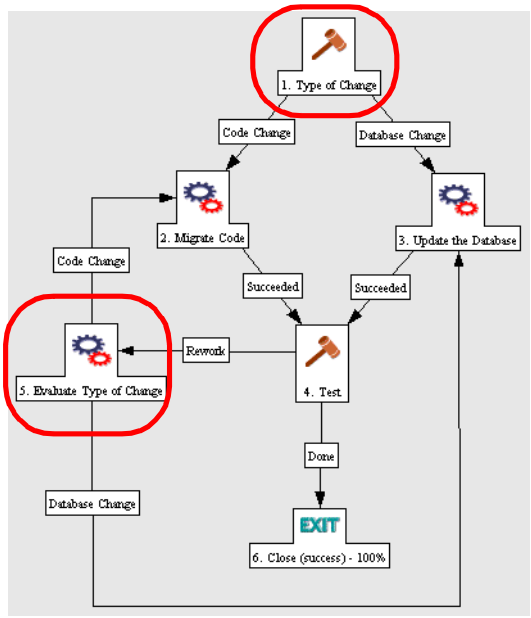


3. Create a Token Execution Step that will resolve the value in the Workflow parameter.

   Note that the Validation used in this step should contain the same values as the Validation specified in the Type of Change Decision Step.



4. Add the steps and transitions as shown below.



Transition to and from Subworkflows

There are special configuration requirements when transitioning to and from Subworkflows. For detailed instructions, see *"Using Subworkflows"* on page 255.

Transition to and from a Request Workflow

There are special configuration requirements when transitioning to and from a Request Workflow using Jump and Receive steps in the Workflows. For detailed instructions, see *"Using Subworkflows"* on page 255.

Chapter 7
Constructing the Object Type

This chapter provides an overview for how to configure the Object Types that will be used to process objects (Package Lines) through a deployment Workflow. This includes configuring Object Type fields and Commands.

This chapter discusses the following topics:

Creating an Object Type - Overview

Object Types are created and configured using the Workbench.

To create a new Object Type:

1. Click the **Change Mgmt** screen group on the Workbench and click the **Object Types** icon.

   The Object Type Workbench window opens.

2. Click **New Object Type**.

   The Object Type window opens.

3. Enter the Object Type general information.

   This includes the Object Type's Name, Description, Extension association, and Object Category.

4. Create fields that describe your Object.

This includes configuring the following:

- Field names
- Validations and component types (dictated by the Validation)
- Field Behaviors: whether fields are displayed, required, any defaulting behavior, etc.

For detailed instructions, see *"Creating Object Type Fields"* on page 125.

5. Configure the Fields' layout.

   For detailed instructions, see *"Modifying the Object Type Layout"* on page 138.

6. Create the Object Type's commands.

   For detailed instructions, see *"Creating Object Type Commands"* on page 144.

7. Set Ownership for the Object Type.

   This controls who can modify or delete the Object Type. For detailed instructions, see *Security Model Guide and Reference*.

It is often more efficient to use the **Copy** functionality to copy an existing Object Type and then edit the new copy. To reduce the amount of editing required choose an existing Object Type similar to the Object Type to be generated.

Only users with the appropriate security can create or edit Object Types. To edit Object Types, you must belong to a Security Group that has the Change Mgmt: Edit Object Types Access Grant. See the *"Setting Configuration Security"* on page 201 for more information.

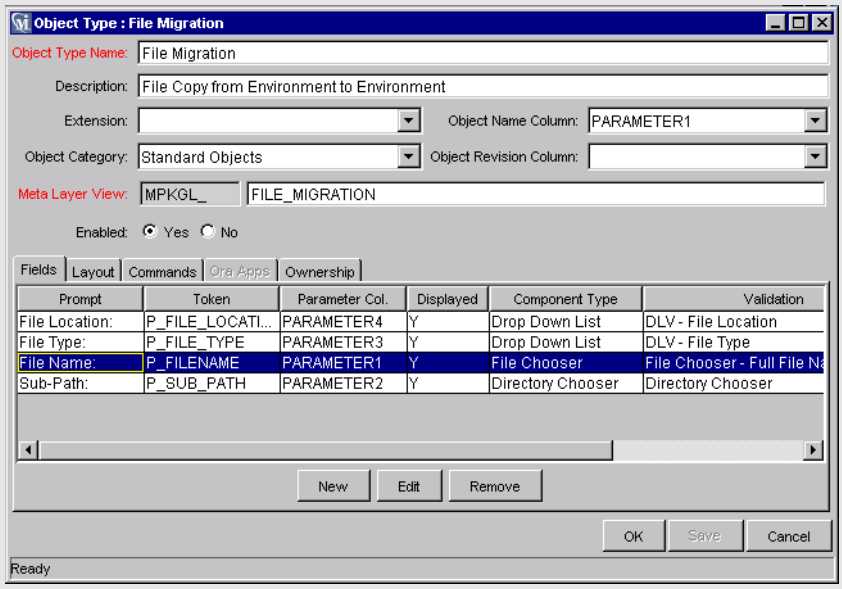
Creating Object Type Fields

Object Type fields define the information collected from the end users when a Package Line is generated. It is possible to configure fields prompts, Tokens, behaviors and Validations for each Object Type.

Object Type fields are critical for deployments. They are often used by the Workflow for routing (Validations). Tokens associated with the field are also often referenced in the Object Type commands, Workflow Step security, and Notification settings.

Example

To migrate a file, Mercury Change Management must know the file name. A field named “File Name” can be created to capture that information.



To configure an Object Type field:

1. Open the Object Type window.
2. Click **New**.

   The Field window opens.
3. Enter the general field information: Field Prompt, Token, and Description.
4. Select a Validation for the field.

If a Validation does not exist that meets the necessary requirements, one must be created. 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).

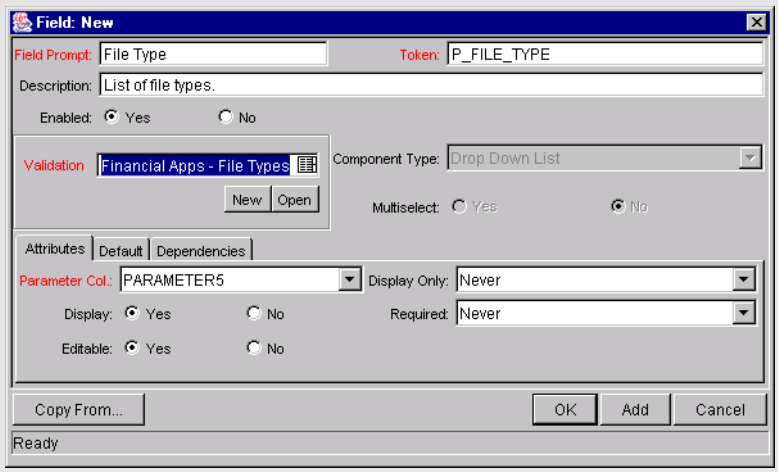
5. Configure the field's behavior.

   This consists of setting options in the field's **Attributes**, **Default** and **Dependencies** tabs. See *"Configuring Field Behavior"* on page 132. 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.

6. Enable the field.

Example

ACME requires a File Type field to describe the objects to be deployed. On their Object Type, they add a field with the following settings:



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

Determining the Field Type (Selecting a Validation)

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). The following sections provide some general information related to Validations.

- *"Available Field Types"* on page 127
- *"Selecting the Validation"* on page 129
- *"Building a Validation"* on page 129

For more detailed implementation instructions, see *"Validations"* on page 287.

Available Field Types

Fields located on the Object Type can be of the following types.

Table 7-1. Field types.

Field component	Description
Text Field, Text Area	Text fields and text areas are generic entry fields. Text fields are displayed on a single line, while text areas are displayed on multiple lines using a scroll bar if necessary. The values that are entered can be constrained. If an attempt is made to type non-conforming values into a text field or text area, the entries are ignored. For example, if the letter "A" is typed into a numeric field, the character does not appear. Text fields can also be configured to display the data according to a certain format. For example, you can configure a text field to accept and format a nine digit, hyphenated social security number or a ten digit telephone number.
Drop down list	Allows the user to select from a predefined set of values. The values in a drop down list can be specified in two ways: • In the Validated By field, by selecting **List** to enter specific values. • By selecting **SQL** to use a SQL statement to build the contents of the list.
Auto-complete list	Allows the user to select from a predefined set of values. The values in an auto-complete list can be specified in the following ways. In the Validate By field, select one of the following: • **List**: used to enter specific values. • **SQL**: uses a SQL statement to build the contents of the list. • **Command With Delimited Output**: uses a system command to produce a character-delimited text string and uses the results to define the list. • **Command With Fixed Width Output**: uses a system command to produce a text file and parses the result on the basis of the width of columns, as well as the headers.

Table 7-1. Field types.

Field component	Description
Radio Button (Yes/No)	Radio buttons are used for fields where there are two or more choices. Selecting on option disables the other associated options. For example, clicking **Yes** in a Yes/No radio button pair disables the **No** option. To select a choice, click the button to the left of the appropriate choice.
Date Field	Date fields can accept a variety of formats. The current date field Validations are separated into two categories: all systems, and systems using only the English language.
Web Address (URL)	The Web Address field is a generic text entry field in which any URL can be entered. When this field is used, a U button appears next to the field. If U is clicked, a Web page is opened using the field value as the Web address.
Directory Chooser	The Directory Chooser field can be used to select a valid directory from an Environment. Mercury Change Management connects to the first Source Environment on a Workflow and allows navigation through the directory structure and the selection a directory from the list. On every Object Type that a Directory Chooser is used, it is also necessary to have a field whose Token is 'P_FILE_LOCATION' and whose Validation is **DLV - File Location**. The possible values for this field are **Client** and **Server**. If **Client** is chosen, the Directory Chooser connects to the Client Base Path of the Source Environment. If **Server** is chosen, the Directory Chooser connects to the Server Base Path of the Source Environment.
File Chooser	A File Chooser field can be used to select a valid file from an Environment. Mercury ITG Center connects to the first Source Environment on a Workflow and provides the ability to view all files within a specific directory and select one from the list. On every Object Type that a File Chooser is chosen, it is necessary to have two other fields defined. The first is a field for the File Location for the directory chooser, described in the previous section. The second is a field whose Token is 'P_SUB_PATH'. This field is the directory from which the file is selected and is usually a Directory Chooser field.
Password	The Password Field component type creates a text field with an associated **C** button. Data is entered through a dialog that asks for the new password and a verification of the password. The text is then displayed in the field as *****.

Mercury Demand Management also support the following additional field component types:

- Table Component
- Budget
- Staffing Profile
- Resource Pool

See *"Validations"* on page 287 for details.

Selecting the Validation

Use the information gathered in *"Gathering Process Requirements and Specifications"* on page 35 to determine the appropriate Validation for the Object Type field. If a Validation does not exist that meets the necessary requirements (such as having the appropriate values), one will need to be created. For a complete list of Validations that are delivered with Mercury Change Management, see *"Validations"* on page 287.

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.

Building a Validation

If a Validation does not exist that meets the necessary requirements (such as having the appropriate values), one may be created. Click **New** in the Field window to open the Validation window. Define your Validation using the instructions in *"Validations"* on page 287.

This section provides some guidance for when to use specific types of Validations. *Table 7-2* highlights when to use certain component types.

Table 7-2. When to use certain field component types.

Component Type	When to use:
Auto-complete list Drop down list	Use when presenting a list of options to the user. For example: • List of all users • List of all users in a specific Security Group (include on a Package Line to specify who should review a change) • Desired actions (copy only; copy and run commands; copy, run commands and delete; etc.) • List of information located in another (non-Mercury ITG) system. (example: list of managers stored in PeopleSoft)
Directory chooser File chooser	Used to specify the location of objects to be deployed.
Text field Text area Date field	Used to capture related information required for processing.
Radio button	Used when only two options exist. (Yes/No)

Auto-Complete Versus Drop Down List

Auto-completes and drop down lists are often applied in similar situations. They both present a predefined / limited list of choices to the user, but both have unique features which could be more appropriate for a given situation. Consider the following comparison chart when selecting to use a drop-down or auto-complete list.

Table 7-3. Auto-complete versus drop down comparison chart.

Action	Auto-complete	Drop-down
Can contain a static list of choices	Yes	Yes
Can contain choices derived from a database query	Yes	Yes
Can contain choices from system executions (commands)	Yes	

Table 7-3. Auto-complete versus drop down comparison chart.

Action	Auto-complete	Drop-down
Can display multiple columns of information	Yes	
Allows users to select multiple values from the list	Yes	
List is determined at the time of page/screen load		Yes
List is determined when the field is selected. this is useful when making the values in the list dependent on other parameters in the screen. For example, listing only users in a specified Security Group.	Yes	
Allows for partial value returns (for example, type “A” and and view only the choices beginning with “A.”)	Yes	

Usability should also be considered. Drop down lists become less efficient when the selection list gets large. Consider using auto-completes in these situations.

Tips for Configuring Validations

Consider the following tips when creating a Validation for your Object Type:

- Be careful when creating Validations (drop down lists and auto-complete lists) that are validated by lists. Each time the set of values changes, the Validation must be updated. Consider, instead, validating using a SQL query or PL/SQL function. For example, to create an auto-complete field that lists all users in a specific department, validate the list by SQL.

```
SELECT U.USER_ID, U.USERNAME, U.FIRST_NAME, U.LAST_NAME
FROM KNTA_USERS U, KNTA_SECURITY_GROUPS SG,
   KNTA_USER_SECURITY US
WHERE SG.SECURITY_GROUP_ID = US.SECURITY_GROUP_ID AND
   US.USER_ID = U.USER_ID
AND SG.SECURITY_GROUP_NAME = 'Support Team'
and UPPER(u.username) like UPPER('?%')
and (u.username like upper(substr('?',1,1)) || '%'
  or   u.username like lower(substr('?',1,1)) || '%')
order by 2
```

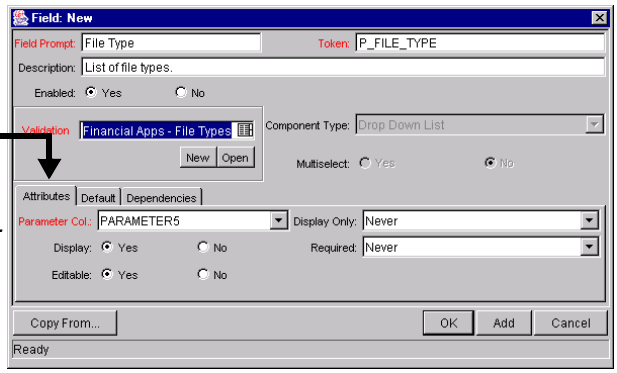
In the above example, when a new user account is created and included in the "Support Team" Security Group, that user will automatically be included in the auto-complete list.

- Reuse SQL and PL/SQL from existing Validations. Review the seeded Validations (see *"System Validations"* on page 359) to see if there are other similar Validations in the system. If there are, copy the Validation and modify the Validated By specifications to meet the necessary requirements.
- It is often possible to use the same Validations for Workflow Step Sources as for the field Validations.

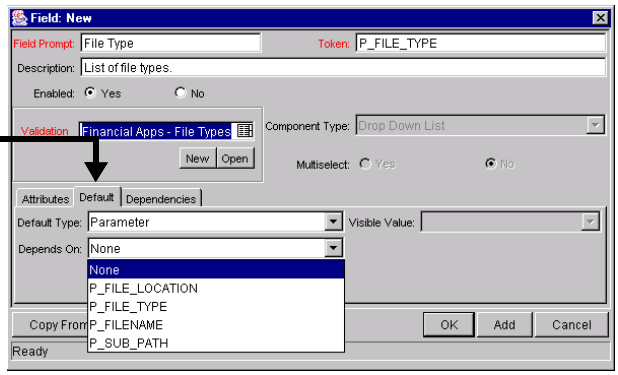
Configuring Field Behavior

It is possible to 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**, **Default**, and **Dependencies**.

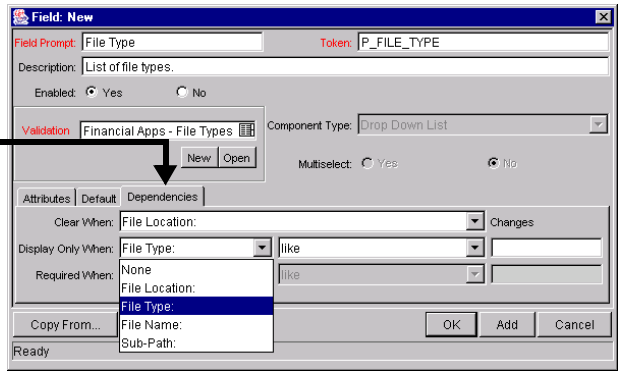
Attributes tab: *used to set basic display, edit and requirement field properties.*



Default tab: *used to set the value in the field.*



Dependencies tab: *used to set clearing, display and requirement field properties based on values in other Object Type fields.*



From the Field window, configure whether the field:

- Is displayed (for example: you may 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

Table 7-4 defines the behavior-related settings in the Field window.

Table 7-4. Attributes Tab - Fields window

Field	Description
Parameter Col.	Determines 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. 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.
Display Only	Determines 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. Display Only: **Always** means that the field is not editable. Display Only: **Never** means that the field is always editable.
Display	Determines if this field is visible in the Package Line region of the Package window.
Required	Determines 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.
Updateable	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.

Table 7-5. Default tab - Fields window

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.

Table 7-5. Default tab - Fields window

Field	Description
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.

Table 7-6. Dependencies tab - Fields window

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.

Since field behavior is often dependent on other fields in the Object Type, the other Object Type fields will often have to be created before configuring a field's behavior.

Configuring Field Dependencies

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

A field can be configured to:

- Clear when another field changes.

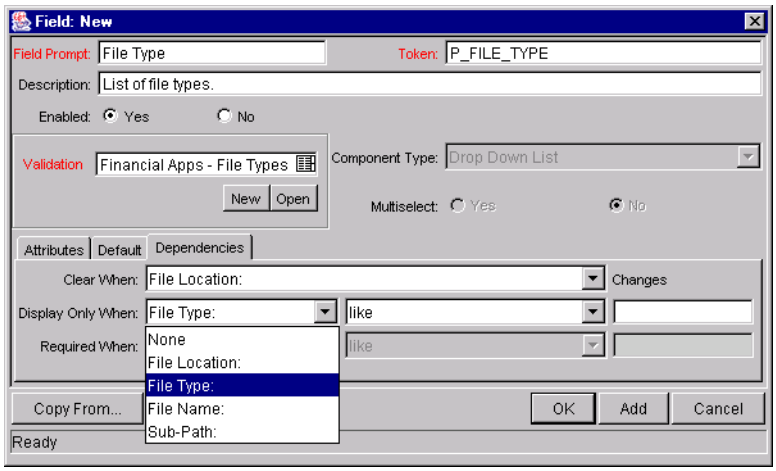
- Become read only when another field meets a logical condition defined in *Table 7-7*.
- Become required when another field meets a logical condition defined in *Table 7-7*.

Table 7-7. Field Dependency logical qualifiers

Logical qualifier	Definition
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. In the Field window, click the **Dependencies** tab.



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

   a. 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.

   b. 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.

   c. 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.

3. Click **OK**.

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

Using Commands to Change Field Values

Commands can also be used to control certain behavior of Object Type fields. At specific points (Workflow Execution 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 on the same Object Type.

The ksc_store Special Command can perform this function. For information on using this and other commands, see *Commands and Tokens 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

Modifying the Object Type Layout

The Object Type layout can be modified by:

- *Changing a Column Width*
- *Moving Fields*

Changing a Column Width

To change the column width of an Object Type field:

1. Open the Object Type window.
2. Click the **Layout** tab.
3. Select the field.
4. 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.

Moving Fields

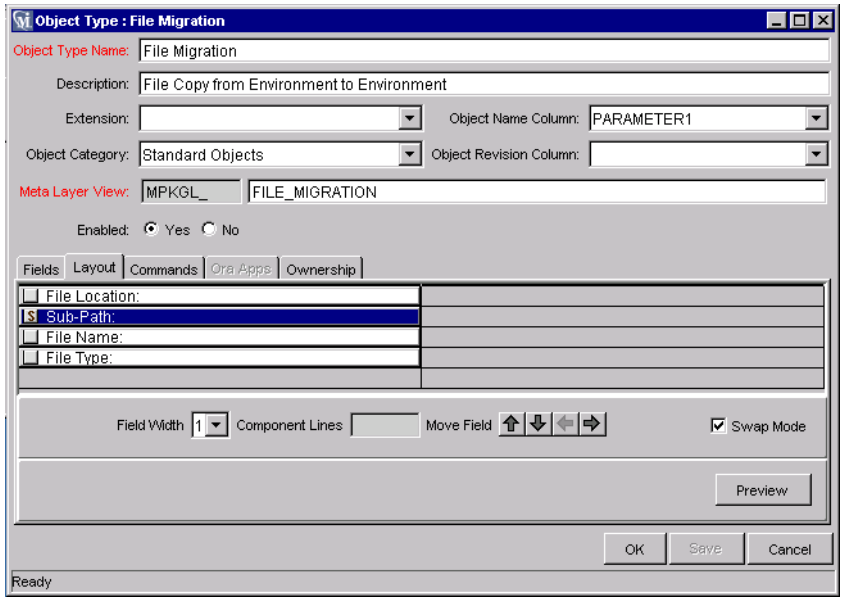
To move an Object Type field or a set of fields:

1. Open the Object Type window.
2. Click the **Layout** tab.
3. Select the field(s).

   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).
4. Use the arrow buttons to move the fields to the desired location in the layout builder.

Note

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.



5. 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.

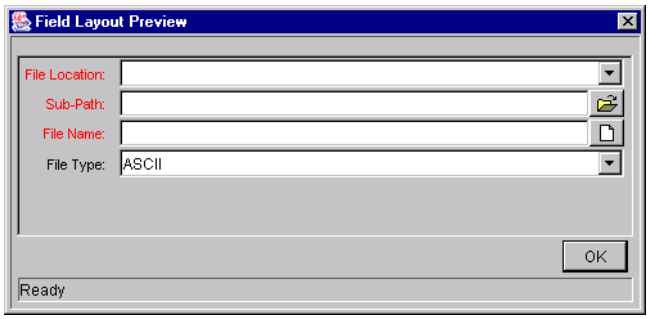
   b. Double-click the second field that will switch positions with the first.

      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.

6. 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:

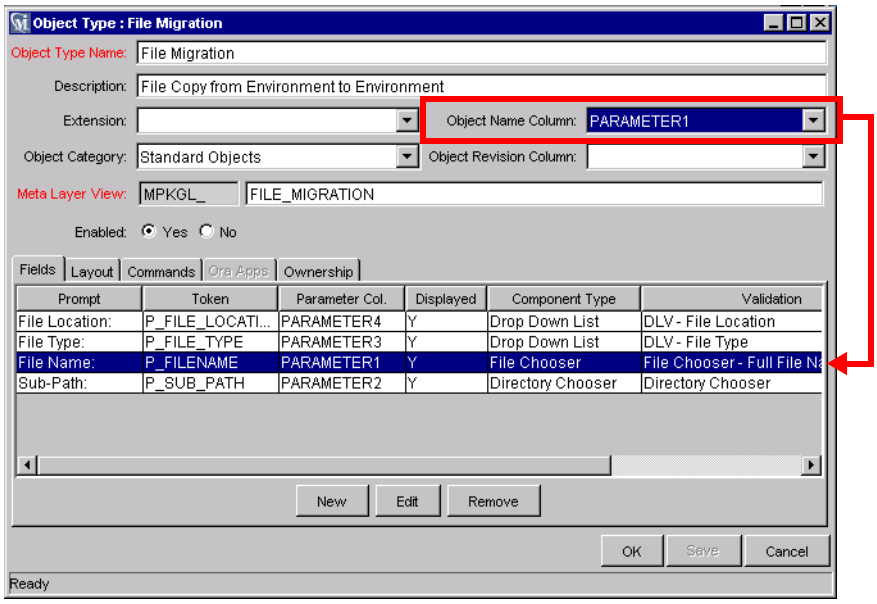
   - If all the fields have a width of one column, all displayed columns will automatically span the entire available area when a Package Line of the given Object Type is viewed or generated.
   - Any rows with no fields are ignored. They do not show up as a blank line.
   - Any non-displayed fields do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



Setting the Object Name

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 the Object Revision

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.

Copying Object Type Fields

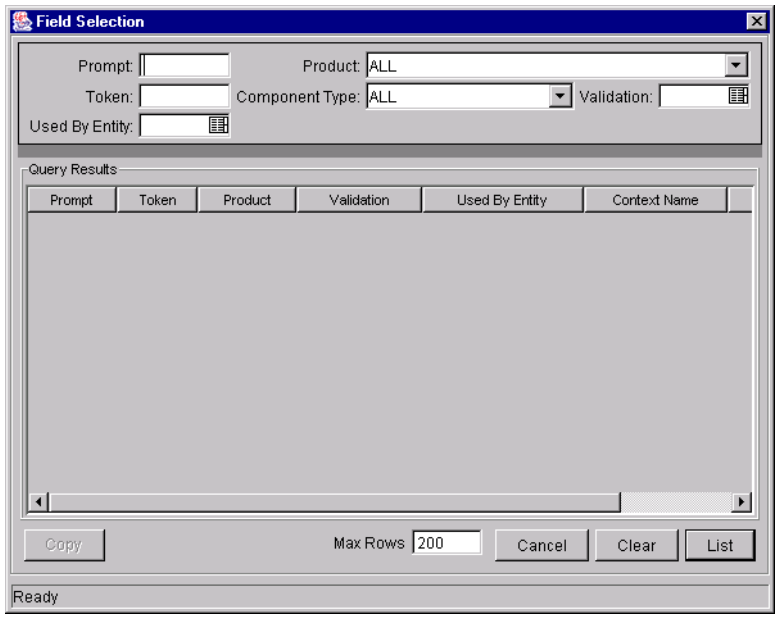
Use the **Copy From** functionality to streamline the process of adding fields to an Object Type by copying the definition of existing fields (from other Object Types).

To copy an Object Type field:

1. Open the Object Type.
2. Click **New** in the **Fields** tab.

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

   The Field Selection window opens.



4. Query for the field that will 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 ITG fields, limit the list of fields by one or more of the query criteria.

5. Select the desired field, and click **Copy**.

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

6. Make any necessary modifications.

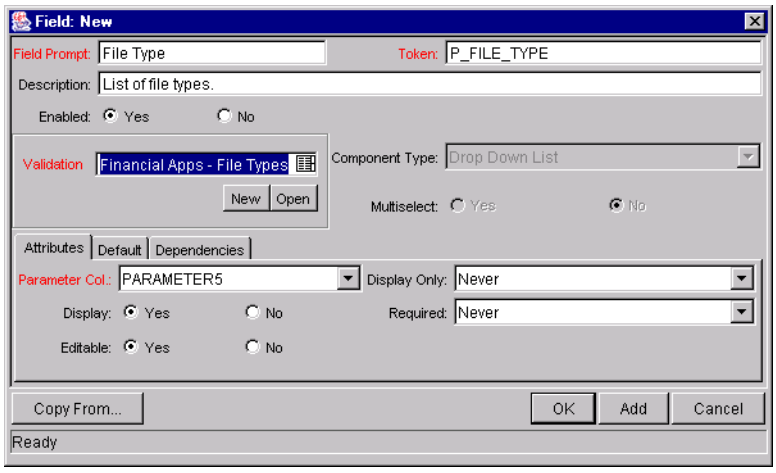
7. Click **OK**.

Editing Object Type Fields

To edit an existing field on an Object Type:

1. Open the Object Type.

2. Either double-click on the field in the **Fields** tab or select the field and click **Edit.**



3. Make the desired changes in the header region, **Attributes** tab, **Defaults** tab, and **Dependencies** tab.

4. Click **Apply** to apply the changes to the **Fields** tab without closing the field window, or click **OK** to apply the changes and close the Field window.

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 Field Prompt and Description should not affect the behavior of existing Package Lines.

Removing Fields

To remove a field permanently from an Object Type:

1. Open the Object Type.
2. Click the field in the **Fields** tab.
3. Click **Remove.**
4. Click **OK** or **Save** to save this change to the database.

This deletes the field from the list of 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.

Creating Object Type Commands

The following sections provide instructions and examples for configuring your Object Type commands.

- *Object Type Commands Overview*

- *Special Commands*
- *Command Steps*
- *Command Conditions*
- *Example Object Type Command Uses*

For additional examples of using Commands, see *Commands and Tokens Guide and Reference*.

Object Type Commands Overview

Object Type commands define the heart of the execution layer within a deployment system. Commands specify precisely which steps must be executed at a specific Workflow Step. This can involve such activities as migrating a file, executing a script, performing some data analysis, or compiling code.

Commands Interface

Commands are accessible through the **Commands** tab of the Object Type screen and consist of command information and command steps. Summaries of both parts of each command associated with the Object Type are visible in the **Commands** tab.

Command steps are the shell script commands that make Object Types function. Double-click the Command Step to open the Edit Command window. The Edit Command window displays the shell script code in the Steps window, as shown in *Figure 7-1*.

Double click the Command to open the Edit Command window.

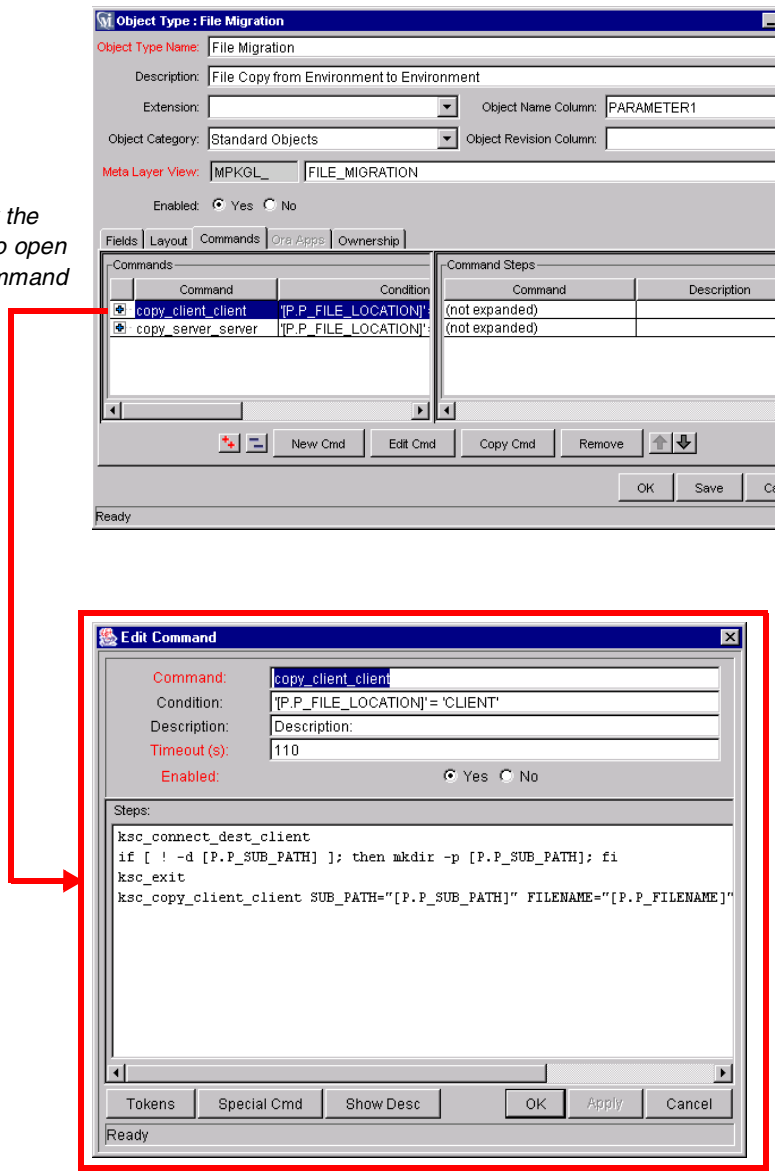


Figure 7-1 Object Type Commands Tab and Edit Command Window

To generate a new command, click **New Cmd** in the **Commands** tab. This opens the New Command window shown in *Figure 7-2*. *Table 7-8* shows the fields included in this window.

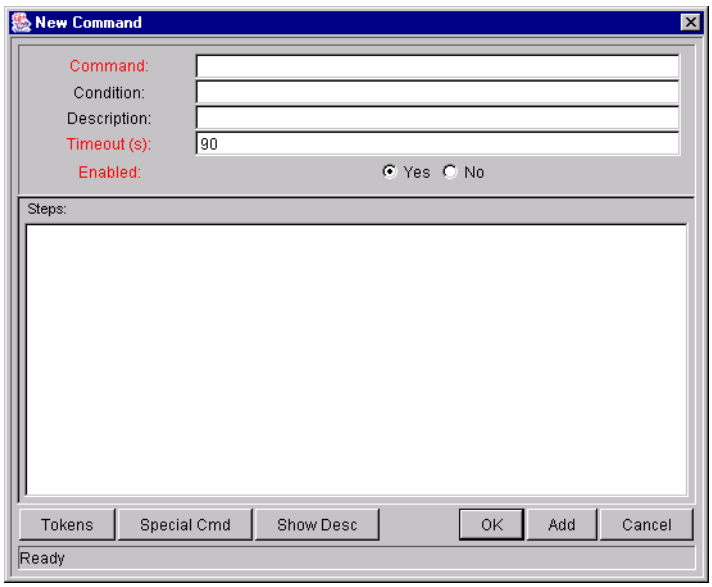


Figure 7-2 New Command Window

Table 7-8. New Command Window Fields

Field	Description
Command	A simple name for the command.
Condition	A condition that determines whether the command steps for the command are executed or not. (See *"Command Conditions"* on page 150 below 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?	Determines whether the command is enabled for execution.

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

One additional level of flexibility is introduced when commands must be executed in certain cases. This is powered by the condition field of the object commands and is discussed in *"Command Conditions"* on page 150.

Object Type Commands and Workflow

Object Type Commands are tightly integrated with the Mercury ITG 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/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.
- The Object Type can be configured to run only certain commands at a particular step. To do this, specify the *"Command Conditions"* on page 150.
- 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.

Special Commands

Object Types, Request Types, Report Types, Workflows and Validations all use commands to access Mercury Change Management's execution layer. In order to simplify the use of command executions, Change Management contains a predefined set of Special Commands. Users can also create their own Special Commands.

Special Commands are commands with variable parameters and are used in Object Types, Request Types, Report Types, Workflows, Workflow Steps, and Validation command steps. These command steps perform a variety of functions, such as copying files between Environments and establishing connections to Environments for remote command execution. The Mercury ITG Center uses two types of Special Commands:

- System Special Commands - These commands are shipped with the Mercury IT Governance Center. System Special Commands are read-only and have the naming convention ksc_command_name. System Special Commands always begin with ksc_.
- User Defined Special Commands - These commands are user-defined and have the naming convention sc_command_name. User-defined Special Commands must begin with sc_.

Special Commands act as sub-programs that can be reused where ever needed. It it often more convenient to create a Special Command for a program that will be used in multiple places rather than placing the individual commands into every Object Type.

Command Steps

Command steps represent the actual directives that Mercury Change Management specifies to an Environment's host as it tries to execute the commands for that instance of an object. *Table 7-9* describes the fields in the Command Steps region of the New/Edit Commands dialog.

Table 7-9. Command Steps

Field	Description
Steps	Defines the command-line directive or Special Command to be issued.
Description	Describes each of the command steps.

A command step can be an actual command-line directive that is sent to the target machine or can be one of many Special Commands.

The Execution Engine will execute the commands and command steps in the order they are displayed in the **Commands** tab. To change the order of the commands or the command steps, in the **Commands** tab, select the given command or command step and use the arrow buttons to move the selected item.

Command Conditions

In many situations, it may 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 an object 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 7-10. 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.

The single quotes are necessary for strings.

Example Object Type Command Uses

This section provides a number of operations that you can execute using Object Type commands. This section will use a combination of UNIX commands, third party commands, and Special Commands.

- Commands for connecting to machines
    - Connect to the destination Environment and run system commands
    - Connect to an alternate Environment and run command (Environment override)
- Commands for manipulating data. (Change Management fields and other info stored in files or database)
    - Set a value in a Package Line
    - Create, run and delete a script
    - Extract information from a file (version number)
- Commands for running operating system-specific commands. (NT and Unix)
    - Starting a server
    - Stopping a server
- Commands for running program-specific commands
    - Checking files in and out of a Version control system
- Commands for copying files

Chapter 8

Defining your Environments

This chapter provides an overview for defining the Environments that will be used in a deployment system. This activity requires some knowledge of the machines, filesystem, and databases that are serving as deployment sources and destinations. Environment definitions include system account information, passwords, and supported communication protocols for those machines.

This chapter covers the following topics:

Environment Requirements

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 logon using any existing username and password. However, it is recommended that a new user (such as “Mercury ITG”) 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 on the Mercury ITG 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 ITG Server home directory. (Any Windows NT computer that Mercury Change Management will access should have been configured as directed in *Installation Guide*.)

Defining Environments

To define a new Environment:

1. Open the Environment Workbench.

   a. Click **Environments** in the shortcut bar.

   b. Click the **Environments** icon.

2. Click **New Environment** on the Environment Workbench Window, or select **File > New > Environment**.

   The Environment window opens.

3. In the top portion of the window, enter the following information:

   - In the Description field, enter a description of the Environment.
   - In the Location field, specify the physical location of the server.
   - In the Enabled radio button field, select **Yes** or **No** to indicate whether the Environment is available for use in a Workflow.

4. Click the **Host** tab and enter the following information related to the Server and Client as described 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.

Fields	Description
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 target 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 target machine.

5. Enter the following information related to the Database.

   From the Server Type field, select the type of database server.

   a. If **Oracle Server** is selected:

      i. In the Host Name field, enter the name of the database.

      ii. In the Connect String field, specify the database connection string to connect to the desired database.

      iii. In the Username field, enter the username of the database schema.

      iv. In the Password field, enter the corresponding password.

      v. In the Oracle SID field, enter the Oracle SID of the database instance.

      vi. Enter the Port Number.

      vii. Enter the DB Link.

      viii. In the DB Version field, enter the database version number.

ix. If you are configuring an Oracle 9i RAC database, in the JDBC URL field, enter the valid RAC JDBC_URL, found in the server.conf file.

b. If **SQL Server** is selected:

i. In the Server Name field, enter the name of the server.

ii. In the DB Name field, enter the name of the database.

iii. In the User Login field, enter the username of the database schema.

iv. In the Password field, enter the corresponding password.

v. Enter the Port Number.

vi. In the DB Version field, enter the database version number.

6. Select the **Applications** tab and click the **New App, Copy App**, or **Copy Apps From** button to select/generate an App Code.

   Enter the relevant information, if any, for the App Codes to be generated. Only enter data for each Application that should override the same data at the Environment Host level. For more information, see *"Using App Codes with Your Environment"* on page 160.

7. Click the **Ownership** tab to specify the users that can edit, copy and delete the Environment.

8. Click the **User Access** tab to specify the users that can use this Environment in Workflow Steps and Environment Groups.

9. Enter any necessary information in the **User Data** tab.

10. If enabled, move to the **Extension Data** tab.

    Select the desired **Extension** tab at the bottom of the window and enter the information for correctly defining that Environment.

For detailed information regarding one or more of the purchased Mercury Change Management Extensions, see the Mercury Web Site at http://www.mercury.com.

11. Click **Save** to register all the Environment information entered so far.

12. Click **Check** to run the Environment checking functionality for this new Environment.

The Enable Server, Enable Client and Enable Database buttons should generally remain checked, and the information below the appropriate headings should be entered. These check boxes can be used as flags in conjunction with Object Type commands to further define multi-tiered Environments.

Copying Environments

To copy an Environment:

1. Open the Environment Workbench by clicking **Environments** in the shortcut bar and clicking the **Environments** icon.
2. Enter the search criteria required to select the Environments in the **Query** tab of the Environment Workbench and click **List**. The **Results** tab opens, displaying the results of the search.
3. Select the Environment to be copied in the **Results** tab.
4. Click **Copy**.

   The Copy window appears.
5. Enter the new Environment name as well as any additional information and click **OK**.

This generates the new Environment. The new Environment can then be opened and the desired changes can be made.

You must be a member of one of the Environment's Ownership Groups in order to copy an Environment. See *"Setting Configuration Security"* on page 201 for details.

Selecting the Environment's Connection Protocol

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 system 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

Selecting the Environment's Transfer Protocol

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

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 system 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.

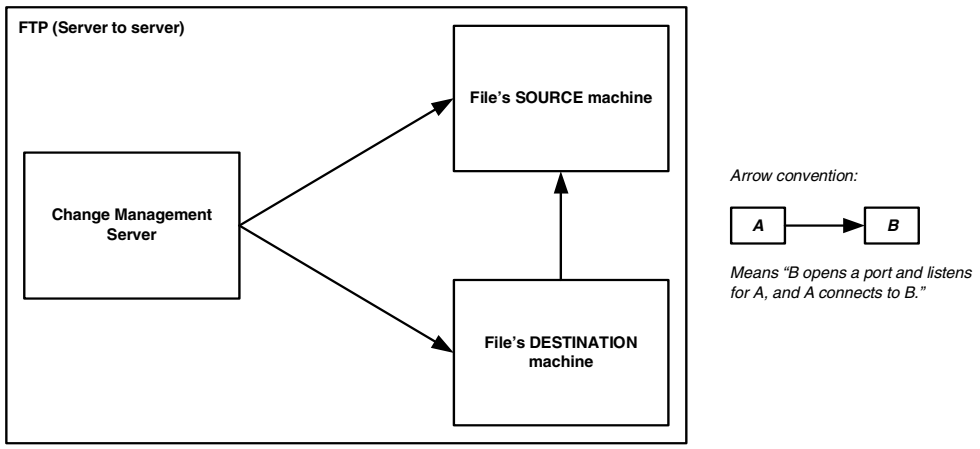
No additional product configuration is required to enable one FTP mode over another. Administrators do, however, need to consider their FTP server configuration (particularly as they relate to security and firewall settings) when selecting an FTP protocol for transferring data.

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:

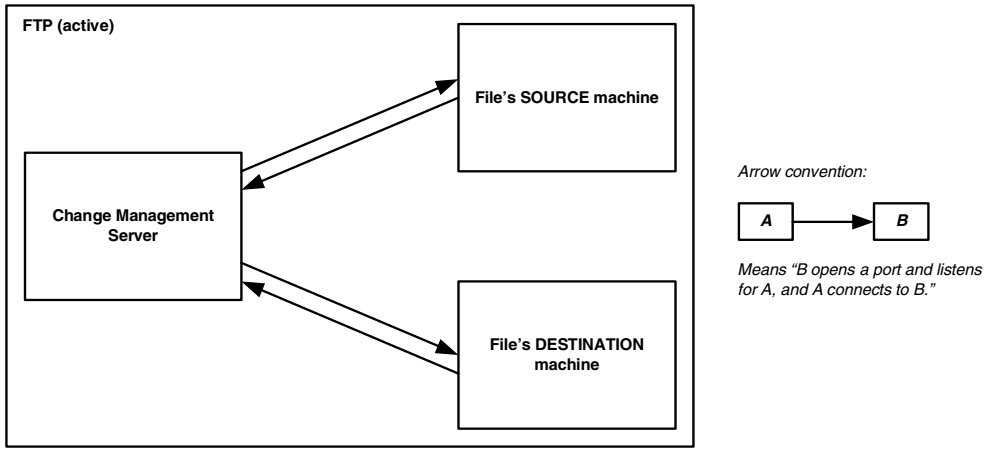
FTP:

- 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



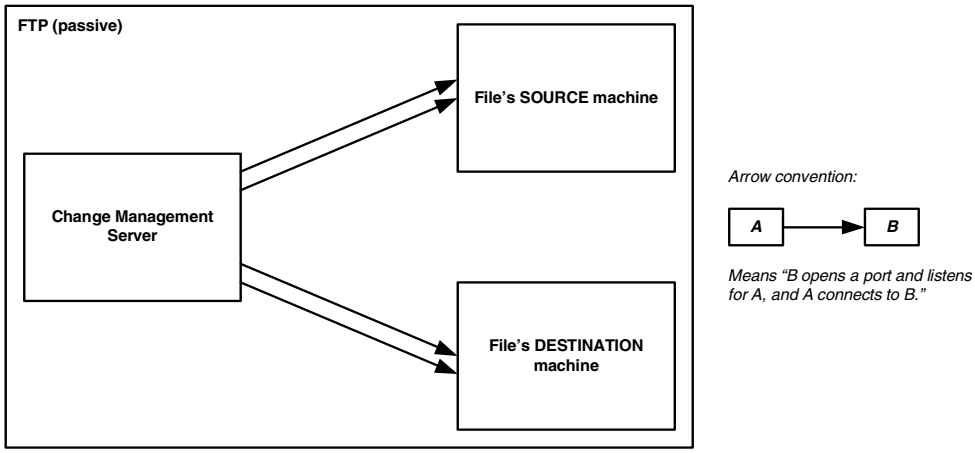
FTP (active):

- PORT command must be enabled on both the source and destination Environments (allows outgoing back to the requestor)



FTP (passive):

- PASV must be enabled on both the source and destination Environments. In this configuration, the Change Management server sends a command to the source or destination instructing that Environment to open a port. The Change Management server then connects to that port.



Using App Codes with Your Environment

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 panel.

When adding a line to a Package there is the option of choosing the App 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.

Note

Environment User Data fields will be inherited by each App Code and will appear in the App Codes' **User Data** tab. App Code User Data fields behave like other App Code fields (such as host name and base path), in that blank field values indicate that the App Code has the same value as its parent Environment.Therefore, required Environment User Data fields are not also required at the App Code level.

Each Environment can contain its own set of applications. The **Applications** tab of the Environment window is shown in *Figure 8-1*.

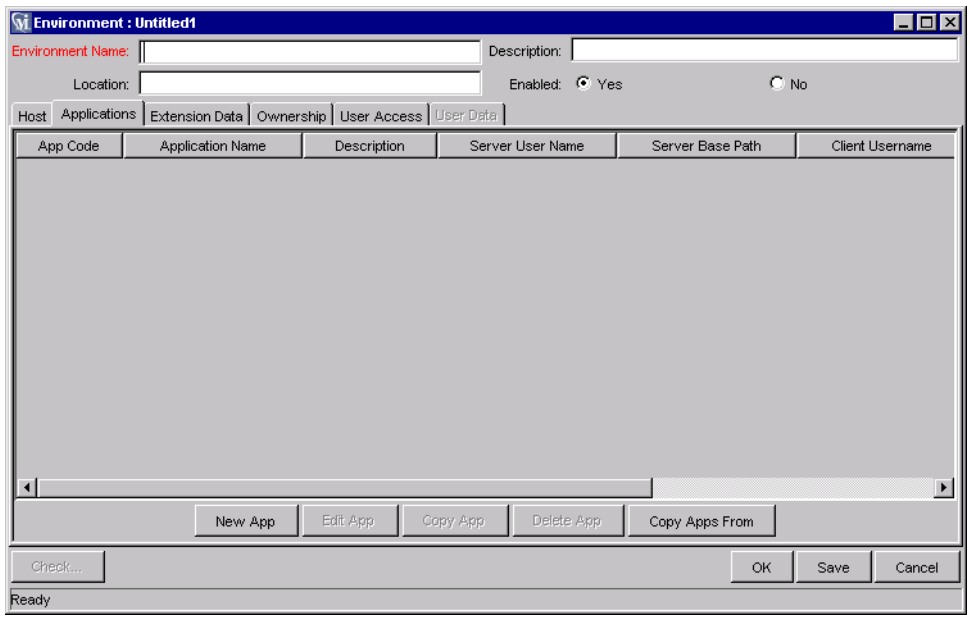


Figure 8-1 Environment Window - Applications Tab

The **Applications** tab consists of the various fields and buttons shown in the *Table 8-1*. Application fields do not always have to be populated. Click **New App** to open the New Application Code window, where a new App Code can be defined.

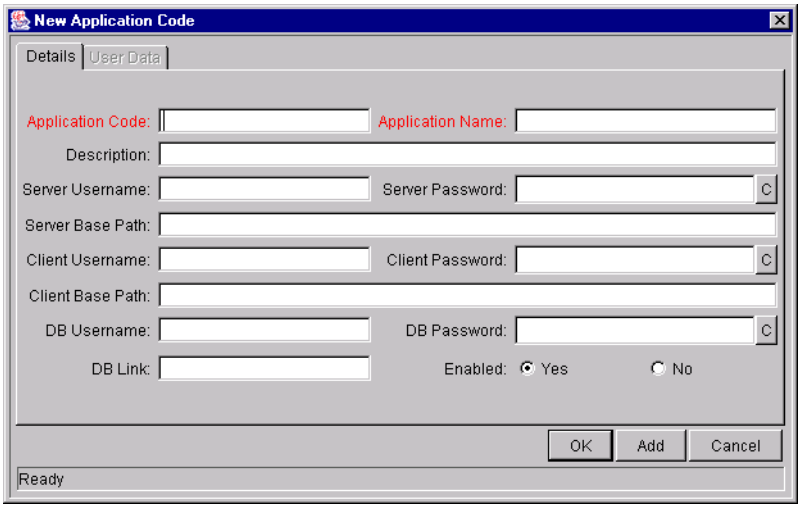


Figure 8-2 Application Code Window

Table 8-1. Environment Window - Application Fields

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	Username that Mercury Change Management should logon 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	Username 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.

Table 8-1. Environment Window - Application Fields [continued]

Field Name	Description
DB Username	DB username for the application. It is used when running database level commands (such as SQL scripts) for this application.
DB Password	DB 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 DB Link.
Enabled	Identifies if this application Environment is currently enabled.
New App	Brings up a dialog allowing the user to generate a new App Code.
Edit App	Allows the user to edit the selected App Code.
Copy App	Brings up a dialog box that allows the user to copy a selected App Code.
Delete App	Removes the selected App Code.
Copy Apps From	Brings up a dialog box that allows a user to copy App Codes from other Environments. For more information see *"Copying App Codes from Other Environments"* on page 163.

Copying App 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 set-up process.

To copy the App Codes:

1. Open the Environment Workbench.

   a. Click **Environments** in the shortcut bar.

   b. Click the **Environments** icon.

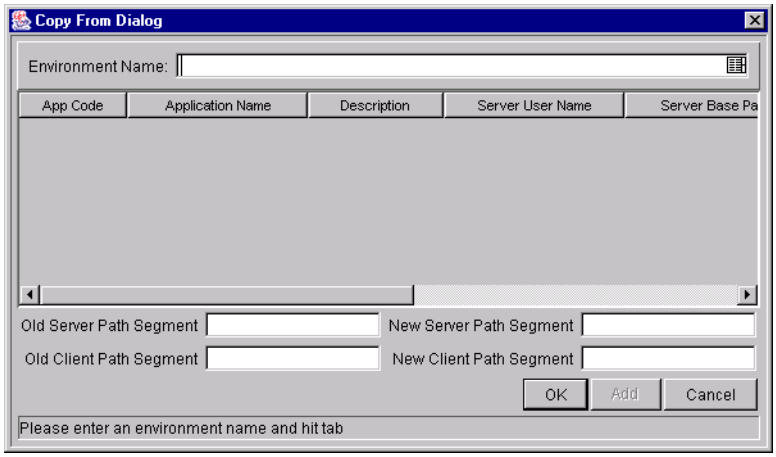
2. Click **New Environment** on the Environment Workbench, or select **File > New Environment**.

   A blank Environment window appears.

3. Click the **Applications** tab.

4. Click **Copy Apps From**.

   The Copy From Dialog opens.



5. Select the Environment from which the applications are to be copied from the Environment Name auto-complete list.

   The names of App Codes are displayed in the list.

6. To change the base path segment of all the app codes selected, enter the old server/client base path segment and the new server/client base path segment in the appropriate fields.

7. 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/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.

After copying the App Codes, any necessary modifications such as adding additional applications, deleting applications, or editing any of the applications can be made.

Setting the Access for Environments

It is possible to control which users can access an Environment for use in Environment Groups and Workflows.

To specify who can use an Environment in Environment Groups and Workflows:

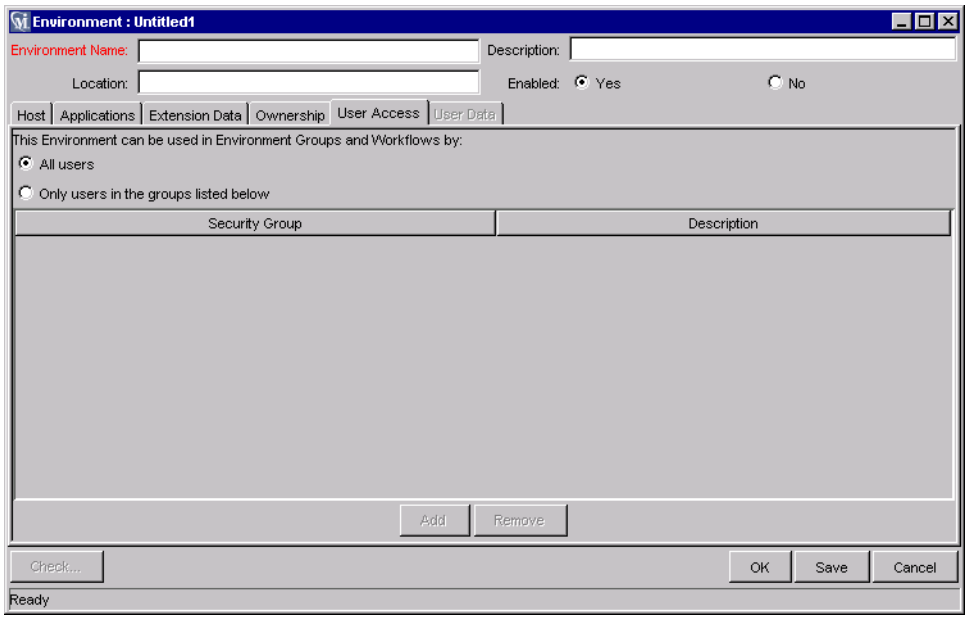
1. Click the **Environments** shortcut bar and click the **Environments** icon.

   The Environment Workbench opens.

2. Click **New Environment** to create a new Environment, or click **List** to open an existing Environment.

   The Environment window opens.

3. Click the **User Access** tab.



4. Select the **Only users in the groups listed below** option.

5. Click **Add**.

   The Add Security Groups window opens.

6. Select one or more Security Groups from the Security Group auto-complete list.

7. To close the Add Security Group window, click **OK**.

   The Security Group you selected display in the **Access** tab.

8. To add more Security Groups, click **Add**. To save the changes and close the window, click **OK**. To save the changes and leave the Environment window open, click **Save**.

Now only members of the Security Group(s) specified in the **User Access** tab can use this Environment in Environment Groups and Workflows.

Creating 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.

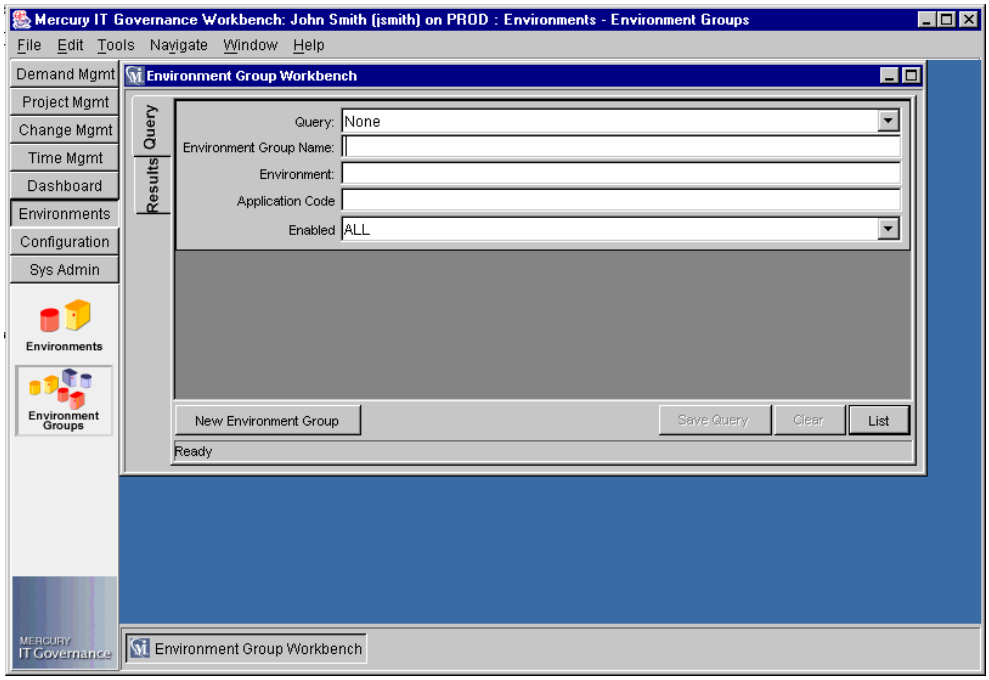
When to Use Environment Groups

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.

Defining an Environment Group

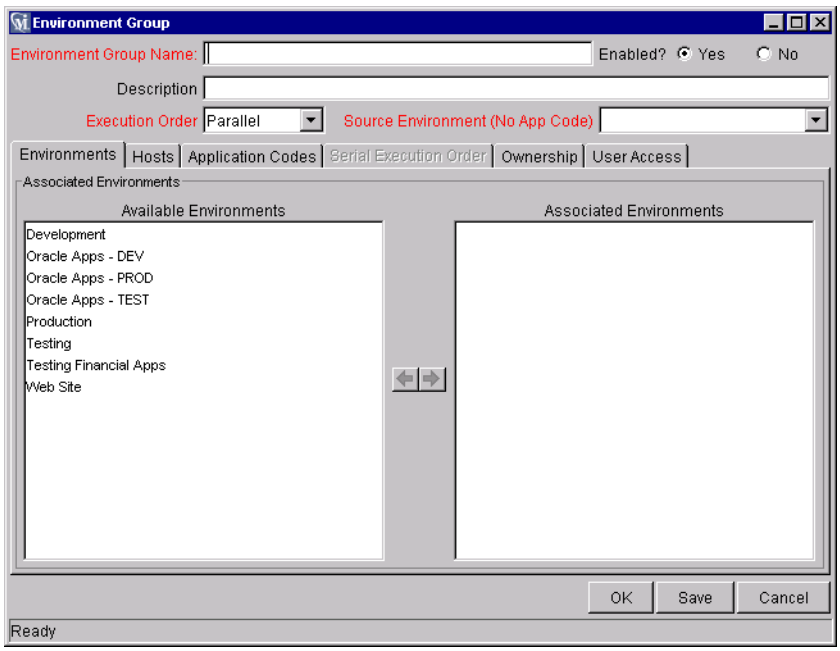
To define a new Environment Group:

1. Click **Environments** in the shortcut bar and click the **Environment Groups** icon.



2. Click **New Environment Group**.

   The Environment Group window opens.



3. Enter an Environment Group Name and Description.

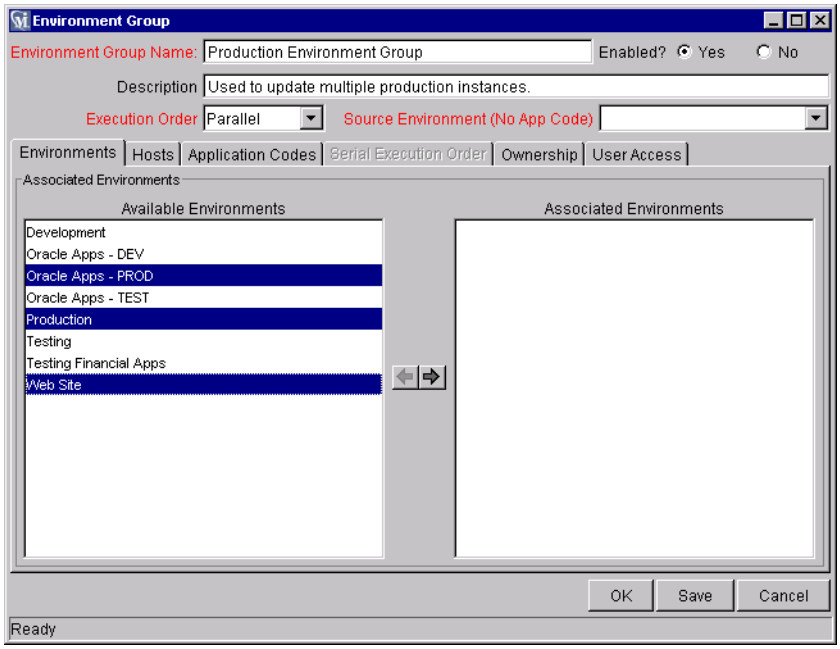
4. From the Execution Order field, select whether the Environments associated with the Environment Group are executed in **Parallel** order (all at once) or in a specific **Serial** order.

5. Select a Source Environment.

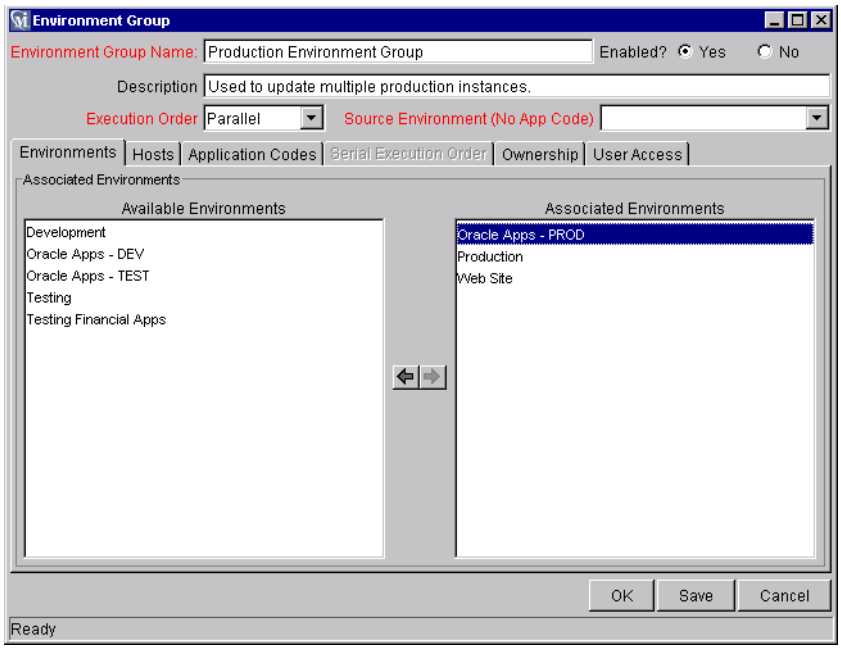
   This is the single Environment that will be used as the source of the deployment when an Environment Group is specified as the Source Environment in a Workflow Step.

6. Select the Environments to be included in the Environment Group from the Available Environments list.

   Press **Ctrl** and mouse click to select nonadjacent items in the list. Press **Shift** and mouse click to select multiple adjacent Environments.



7. Click the right arrow to move the selected Environments to the Associated Environments list.



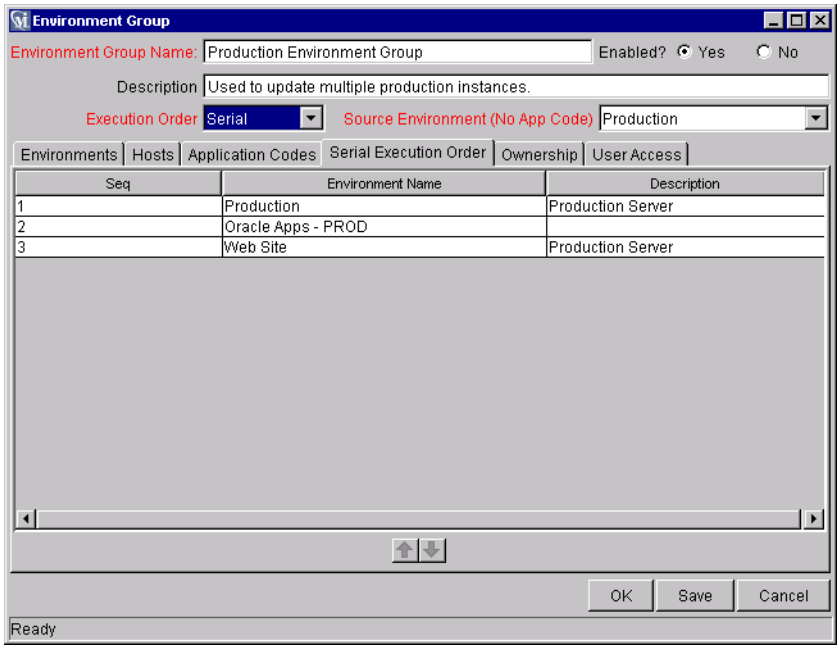
8. Click the **Application Codes** tab.

9. Select a Primary Source Environment for the listed Application Codes.

   Note that this is useful when specifying a Source Group Environment for an Execution step. See *"Choosing the Source Environment Based on Selected App Code"* on page 179 for more information.

10. Click the **Serial Execution Order** tab.

    This tab is only enabled if **Serial** is selected from the Execution Order drop down list.



11. Change the execution order by selecting an Environment and clicking the up and down arrows.

12. Click the **Ownership** tab to specify the users that can edit, copy and delete the Environment Group.

13. Click the **Access** tab to specify the users that can use this Environment Group in Workflows.

14. Verify that the Enabled radio button is set to **Yes** to enable use of this Environment Group in Workflow Step configuration.

15. Click **OK**.

This defines the new Environment and closes the Environment Group window.

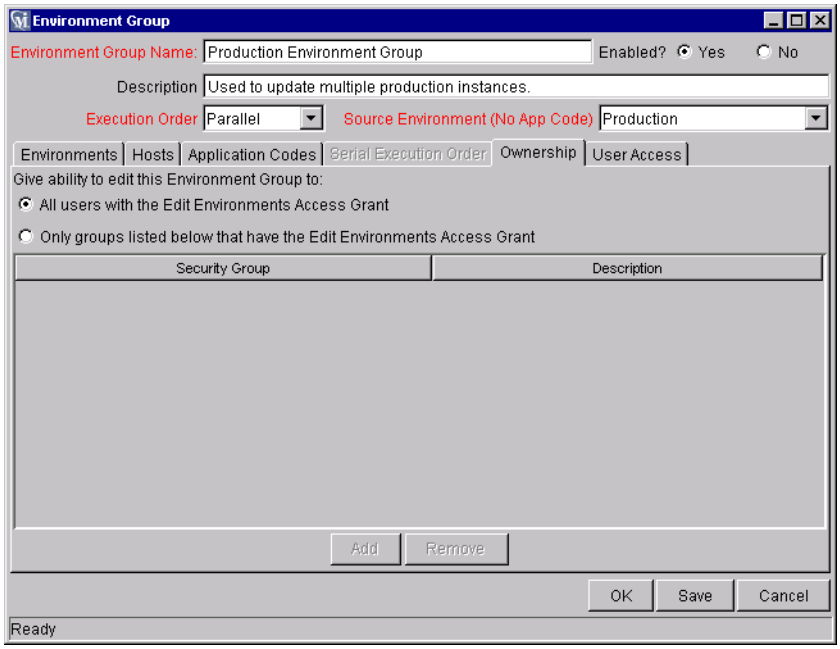
Setting Ownership for Environment Groups

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 Group window.
2. Click the **Ownership** tab.



3. Select the **Only groups listed below that have the Edit Environments Access Grant** option.
4. Click **Add**.

   The Add Security Groups window opens.
5. Select one or more Security Groups.
6. Click **OK** to close the Add Security Group window.

   The selected Security Groups display in the **Ownership** tab under the Security Group column.

7. Click **Add** to add more Security Groups. Click **OK** to save the changes and close the window. Click **Save** to save the changes and leave the Environment Group window open.

Only members of the Security Group(s) specified in the Ownership window can edit, delete or copy this Environment Group.

If no Ownership groups are associated with the entity, the entity is considered global and any user with the Edit Access Grant for the entity can edit, copy or delete it. Refer to *Security Model Guide and Reference* for more information on Access Grants.

Mercury Change Management administrators have the Ownership Override Access Grant and can access configuration entities even if the administrator is not a member of one of the Ownership Groups and does not have the Edit Access Grant.

If a Security Group is disabled or loses the Edit Access Grant, that group will no longer be able to edit the entity.

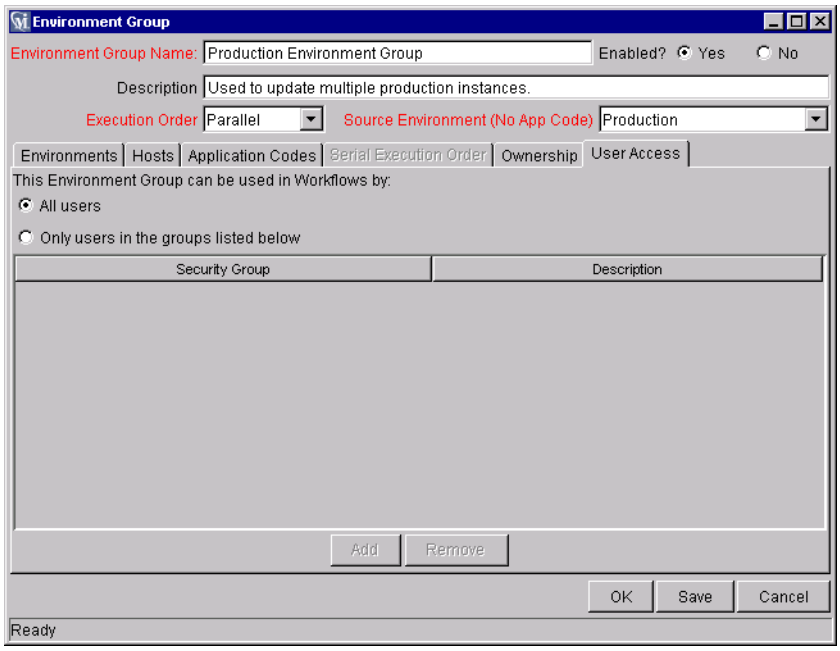
Setting the Access for Environments

It is possible to control which users can access an Environment Group for use in Workflows.

To specify who can use the Environment Group when defining a Workflow:

1. Open the Environment Group window.

2. Click the **Access** tab.



3. Select the **Only users in the groups listed below** option.
4. Click **Add**.

   The Add Security Groups window opens.
5. Select one or more Security Groups.
6. To close the Add Security Group window, click **OK**.

   The Security Group you selected display in the **User Access** tab.
7. Click **Add** to add more Security Groups. Click **OK** to save the changes and close the window. Click **Save** to save the changes and leave the Environment Group window open.

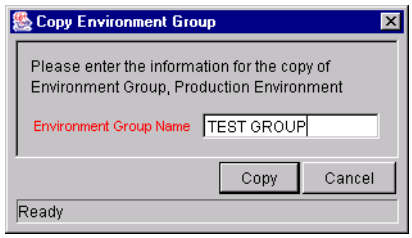
Only members of the Security Group(s) specified in the **User Access** tab can use this Environment Group in Workflows.

Copying an Environment Group

To generate a new Environment Group by copying an existing Environment Group:

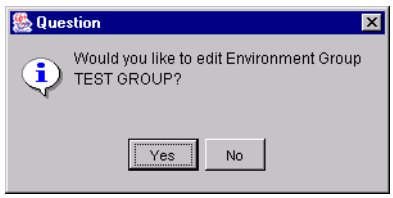
1. Enter search criteria in the **Query** tab of the Environment Group Workbench for the Environment Group to be copied.
2. To find all Environment Groups that match the search criteria, click **List**.
3. In the **Results** tab of the Environment Group Workbench, select the Environment Group to be copied.
4. Click **Copy**.

   The Copy Environment Group window opens.



5. Enter the new name for the copied Environment Group.
6. Click **OK** to copy the Environment Group.

   The following Question dialog opens.



7. Click **Yes** to edit the Environment Group or **No** to exit.

Note

You must be a member of one of the Environment Group's Ownership Groups in order to copy an Environment Group. See *"Setting Ownership for Environment Groups"* on page 170 for details.

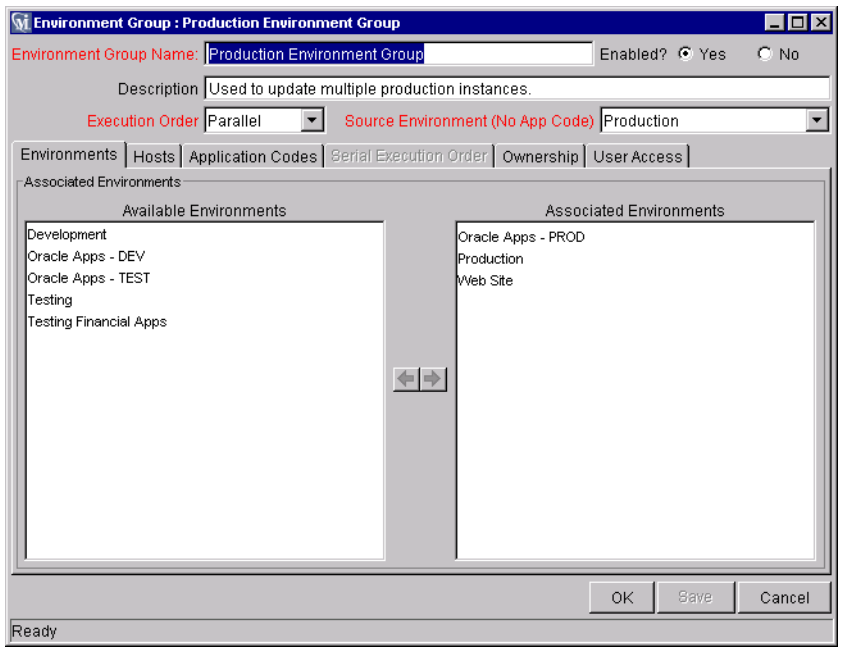
Adding Environments to an Environment Group

To add an Environment to an existing Environment Group:

1. Enter search criteria in the **Query** tab of the Environment Group Workbench for the Environment Group to be altered.

2. Click **List** to find all Environment Groups that match the search criteria.

3. Select the Environment Group in the **Results** tab and click **Open**.



4. Select the Environment(s) to be added to the Environment Group in the Available Environments list.

5. Click the right arrow to move the selected Environment(s) into the Associated Environments list.

6. Click the **Application Codes** tab and modify the Primary Source specification as desired.

7. Click the **Serial Application Order** tab, if enabled, and modify the Environment sequence as desired.

8. Click **OK**.

This adds the new Environment to the existing Environment Group.

Note

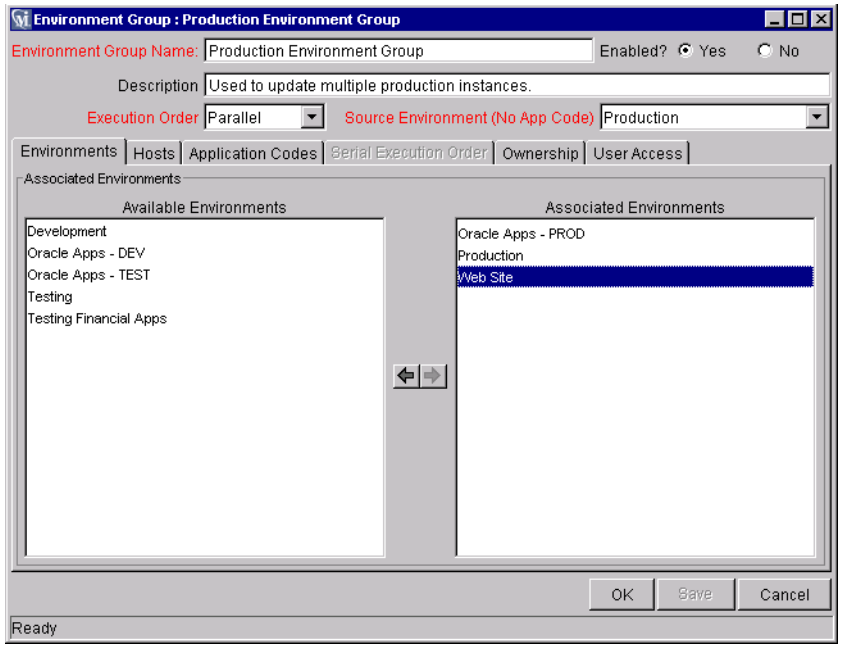
You must be a member of one of the Environment Group's Ownership Groups in order to modify an Environment Group. See *"Setting Ownership for Environment Groups"* on page 170 for details.

Only Environments that allow access (defined in the Access tab) will be included in Available Environments list. See *"Setting the Access for Environments"* on page 165 for more information.

Removing Environments from an Environment Group

To remove an Environment from an existing Environment Group:

1. Enter search criteria in the **Query** tab of the Environment Group Workbench for the Environment Group to be altered.
2. Click **List** to find all Environment Groups that match the search criteria.
3. Select the Environment Group in the **Results** tab and click **Open**.



4. Select the Environment(s) that to be removed from the Environment Group in the Associated Environments list.

5. Click the left arrow to move the selected Environment(s) into the Available Environments list.

6. Click on the **Application Codes** tab and modify the Primary Source specification as needed.

   If the Environment which was designated as the Primary Source is removed, then it is necessary to select a new Primary Source for the Application Code. See *"Choosing the Source Environment Based on Selected App Code"* on page 179 for more information.

7. Click the **Serial Application Order** tab, if enabled, and modify the Environment sequence as desired.

8. Click **OK**.

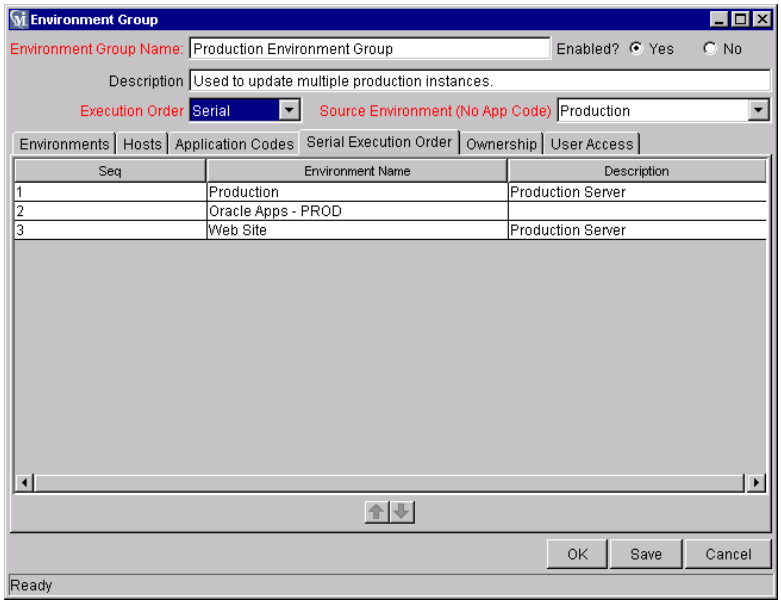
This removes the Environment from the Environment Group and closes the window.

You must be a member of one of the Environment Group's Ownership Groups in order to modify an Environment Group. See *"Setting Ownership for Environment Groups"* on page 170 for details.

Setting the Environment Execution Order

To set the Environment execution order:

1. In the Environment Group window, click the **Serial Application Order** tab.



2. Select a row containing the Environment to be moved.

3. Click the up or down arrow to move the selected Environment to a new sequence position.

   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.

4. Click **OK**.

This saves the changes and closes the window.

Note

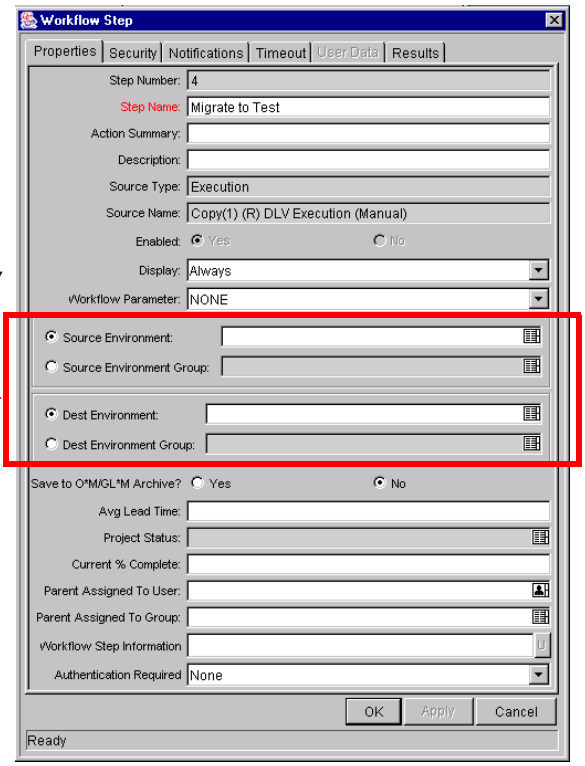
You must be a member of one of the Environment Group's Ownership Groups in order to modify an Environment Group. See *"Setting Ownership for Environment Groups"* on page 170 for details.

Linking Environments and Environment Groups to Workflows

Environments must be linked to Workflow Execution Steps that require connection, communication, or transfer between the clients, servers and databases used in the deployment system.

Environments are specified on 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 below.

Environments are specified for the execution Workflow steps.

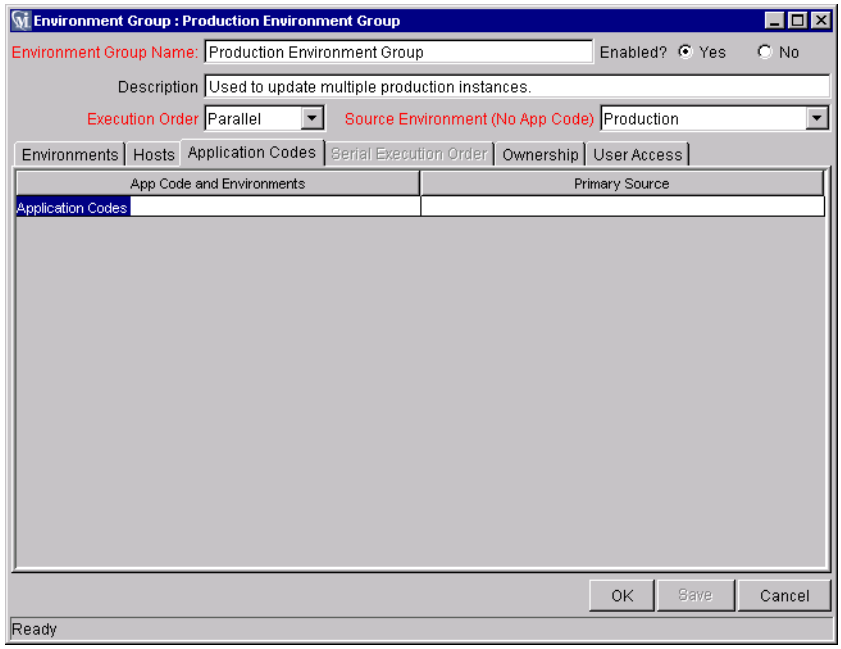


Choosing the Source Environment Based on Selected App Code

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 Apps Codes associated with an Environment are inherited by the Environment Group.

To enable the dynamic selection of a Source Environment based on the Application Code:

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



2. 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.

Note

You must be a member of one of the Environment Group's Ownership Groups in order to modify an Environment Group. See *"Setting Ownership for Environment Groups"* on page 170 for details.

Environment Maintenance and Utilities

This section provides information on validating and maintaining the Environment definitions in Mercury Change Management.

This section covers the following topics:

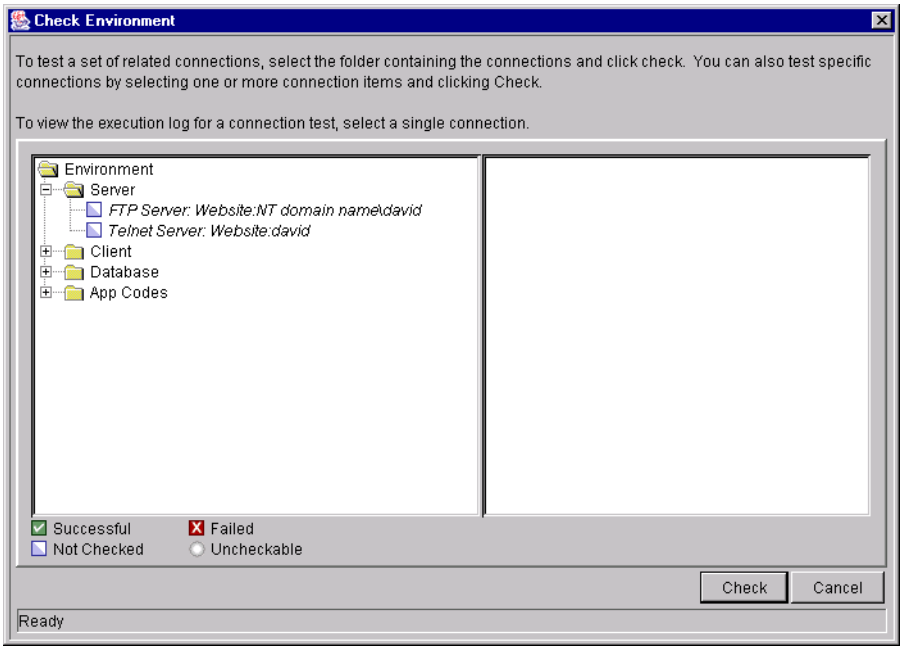
- *Testing the Environment Setup*
- *Mass Update of Base Paths*
- *Environment Password Management Utility*
- *Deleting Environments*

Testing the Environment Setup

To check the validity of the Environment:

1. From the Environment window, click **Check**.

   The Check Environment window opens.



2. Select the Environment connections to check.

   Select a folder (for example 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.

3. 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 set-up 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.

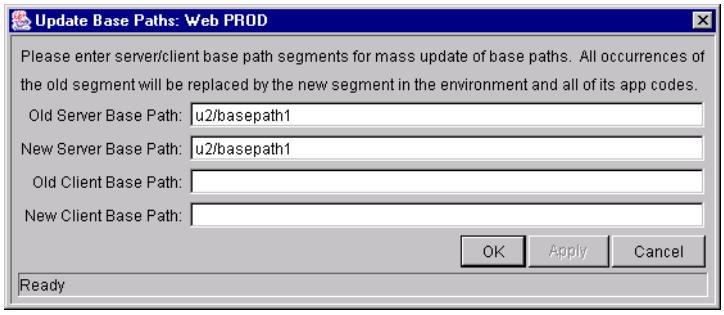
Mass Update of Base Paths

It is possible to update server/client base path segments in the Environment and all of its applications at the same time. This functionality is useful, for example, to relocate a particular Environment and all of its applications onto a new disk or partition.

To perform a mass update of base paths:

1. Select **Environment->Update Base Paths**.

   The Update Base Path window opens.



2. 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.

3. 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.

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. These updates can be made using the Mercury ITG Environment interface or directly at the command line.

This utility will only mass update passwords with matching parameters (such as Hostname, Username, Old Password, and Connect String).

This section covers the following topics:

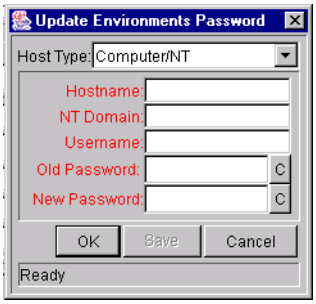
- *Updating Passwords Using the Workbench*
- *Updating Passwords Using the Command Prompt*

Updating Passwords Using the Workbench

To change a user's Environment password(s) using the Workbench:

1. Click the **Environments** screen in the **Environments** screen group.
2. From the **Environments** menu, select **Update Password**.

   The Update Environments Password window opens.



3. Select the Host Type.

   The required fields will dynamically change to match requirements of the selected Host Type.
4. Fill in the all of the fields.
5. Click **OK** to implement the changes.

Updating Passwords Using the Command Prompt

To change a user's Environment password(s) using the UNIX Command Line:

1. On the Mercury ITG Server, `cd` to the `bin` directory where Mercury Change Management is installed.
2. Run the following script:

   `kEnvUpdatePassword.sh`
3. Follow the command prompts to update the password.

   The prompts correspond to user and Environment information.

Deleting Environments

To delete an Environment:

1. Open the Environment Workbench window.
2. Enter the search criteria required to select the Environments in the **Query** tab and click **List**.

   The **Results** tab opens, displaying the results of the search.
3. Select the Environment to be deleted and click **Delete**.

   A dialog box appears asking for confirmation that the Environment is to be deleted.
4. Click **OK** to delete the Environment, or **Cancel** to leave it unchanged.

Only members of one of the Environment's Ownership Groups can delete an Environment. See *"Setting Ownership for Environments"* on page 659 for details.

Chapter 9
Integrating Participants into Your Deployment System

This chapter provides an overview for how to integrate users into the deployment process. It provides information for defining Security Groups and controlling users' access to actions in Mercury Change Management.

This chapter covers the following topics:

Only users with a Administrator license can create User accounts and Security Groups, which are critical when integrating participants into your deployment system. Work with the Administrator to configure the User accounts and Security Groups required for the process.

User Security and Participation - Overview

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, access can be enabled by specifying specific users or Security Groups. Access can also be dynamically provided by having a Token resolve to provide access.
  - Whether only "Participants" will 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 the deployment process:**
  - Who can change the Workflow.
  - Who can change each Object Type.
  - Who can change the Environment definitions.
  - Who can change the Security Group definitions.

Use Security Groups or dynamic access (Tokens) whenever possible. 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 any departmental reorganization), that list would need to be updated in many places on the Workflow. 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.

Establishing Security Groups

Security Groups are used to control who can access certain screens and functionality in Mercury Change Management. See *"Security Groups"* on page 17 for an overview. The following sections provide instructions on defining Security Groups:

- *Creating a Security Group by Specifying a List of Users*
- *Using Resource Management to Control User Security*

The general process for creating a Security Group is as follows:

1. Specify Security Group membership on the **Users** tab.

   This can be accomplished by providing a list of users or by associating the group with an organization unit that is defined in the Mercury IT Governance Center.

2. Specify the screen and feature access by linking the appropriate Access Grants.

   See *Security Model Guide and Reference* for details.

3. Specify which Workflows users in this Security Group can use when deploying changes.

   This is set in the **Change Management Workflows** tab.

4. Restrict the Security Group from using certain Application Codes when creating a Package Line.

   This restricts which applications each user can process objects through.

Consider creating and maintaining two types of Security Groups:

- Security Groups to control who can act on a specific Workflow Step (list of users without any special Access Grants)
- Security Groups to control who can access a particular screen or function (list of users and appropriate Access Grants)

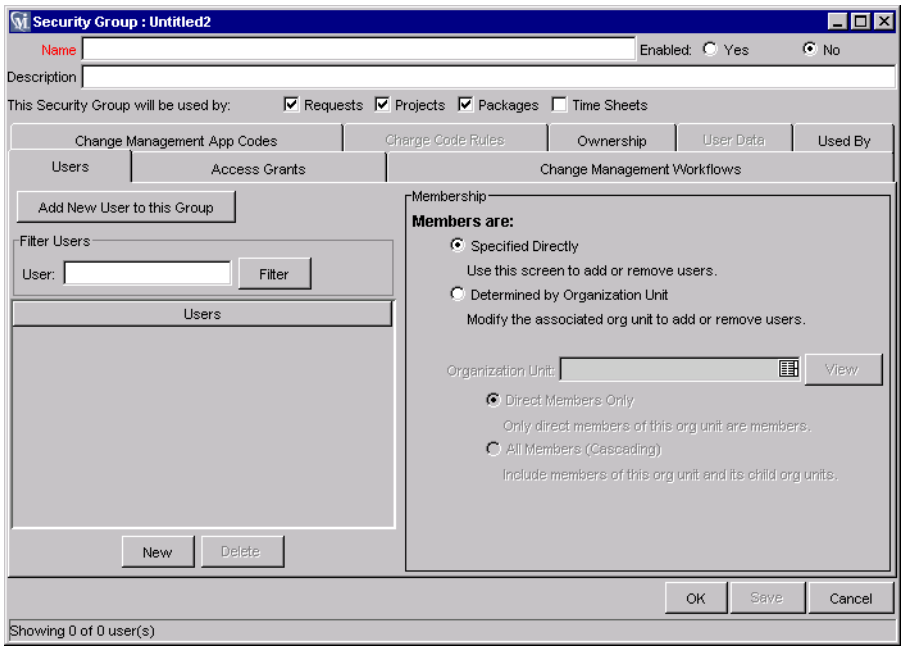
This can greatly simplify maintenance of a security model around deployment processes. As new users are added to the system, they can be granted to appropriate screen and function access and associated with specific Workflows.

Creating a Security Group by Specifying a List of Users

To generate and define a new Security Group:

1. Click **New Security Group** in the Security Group Workbench, or select **File > New > Security Group** from the menu.

   The Security Group window opens.



2. Enter the Name and Description.

3. Select **Yes** to enable this Security Group.

   If the Security Group is not enabled, it does not appear as a choice when generating or updating users or Workflows.

4. Select which entities (Requests, Projects or Packages) will use the Security Group by clicking their respective check boxes in the This Security Group will be used by field.

5. Link the desired Users to the Security Group.

   a. Click **New** in the **Users** tab.

      The Users window opens.

   b. Enter the desired username into the Users field and click **Add,** or click on the Users auto-complete list to display all available users.

The Validate window opens.

c. Select the desired User Name.

d. Click **OK**.

The Validate window closes.

e. Click **OK** to add your selection to the **Users** tab.

6. Link the desired Access Grants.

Each Access Grant enables certain functions performed on a Mercury ITG screen. See *Security Model Guide and Reference* for a description of each available Access Grants.

a. Select the desired Access Grants in the Available Access Grants list.

b. Click the right arrow button pointing to the Linked Access Grants list.

The selected Access Grants are moved into the column.

7. Restrict the Security Group from using certain Workflows.

a. Click the **Change Management Workflows** tab.

b. Select the Workflows in the Allowed Change Management Workflows list.

c. Click the left arrow button pointing to the Restricted Change Management Workflows list.

The selected Workflows are moved into the column.

d. If all future Workflows should also be excluded, select the Always restrict new Workflows check box.

8. Restrict the Security Group from using certain Application Codes when creating a Package Line.

This restricts which applications each user can process objects through.

a. Click the **Change Management App Codes** tab.

b. Select the App Codes in the Allowed Change Management App Codes list.

c. Click the left arrow button pointing to the Restricted Change Management App Codes list.

   If all future App Codes are to be excluded, select the **Always restrict new App Codes** check box.

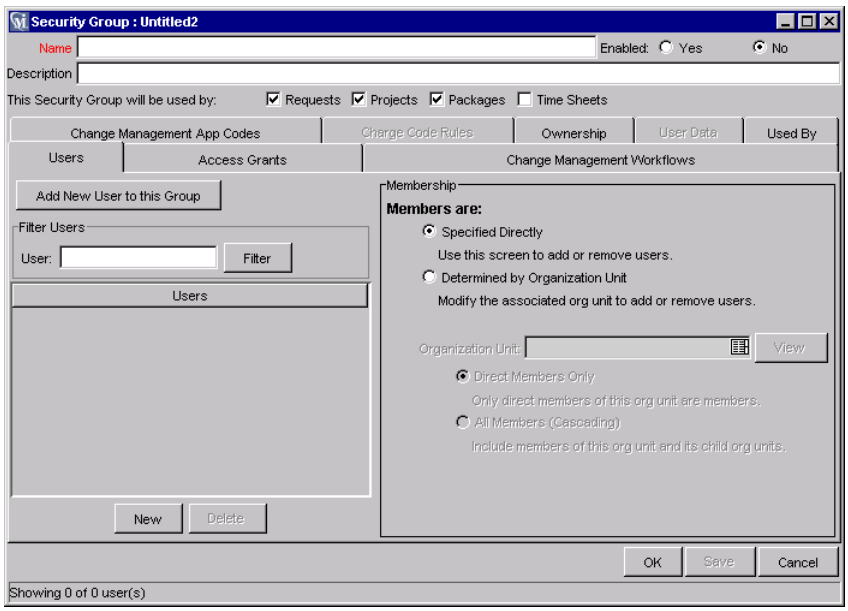
9. Click the **Ownership** tab and select the Ownership Groups that have the right to edit, copy or delete the current Security Group.

10. (Optional) Enter any necessary information in the **User Data** tab's custom fields.

11. To register the current Security Group and close the Security Group window, click **OK**. To save the information and leave the Security Group window open, click **Save**.

Using Resource Management to Control User Security

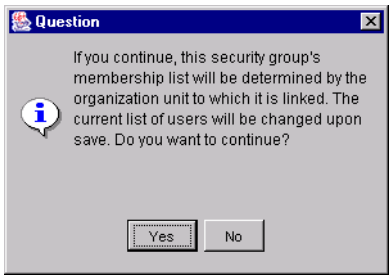
Users can also be associated to Security Groups through their inclusion in an organization model definition. Using the Mercury IT Governance Center's resource management capabilities, a user can be placed into a model that includes security and access information. See *Managing Your Resources (Resource Management)* for details.



To define a Security Group to use the members of an organization unit:

1. Open the Security Group window.
2. Select Determined by Organization Unit in the Membership section of the **Users** tab.

   The following question dialog opens.



3. Click **Yes**.

Note

When selecting an Organization Unit to control user access to the Security Group, any users specified in the Users list will be replaced with the members of the organization unit.

4. Select the Organization Unit.
5. Select whether to include:
   - Direct Members Only:
     Only direct members of the specified organization unit.
   - All Members (cascading)
     Members of this organization unit and its child units.
6. Click **Save**.

Related Topics:

- *Managing Your Resources (Resource Management)*
- *Security Model Guide and Reference*

Setting Package Creation Security

It is possible to control who can create certain Packages or use specific Object Types and Workflows. This provides a great deal of control over who can process changes of a certain type to specific Environments.

The following sections discuss how to control security related to Package creation:

- *Enabling Users to Create Packages*
- *Restricting Users from Selecting a Specific Workflow*
- *Restricting Users from Selecting a Specific Object Type*

Enabling Users to Create Packages

It is possible to control which users have the ability to create and submit Packages. To enable a user to create and submit a Package, ensure that the following settings are configured properly.

Table 9-1. Settings required to enable a user to create Packages

Setting	Value	Description
License	Change Management: Power License	The Power License provides a user with access to the Workbench, where the Package is defined. This is set in the User window on the Workbench.

Table 9-1. Settings required to enable a user to create Packages [continued]

Setting	Value	Description
Access Grants linked to the Security Group	Change Mgmt: Edit Packages	This Access Grant allows the user to generate, edit and delete certain Packages. • User cannot delete a Package if it has been released or if user is not the owner. • To edit the Package, user must be its creator, the 'assigned to' user, a member of the assigned group or a member of the Workflow Steps Security Group. Access Grants are set in the Security Group window.
	Change Mgmt: Manage Packages	This Access Grant allows the user to create, edit and delete Packages at anytime. Access Grants are set in the Security Group window.
Allowed Change Management Workflows in the Security Group window	At least one Workflow must be allowed.	When creating a Package, you are required to select a Workflow for the Package to proceed through. At least one Workflow must be enabled to be able to create and submit a Package. The user should select the Workflow intended to process the deploying objects. This is set on the Security Group window - **Change Management Workflows** tab.
Allowed Change Management Object Types in the Workflow window.	At least one Object Type in each Workflow used to deploy changes must be allowed.	Object Types can be associated with Workflows such that only certain Object Types can be processed through the Workflow. At least one Object Type must be enabled so that the user can create a Package Line when using that Workflow. This is set on the Workflow window - **Change Management Settings** tab, under the Package Line selection.

Screen and function access provided through Access Grants are cumulative. If a user belongs to three different Security Groups, the user will have all access provided to each of the groups. Therefore, to restrict certain screen and feature access, remove the user from any Security Group granting that access.

Use the **Access Grants** tabs in the User window to see all Security Groups where specific Access Grants are included, then:

- Remove the user from the Security Group (using the **Security Group** tab on the User window)
- Remove the Access Grants from the Security Group (in the Security Group window). Note: you should only do this if no one in that Security Group needs the access provided in that Access Grant.

Restricting Users from Selecting a Specific Workflow

It is possible to restrict users from selecting specific Workflows when creating a new Package. To do this, ensure that the following conditions are met.

Table 9-2. Settings required to restrict Workflow selection

Setting	Value	Description
Restricted Change Management Workflows in the Security Group window	Include the Workflows that you would like to restrict.	When creating a Package, you are required to select a Workflow for the Package to proceed through. Users (in the Security Group) will not be able to select any Workflows included in the Restricted Change Management Workflows list. Note: If a user belongs to another Security Group that allows the use of that Workflow, the user will be able to select it. This is set on the Security Group window - **Change Management Workflows** tab.

Restricting the Workflow selection also controls who can deploy changes to specific Environments, because the source and destination Environments are defined in the Workflow Step.

Restricting Users from Selecting a Specific Object Type

It is possible to restrict users from selecting specific Object Types when creating a new Package. To do this, ensure that the following conditions are met.

Table 9-3. Settings required to restrict Object Type selection

Setting	Value	Description
Restricted Change Management Object Types in the Workflow window.	Include the Object Type that you would like to restrict.	You can associate Object Types with Workflows such that only certain Object Types can be processed through the Workflow. Users (in the Security Group) will not be able to select any Object Types included in the Restricted Change Management Workflows list. This is set on the Workflow window - **Change Management Settings** tab, under the Package Line selection.

Setting Package Processing Security

It is possible to control who can process Packages following a Package submission. It is also possible to control who can act on certain steps (decisions and executions) in the process. The following sections discuss how to control security related to Package processing:

- *Providing Users with General Access to Update Packages*
- *Enabling Users to Act on a Specific Workflow Step*
- *Restricting Package Processing to Participants*

Providing Users with General Access to Update Packages

All users who will be processing Packages must meet the following conditions:

Table 9-4. Settings required to enable a user to process Packages

Setting	Value	Description
License (at least one is required)	Change Mgmt: Power License	The Power License provides a user with access to the Workbench. Users can act on all Workflow Steps (decisions and executions) in the Workbench. This is set in the User window on the Workbench.
	Change Mgmt: Standard	The Standard License provides a user with access to the standard interface. Users can act on all decision Workflow Steps. Note: you must have a Power Licence to process Execution Steps. This is set in the User window on the Workbench.
Access Grants linked to the Security Group	Change Mgmt: Edit Packages	This Access Grant allows the user to generate, edit and Packages. • User cannot delete a Package if it has been released or if user is not the owner. • To edit the Package, user must be its creator, the assigned to user, a member of the assigned group or a member of the Workflow Steps Security Group. Access Grants are set in the Security Group window.
	Change Mgmt: Manage Packages	This Access Grant allows the user to edit or delete Packages at anytime. Access Grants are set in the Security Group window.

Enabling Users to Act on a Specific Workflow Step

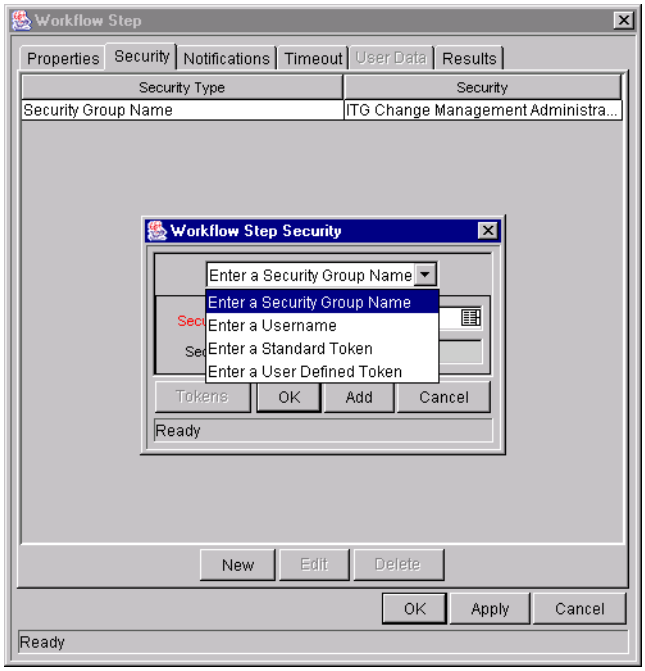
Specify who can act on each step in the deployment Workflow. Only people who are specified on the **Security** tab in the Workflow Step window will be able to process Packages and Package Lines at that step.

To specify the users who can act on a specific Workflow Step:

1. Open the Workflow.
2. Click the **Layout** tab.
3. Double click on the step to configure.

   The Workflow Step window opens. Note: the Workflow Step window also opens when first adding a step to the **Layout** tab.
4. Click the **Security** tab.
5. Click **New**.

   The Workflow Step Security window opens.



6. Select the method for specifying the step security from the drop down list: Security Group Name, Username, Standard Token, User Defined Token.

   Selecting a value from this field automatically updates the other fields on this window. For example, selecting **Enter a Username** will change the Security Group field to Username.
7. Specify the Security Groups, Usernames, or Tokens that will control the access to this step.

8. Click **OK**.

   The security specification is added to the **Security** tab. Add additional specifications to the step by clicking **New** and repeating the above process. The step's security can therefore be controlled using a combination of multiple Security Groups, Usernames and Tokens.

9. To save and close the window, click **OK**.

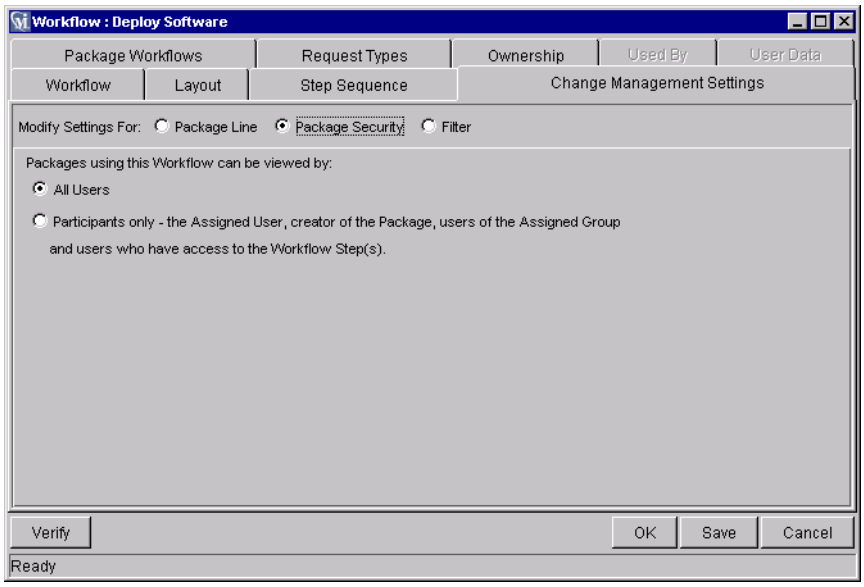
- Consider assigning a Security Group to each decision, execution and Condition Step, even though many of these steps will proceed automatically. If a command fails or a condition is not met, it may be necessary to manually override the step.
- Consider assigning a "Deployment Manager" Security Group to each step. That group could be configured with global access to act on every step in the process. Again, this could help avoid bottlenecks by providing a small group with permission to process stalled Packages.
- Avoid specifying a single user as the only person who can act on a Workflow Step. This would require a process update (re-configuration) when that user changes roles or leaves the company. Better to grant access dynamically using a Token or Security Group.

Restricting Package Processing to Participants

The Package Security section in the **Change Management Settings** tab in the Workflow window determines who can have access to Packages that use the current Workflow. Restricting access to Participants means that when non-Participant users search for Packages, a Package that uses the current Workflow will not be displayed. In this instance, Participants are defined as:

- The Assigned User
- The creator of the Package
- Members of the Assigned Group
- Any users who have access to the Workflow Step(s)

To allow all users access Packages using the current Workflow, select **All Users**.



To restrict the number of users who can access Packages using the current Workflow to Participants of the Packages, select **Participants only - the Assigned User, creator of the Package, users of this Assigned Group, and users who have access to the Workflow Step(s)**.

Setting Configuration Security

A critical part of ensuring successful deployments is ensuring that the deployment process is altered only by the correct people. Set configuration security around all of the Change Management configuration entities. This includes such activities as controlling:

- Who can change the Workflow.
- Who can change each Object Type.
- Who can change the Environment definitions.
- Who can change the Security Group definitions.

The following sections discuss some options for securing your configurations:

- *Setting Ownership for Configuration Entities*

- *Removing Access Grants*

Setting Ownership for Configuration Entities

Different groups of users have ownership and control over the Change Management entities. These groups are referred to as Ownership Groups. Unless a global permission has been designated to all users for an entity, members of Ownership Groups are the only users who have the right to edit, delete or copy that entity. The Ownership Groups must also have the proper Access Grant for the entity in order to complete those tasks. For example, the Edit Workflows Access Grant is needed to edit Workflows and Workflow Steps.

It is possible assign multiple Ownership Groups to the various entities. Ownership Groups are defined in the Security Group window. Security Groups become Ownership Groups when used in the Ownership capacity.

Ownership Groups can be specified for the following entities involved in the deployment process:

- Workflows
- Workflow Steps
- Environments
- Environment Groups
- Object Types
- Security Groups
- User Definitions
- Report Types
- Validations
- Special Commands

The Ownership setting is accessed through the individual entity windows in the Workbench.

For example, to set the Ownership for Workflows:

1. Open the Workflow.
2. Click the **Ownership** tab.

3. Click the Only groups listed below that have the Edit Workflows Access Grant radio button.

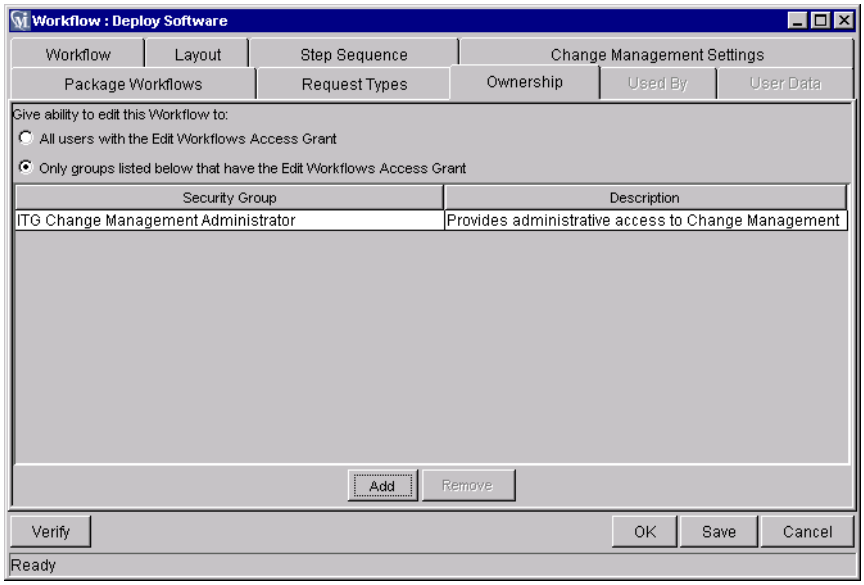
4. Click **Add**.

   The Add Security Group window opens.

5. Select the Security Group.

6. To add the current Security Group and continue adding more Security Groups, click **Add**. To add the current Security Group and close the Add Security Group window, click **OK**.

   The Security Group(s) you selected displays in the **Ownership** tab under the Security Group column.



7. To save the selection and close the Workflow window, click **OK**. To save the selection and leave the Workflow window open, click **Save**.

Note

The System: Ownership Override Access Grant allows the user to access and edit configuration entities even if the user is not a member of one of the entity's Ownership Groups.

Removing Access Grants

It is also possible to restrict the ability to modify Change Management configuration entities by removing the user from any Security Group granting that access.

Use the **Access Grants** tabs in the User window to see all Security Groups where specific Access Grants are included to either:

- Remove the user from the Security Group (using the **Security Group** tab on the User window)
- Remove the Access Grants from the Security Group (in the Security Group window). Note: only do this if no one in that Security Group needs the access provided in that Access Grant.

The following table lists the Access Grants that provide edit access to different configuration entities.

Table 9-5. Access Grants for editing configuration entities

Access Grant	Description
Config: Edit Workflows	Allows the user to generate, update and delete Workflows.
Config: Edit Report Types	Allows the user to generate, update and delete Report Types.
Config: Edit Special Commands	Allows the user to generate, update and delete Special Commands.
Config: Edit User Data	Allows the user to generate, update and delete User Data.
Config: Edit Validation Values	Allows the user to generate, update and delete Validation Values.
Config: Edit Validations	Allows the user to generate, update and delete Validations.
Config: Edit Workflows	Allows the user to generate, update and delete Workflows.
Change Mgmt: Edit Object Types	Allows the user to generate, update and delete Object Types.
Environments: Edit Environments	Allows the user to generate, update and delete Environments.
Sys Admin: Edit Security Groups	Allows the user to generate, update and delete Security Groups.

Table 9-5. Access Grants for editing configuration entities [continued]

Access Grant	Description
Sys Admin: Edit Users	Users Allows the user to generate, update and delete Users.

Chapter 10
Setting Up Communication Paths

This chapter provides an overview for different modes of communication that can be used in a deployment system. The following features help increase visibility into the deployment processes and data:

- Email Notifications:
  Each Workflow Step can be configured to send an email to specified users when the step becomes eligible, has a specific result, or encounters an error. Using Notifications at key points in the process ensures a speedy deployment by notifying appropriate parties of actions required by them or complications during deployment.
- Dashboard:
  The Dashboard provides an interface through which the current state of the deployments can be quickly assessed. Personalize the Dashboard to display status information that is most meaningful to any given role. For example, software engineers may only want to see the Packages that they submitted, whereas the IT manager may want to have visibility into each critical Package currently in progress.
- Reports:
  Mercury Change Management includes a number of reports that can be used to assess deployment status. Change Management also publishes a reporting meta layer that enables custom reports to be built. Reports can be scheduled to run periodically.

This chapter discusses the following topics:

Adding Notifications to Workflow Steps

When configuring a Notification for a Workflow Step, consider the following:

- When to send it.
- Who should receive it.
- What the message should say.

This section covers the following topics:

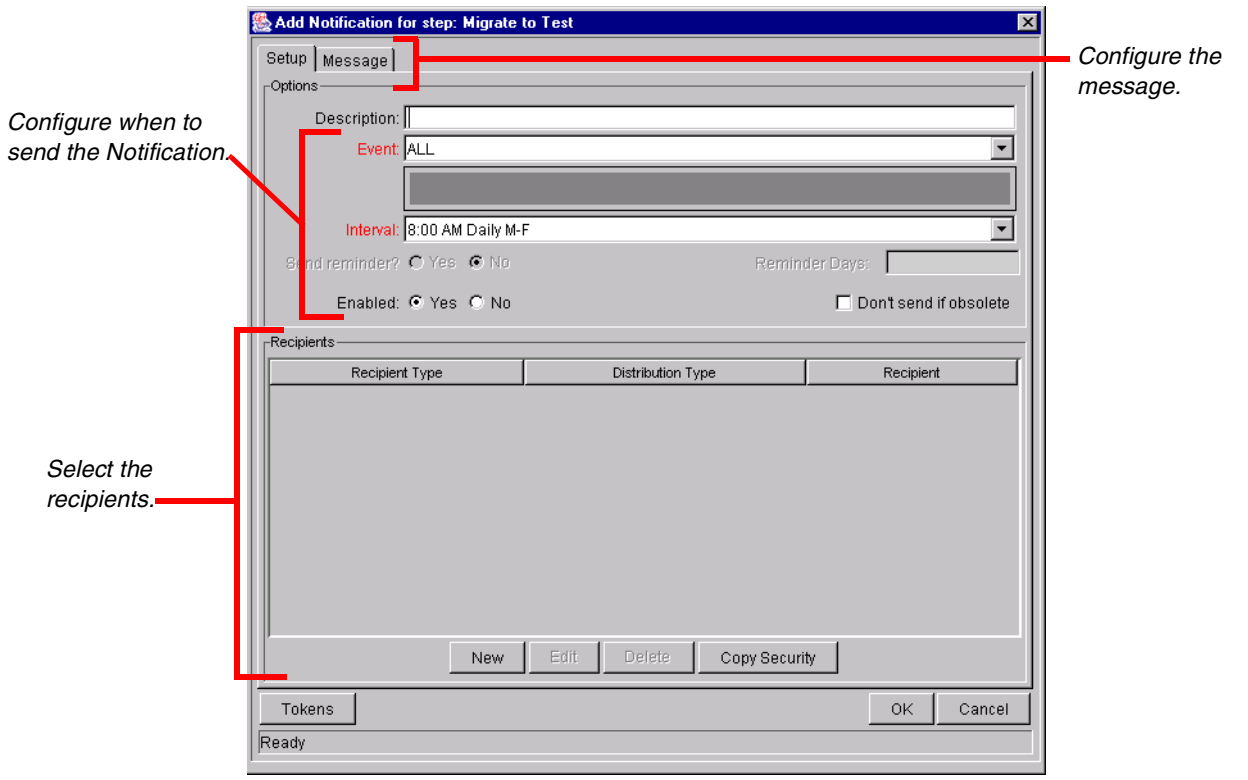
Adding a Notification to a Workflow Step - Overview

To add a Notification to a Workflow Step:

1. Open the Workflow.
2. Click the **Layout** tab.
3. Double click on the step to configure.

   The Workflow Step window opens. Note: the Workflow Step window also opens when first adding a step to the **Layout** tab.
4. Click the **Notifications** tab.
5. Click **New**.

   The Add Notification for Step window opens.



6. Configure the following:
   - When the Notification is sent (Event and Interval)
   - Who receives the Notification (Recipients)
   - The body of the Notification (Message)
7. Click **OK**.

   The Notification specification is added to the **Notifications** tab. Add additional specifications to the step by clicking **New** and repeating the above process. A different Notification can then be configured to be sent to different recipients for different events.
8. To save and close the window, click **OK**.

Configuring When to Send a Notification

Each Workflow Step can be configured to send an email when the step becomes eligible, has a specific result, or encounters an error.

This section covers the following topics:

- *Sending a Notification when a step becomes eligible*
- *Sending a Notification when a step has a specific result*
- *Sending a Notification when the step has a specific error*
- *Configuring multiple Notifications for a single step*
- *Specifying the Time the Notification is Sent*

Sending a Notification when a step becomes eligible

To send a Notification when a Workflow Step becomes eligible, configure the Notification as indicated below.

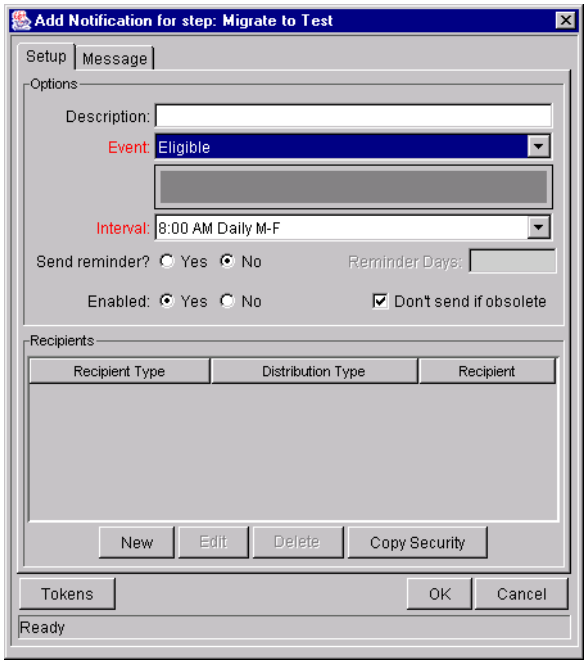


Table 10-1. Workflow Step Notification - send on eligible

Field	Value	Notes
Event	Eligible	
Interval	Immediate	Select to send the Notification 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 batched and sent together. Selecting an interval other than "Immediate" will allow this batching to occur. See *"Configuring the Notification Intervals"* on page 218 for instructions on configuring this.
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	

Sending a Notification when a step has a specific result

It is possible to configure the Notification to be sent when a Workflow Step results in a specific decision or execution result. The value for these events is determined by the Workflow Step Source's Validation.

To send a Notification when a Workflow has a specific result, configure the Notification as indicated below.

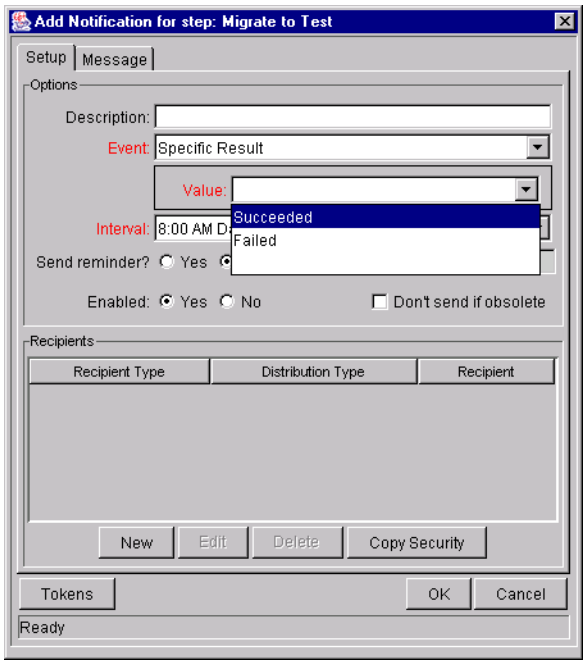


Table 10-2. Workflow Step Notification configuration - send on step result

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.

Table 10-2. Workflow Step Notification configuration - send on step result

Field	Value	Notes
Interval	Immediate	Select to send the Notification 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 batched and sent together. Selecting an interval other than "Immediate" will allow this batching to occur. See *"Configuring the Notification Intervals"* on page 218 for instructions on configuring this.
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	

Sending a Notification when the step has a specific error

To send a Notification when a Workflow has a specific error, configure the Notification as indicated below.

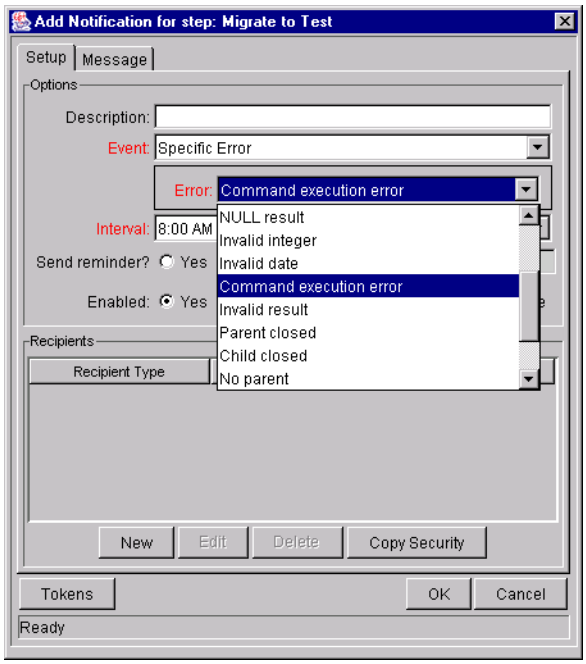


Table 10-3. Workflow Step Notification configuration - send on error

Field	Value	Notes
Event	Specific Error	
Error	Select the value to trigger the Notification.	This is a standard set of errors. See *"Specific Errors for Workflow Steps"* on page 215.
Interval	Immediate	Select to send the Notification 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 batched and sent together. Selecting an interval other than "Immediate" will allow this batching to occur. See *"Configuring the Notification Intervals"* on page 218 for instructions on configuring this.

Table 10-3. Workflow Step Notification configuration - send on error

Field	Value	Notes
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	

Specific Errors for Workflow Steps

The following errors can cause a Notification to be sent.

Table 10-4. 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 Package Line is expecting a message from a Request that is cancelled or closed.

Table 10-4. Specific Errors for Workflow Steps

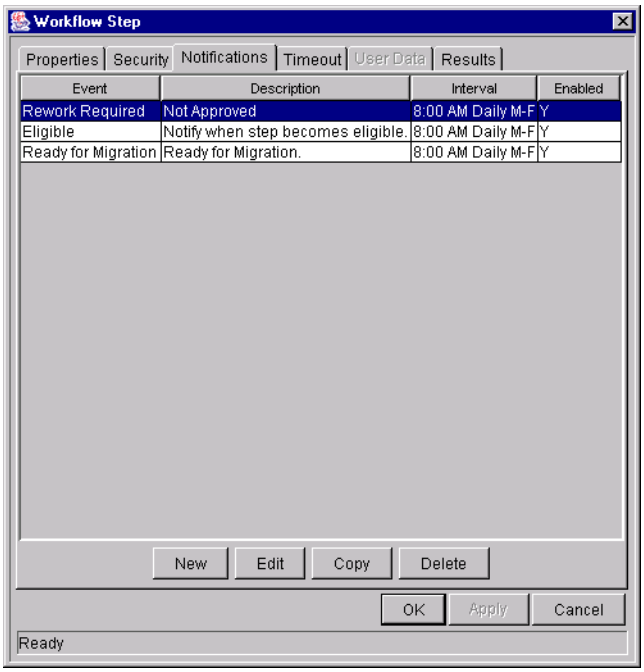
Specific Error	Meaning
Child closed	For wf_receive or wf_jump steps, a Request is expecting a message from a Package Line that is cancelled or closed.
No parent	For wf_receive or wf_jump steps, a Package Line is expecting a message from a Request that has been deleted.
No child	For wf_receive or wf_jump 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.
Multiple Return Results	When the Package Level subworkflow receives multiple results from Package Lines that traversed through it.

Configuring multiple Notifications for a single step

It is possible to configure multiple Notifications for each Workflow Step. This can be useful in the following sample situations:

- Sending a different message depending on the result of the step
- Sending a different message depending on the type error
- Sending the Notification 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

To configure multiple Notifications for a Workflow Step, simply add multiple Notifications to the same Workflow Step window.

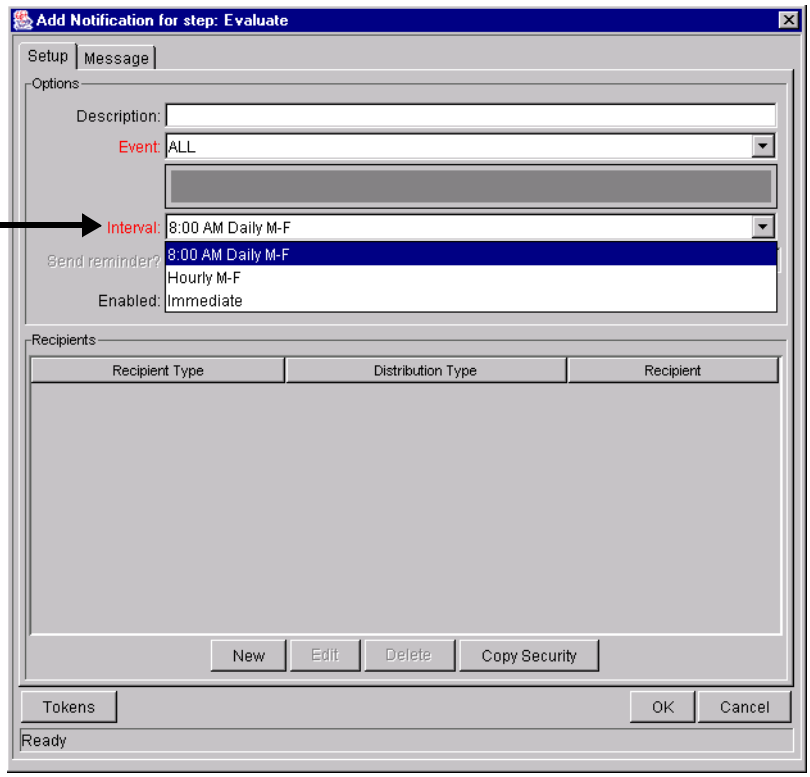


In this example, one set of users is notified when the step becomes eligible. Then, depending on the outcome of the step, different groups are notified. If the step results in the specific result of "Ready for Migration," then a Notification is sent to an engineer who will execute the migration. If the step results in the specific result of "Rework Required," then a Notification is sent to the deployment manager.

Specifying the Time the Notification is Sent

Use the Interval field on the Workflow Step to specify when the Notification will be sent.

Select an Interval to specify when and how often the Notification is sent.



The interval determines when the Notification will be sent. The following intervals are pre-configured:

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

Configuring the Notification Intervals

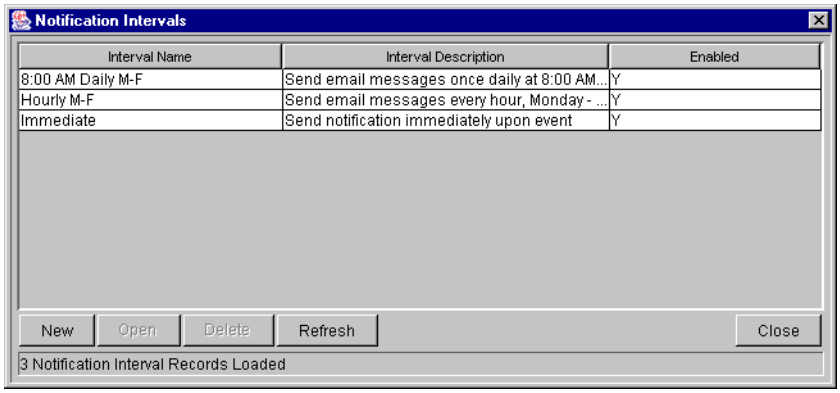
Notifications are configured on the Notification Templates Workbench.

To configure the Notification Intervals:

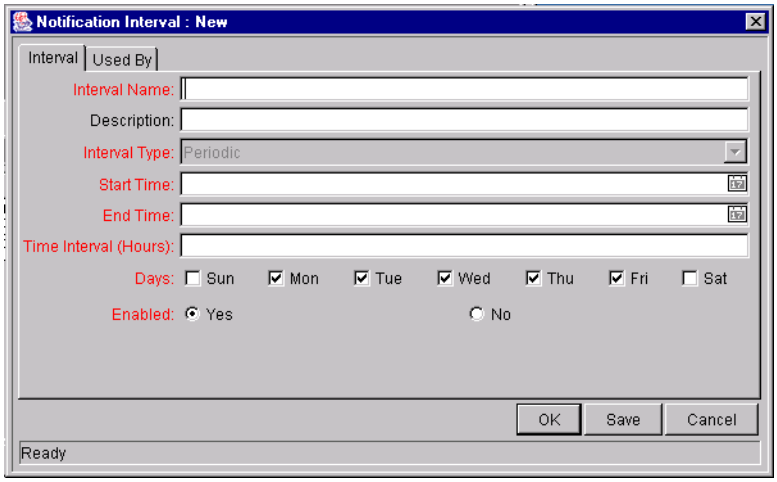
1. Click the **Configuration** screen group and click the **Notification Templates** icon.

2. Select **Notification Templates ->Intervals** from the menu.

   The Notification Intervals window opens.



3. Click **New** or **Open** to access the Notification Interval: New window.



4. Enter the required information on the **Interval** tab.

   These fields are defined in *Table 10-5*.

Table 10-5. Notification Intervals

Field Name	Description
Interval Name	This is the name assigned to the interval.
Description	Free form description of this interval.

Table 10-5. Notification Intervals

Field Name	Description
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. Click **OK**.

   The new interval is added to the Notification Intervals window.

6. Click **Close** to close the window.

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.

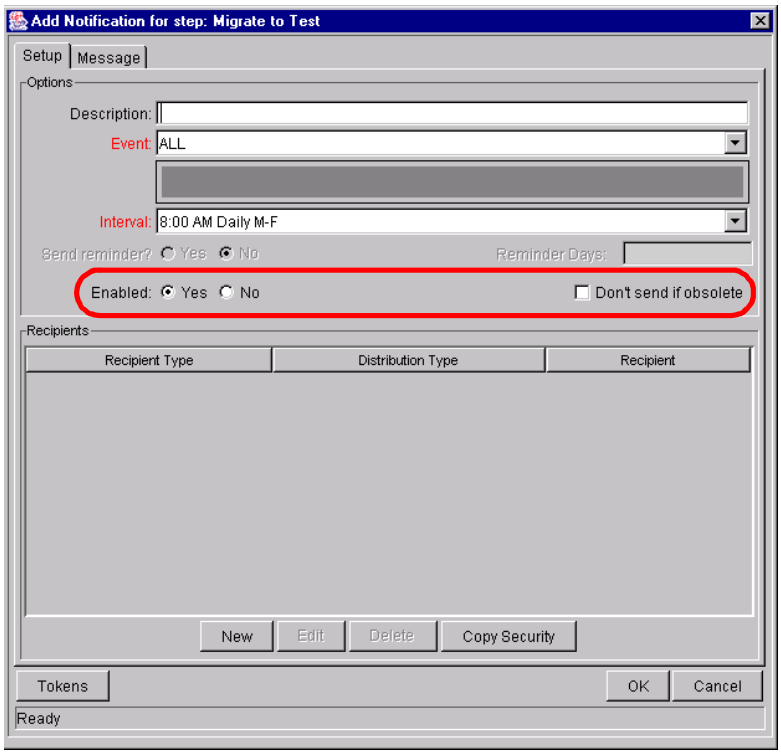
Sending a follow up Notification (reminder)

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 configure a Notification to re-send after a period of time, configure the Notification as indicated below.

Table 10-6. Workflow Step Notification configuration - send on error

Field	Value	Notes
Event		Select 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.

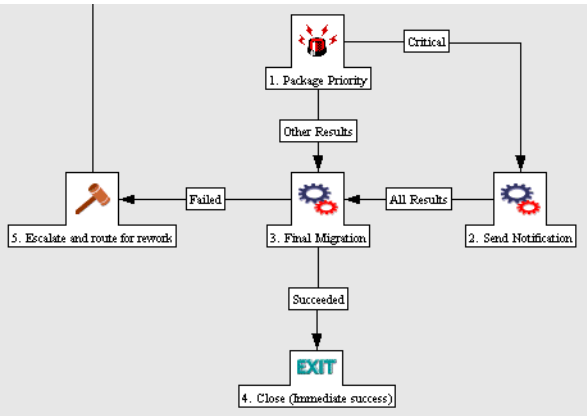


Configuration Tip: Sending a Notification Based on a Field Value.

Use the following configuration to send a Notification at a point in the process based on the value in a field. In this example, you want to send a Notification to the Deployment manager at the “Final Migration” step for all critical Packages. This will alert him to other actions/communications that he may want to prepare for. The configuration consists of the following:

- Token evaluation step for the priority field.

- Transition to another step (immediate execution) if the Priority = “Critical.”
- Notification upon entering the “Send Notification” step.



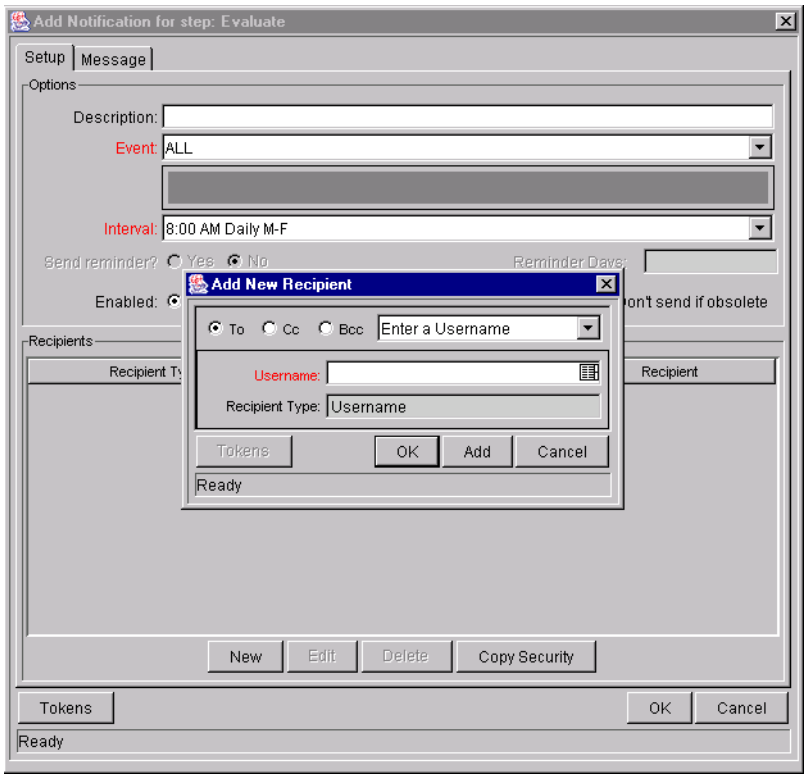
Configuring the Notification Recipients

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. Click **New** in the Add Notification for Step window.

   The Add New Recipient window opens.



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

   - **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 to send the Notification to.

   - **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 field below. For example, selecting **Enter a Security Group** will change the field below to Security Group.

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

   This can be a Username, Email Address, Security Group, or a Token.

Recipient Configuration Tips

Tip 1:
Use Security Groups or dynamic access (Tokens) 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 need 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.

Tip 2:
Use Tokens when sending a Notification to an undetermined party. For example, you can configure the Notification to be sent to the Assigned to User by specifying **[PKG.ASSIGNED_TO_USERNAME]** in the Add New Recipient window.

Configuring the Notification Message

Construct the Notification's message to ensure that it contains the correct information or instructions for the recipient. For example, if the Notification was sent to instruct the user that a Package deployment requires his approval, the message should instruct the user to log onto Mercury Change Management and update the Package's status. Additionally, the Notification should include a link (URL) to the referenced Package.

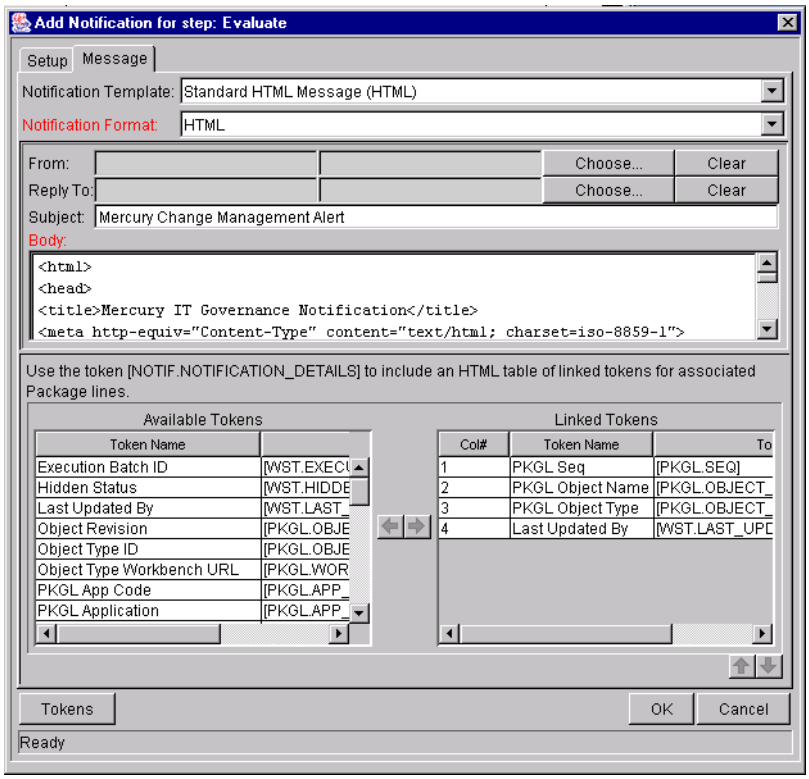
Notifications include the following features which make them easier to configure and use:

- Select from pre-configured Notification Templates to more quickly construct the body of your message.
- The body of the Notification can be plain text or HTML.

- Include multiple Tokens in the Notification. These Tokens will resolve to information relevant to the recipient. For example, include Tokens for the URL to the Package approval page, information on Package status and priority, and emergency contacts.

To configure the message in a Notification:

1. Click the **Message** tab on the Add Notification for Step window.



2. Select a Notification Template.

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

3. From the Notification Format field, select **HTML** or **Plain Text**.

   Selecting **HTML** allows you more flexibility when formatting the look and feel of the Notification. Write and test the HTML code in any HTML editor and then paste the code into the Body window.

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

5. Construct the Body of the message.

   When constructing the body, consider utilizing the following:

   - Token for the URL to the Package (Workbench or HTML interface). See *Table 10-7* for a list of these Tokens.

   - Tokens in the Body of the message:
     Click the **Tokens** button to access the Token Builder window where Tokens can be selected to add to the message body.

   - Tokens related to specific Package Lines:
     Add Tokens to the Linked Token list to include Tokens that resolve information related to the individual Package Line.

6. To save the Notification specification, click **OK**.

Using Tokens in the Message Body

Select any of the available Tokens accessed through the Token builder to include in the body of your message. Note, however, that not all Tokens will resolve in all situations. As a general rule, Tokens associated with the Package, Package Line, or Workflow will resolve.

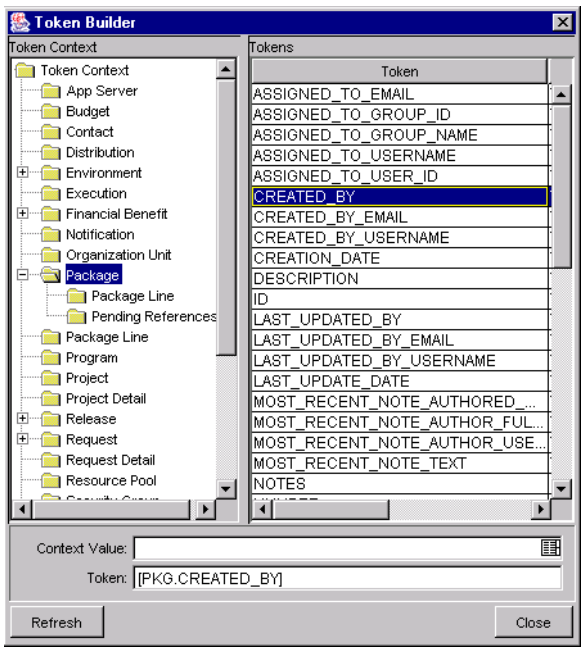
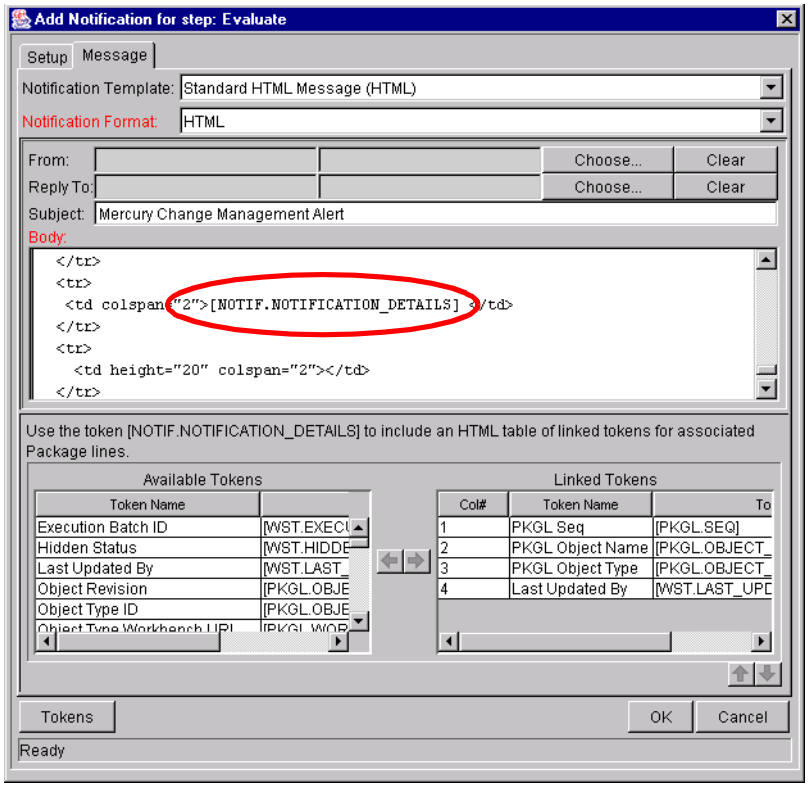


Figure 10-1 Token Builder Window

Special Case: Tokens in HTML Message

To use an HTML formatted Notification in the Package deployment process, include an additional Token in the body to include all of the linked Tokens. Each Change Management (Package) HTML Notification should include the Token **[NOTIF.NOTIFICATION_DETAILS]** within the <body> tags to incorporate linked Tokens.



Including URLs to Open the Package (Smart URLs)

When a user receives a Notification, it is often helpful to include a link to the item that needs their attention. For example, John Smith receives a Notification stating that his Package is ready for final approval. He clicks the URL included in the body of the Notification and proceeds directly to the Package approval page.

Notifications can be configured in the body of the email Notification to include the Web address (URL) for the following entities:

- Packages

- Requests
- Request Types
- Projects
- Tasks
- Workflows
- Validations
- Object Types
- Environments

If end-users are viewing their mail with a Web-based mail reader (such as Microsoft Outlook or Netscape Messenger), they can then 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 the user 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 10-7*.

Table 10-7. Smart URL Tokens

Smart URL Token	Description
PACKAGE_URL	Provides a URL that opens a Web page displaying the Package details.
WORKBENCH_PACKAGE_URL	Provides a URL that loads the Package window in the Workbench.
REQUEST_URL	Provides a URL that opens a Web page displaying the Request details.

Smart URLs in an HTML Formatted Messages

When using an HTML formatted message, use an alternate Token to provide a link to the Package.

Table 10-8. Smart URL Tokens in HTML Format

Smart URL Token	Description
PACKAGE_NO_LINK	Provides a link that loads the Package window in the Workbench.

The Token will resolve to the following format:

Package Name

In the Notification, the link would appear as:

Package Name

These Tokens can also be used in plain-text formatted Notifications. They will appear with the HTML tags showing.

Using Notification Templates

Notification Templates are pre-configured Notifications that can be used to quickly construct the body of a message. Notification Templates can be used in Workflow Steps, Report submissions, and Task State changes.

The following sections provide detailed instructions for:

- *Creating New Notification Templates*
- *Copying Notification Templates*
- *Deleting a Notification Template*

Creating New Notification Templates

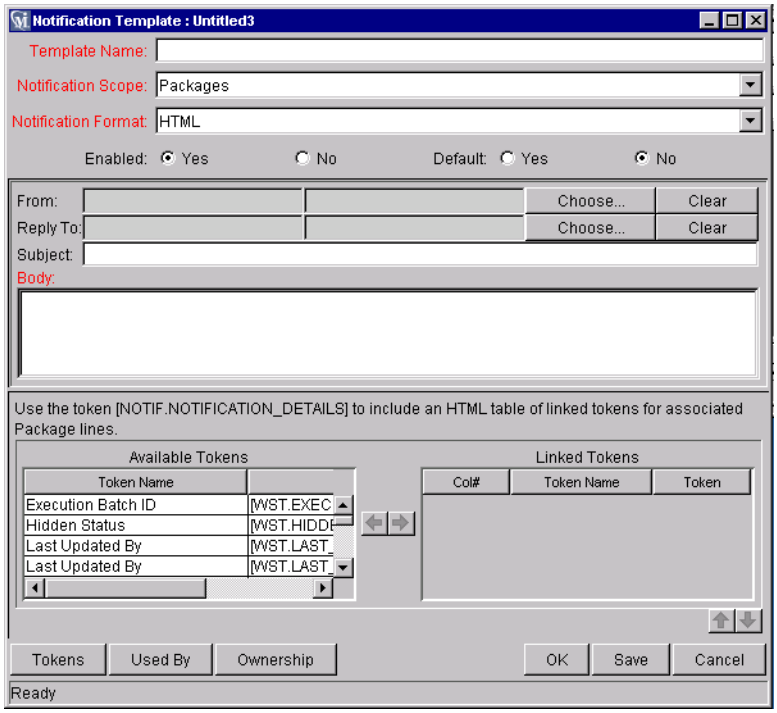
To create a new Notification Template:

1. Click the **Configuration** screen group and click the **Notification Templates** screen.

   The Notification Template Workbench opens.

2. Click **New Notification Template**.

   A Notification Template window opens.



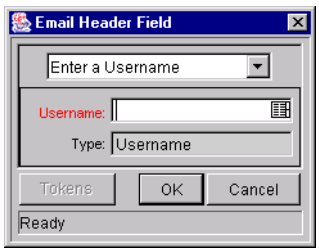
3. In the Template Name field, enter the name of the Notification Template.

4. In the Notification Scope field, select the product area where this Notification Template will be used.

5. Enter a From address.

   a. 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**).

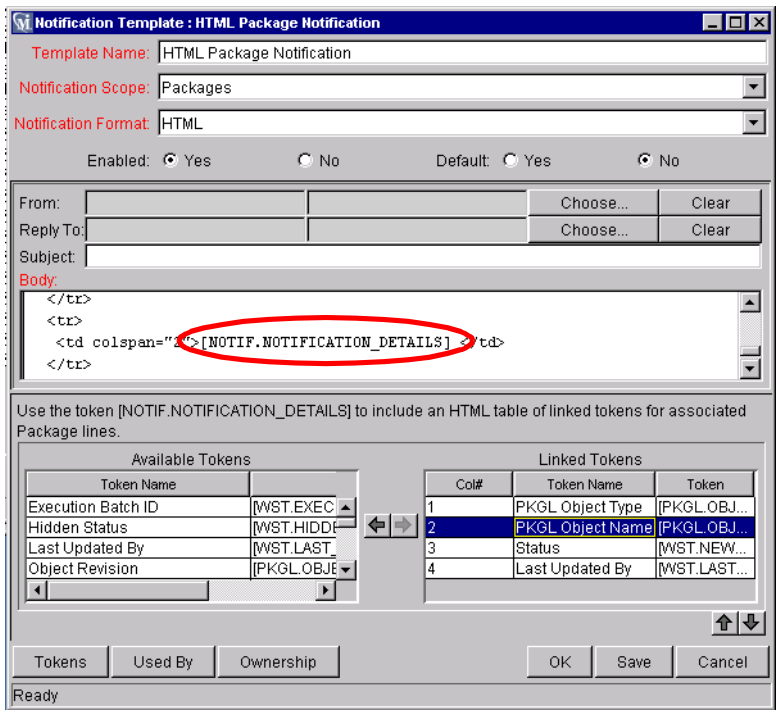
The context-sensitive required field is dynamically updated to gather the necessary information for that category. For instance, if **Enter an Email Address** is selected from the drop down list, then it is necessary to enter an Email Address. If a **User Defined Token** is selected, click **Tokens** 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 type that corresponds with the evaluated Token value.

6. Enter the Notification text in the Body area.

If using **HTML** for the Notification Format, put HTML code in the Body area. HTML Notifications for Mercury Change Management (Packages) should include the Token '[NOTIF.NOTIFICATION_DETAILS]' within the <body> tags to incorporate linked Tokens.



7. Click **OK** to save and close the window.

8. To enter a Reply To address, click **Choose...**

   The Email Header Field window opens. Follow the instructions described in step *5* for entering a From address.

9. Click **Save** to save the new Notification Template definition.

Copying Notification Templates

To copy a Notification Template:

1. Click the **Configuration** screen group and click the **Notification Templates** screen.

   The Notification Template Workbench opens.

2. Enter the search criteria and click **List**.

3. Select the Notification Template to be copied in the **Results** tab of the Notification Templates Workbench.

4. Click **Copy**.

   The Copy Notification Template window opens.

5. Enter the new Template Name.

6. Click **Copy**.

   A question dialog asks if you want to edit the new Notification Template.

7. Click **Yes** to edit or **No** to return to the Notification Template Workbench.

Deleting a Notification Template

To delete a Notification Template:

1. Click the **Configuration** screen group and click the **Notification Templates** screen.

   The Notification Template Workbench opens.

2. Click **List**.

3. Select the Notification Template to be deleted in the **Results** tab of the Notification Template Workbench.

4. Click **Delete**.

   A dialog box appears asking for confirmation that the Notification Template is to be deleted.

5. Click **Yes** to delete the Notification Template.

Notification Templates that are referenced from a Mercury ITG Notification can not be deleted. To delete a Notification Template, first remove these references. Referenced Notification Templates can be disabled.

Configuring Your Dashboard

The Dashboard provides an interface through which the current state of the deployments can quickly be assessed. Personalize the Dashboard to display status information that is most meaningful to any given role. For example, software engineers may only want to see the Packages that they submitted, whereas the IT manager may want to have visibility into each critical Package currently in progress.

Each user can personalize their own Dashboard to display only information relevant to their role. When configuring a deployment system, consider the following configuration topics:

- *Controlling User Access to Portlets*
- *Creating Custom Portlets*
- *Distributing Dashboard Pages*

This section provides an overview of configuring the Dashboard for a deployment process. See *Configuring the Dashboard* for additional details.

Related Topics:

- *Using the Dashboard*
- *Configuring the Dashboard*

Controlling User Access to Portlets

User access to Portlets can be controlled by:

- *Disabling Portlets*
- *Restricting User Access*

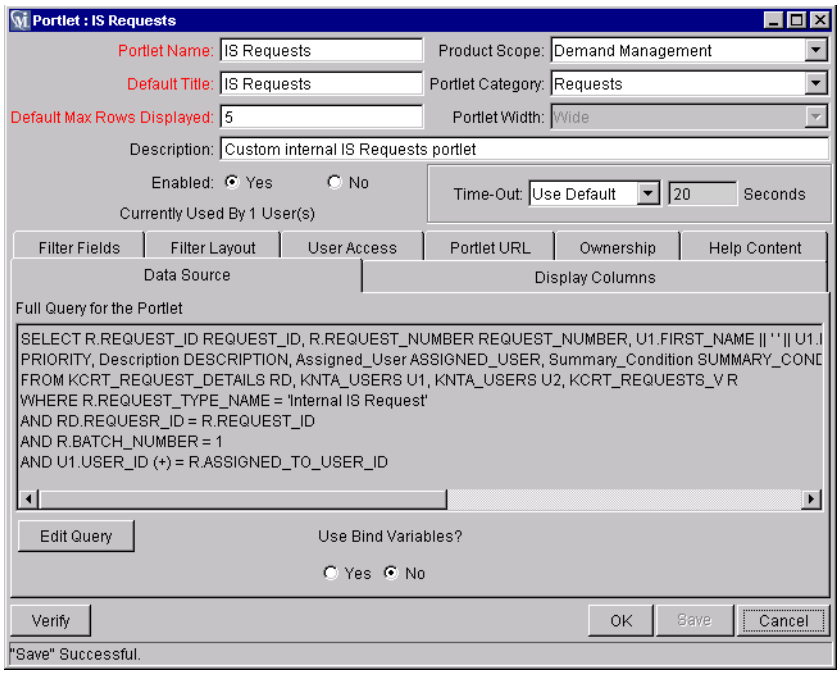
Disabling Portlets

It is possible to disable custom-built Portlets at your site.

To disable a Portlet:

1. Click the **Dashboard** screen group and click the **Portlets** icon.
2. Search for and open the custom Portlet to disable.

   Note that the system Portlets can not be disabled. To control access to these Portlets, restrict user access. See *"Restricting User Access"* on page 235 for details.



3. Click Enabled = **No**.

If there are any users currently using the Portlet on their Dashboard, disabling the Portlet will delete it from their Dashboards.

4. Click **Save**.

Restricting User Access

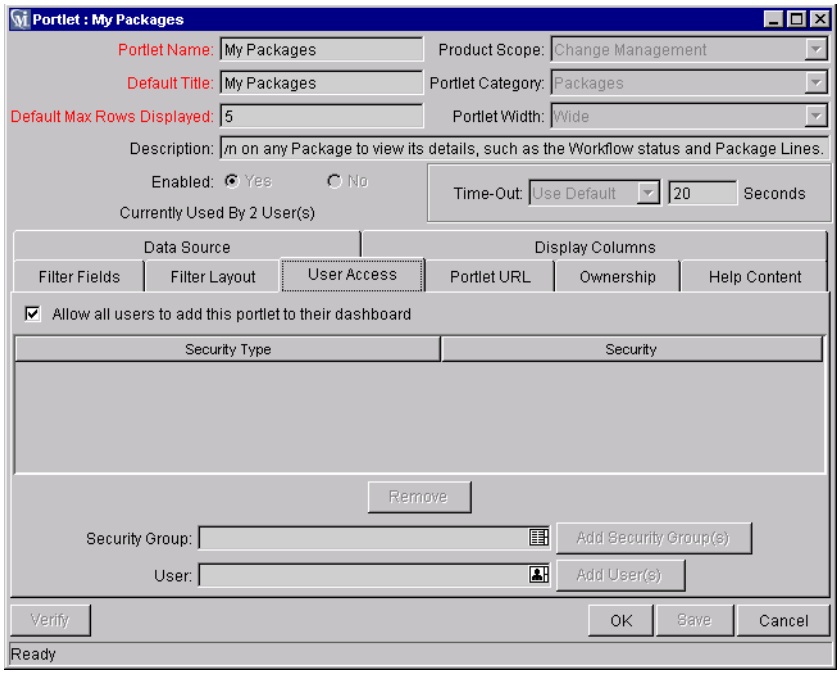
It is possible to control which users can add a Portlet to their Dashboard. For example, it may be desirable to restrict the Package-related Portlets to only members involved in the deployments. Enabling only the Portlets that a specific user needs will make it easier for that user to personalize their Dashboard, because there are less (non-relevant) Portlets to choose from.

To specify which users can use the Portlet on their Dashboard:

1. Click the **Dashboard** screen group and click the **Portlets** icon.

2. Search for and open the Portlet that you would like to configure.

3. Click the **User Access** tab.

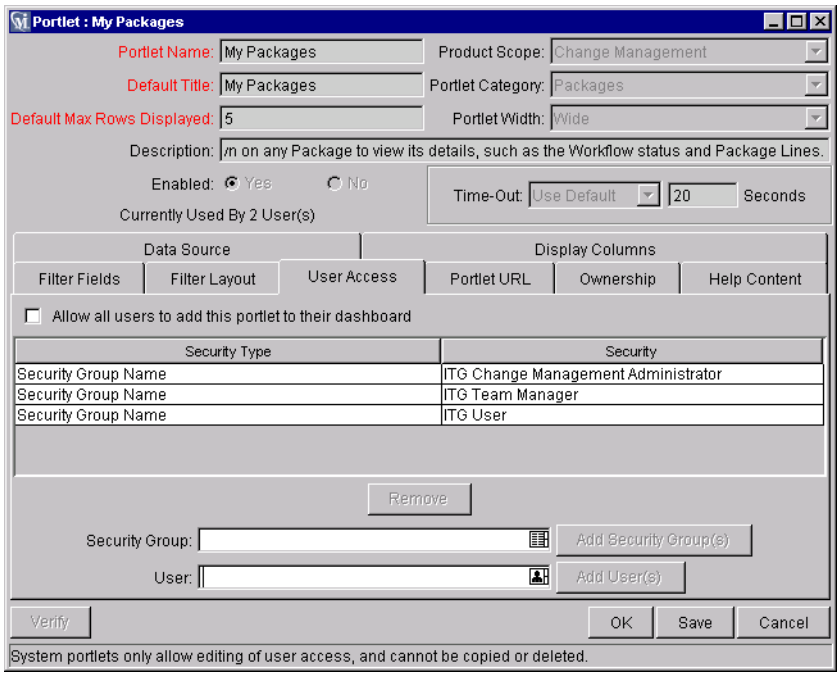


4. Uncheck the Allow all users to add this Portlet to their dashboard field.

   The Security Group and User fields are enabled.

5. Select the desired Security Groups or Users and click the respective **Add** button.

   They are added to the **User Access** tab.



6. Click **Save**.

Restrict access by specifying multiple Security Groups and Users for each Portlet. Only members of the specified Security Group or the specified users can add this Portlet to their Dashboard.

It is possible to restrict user access for both custom and system Portlets.

Creating Custom Portlets

Portlets are visual displays that act as windows into your Change Management. While Mercury ITG system Portlets (provided at the time of installation) are personalizable by end-users and provide wide access to your data, custom Portlets can also be created to access additional information in the system or in other databases. These custom Portlets behave the same as the system Portlets, using filter fields to limit the displayed data. It is also possible to create textual or graphical Portlets.

Since custom Portlets are data-driven entities and require extracting information stored in the database, knowledge of SQL is required for users who wish to create or configure Portlets.

For detailed instructions on creating custom Portlets, see *Configuring the Dashboard*.

Before creating custom Portlets, consider using one of Mercury ITG's system Portlets. Review the business and data presentation Portlet requirements and compare against the system Portlets. See *Configuring the Dashboard* for a complete list of Mercury IT Governance Portlets.

Distributing Dashboard Pages

To ensure that all participants in the deployment process have visibility into the most relevant information, configure and distribute Dashboard pages for those users. This section discusses the following topics:

- *Publishing Required Dashboard Pages*
- *Distributing Optional Dashboard Pages*
- *Creating a Default Dashboard*

For instructions on implementing these functions, see *Configuring the Dashboard*.

Publishing Required Dashboard Pages

Publishing involves the dissemination of required Dashboard pages and Portlets to one or many user Dashboards at one time. Publishing is a regimented approach to disseminating Dashboard pages and Portlets. The Dashboard pages and Portlets of a published Module can not be edited or removed by the owner of the Dashboard.

Distributing Optional Dashboard Pages

Distributing involves the dissemination of optional Dashboard pages and Portlets to one or many Dashboards at one time. Distributing provides a more flexible approach to disseminating Portlets and Dashboard pages than publishing. Once distributed, the Portlets and Dashboard pages can be edited by the user.

Creating a Default Dashboard

The Default Dashboard allows first-time users quick and easy integration of the Dashboard into their business processes. The Default Dashboard can be published or distributed. Published Default Dashboard pages are disseminated to Dashboards as read-only pages. Distributed Default Dashboard pages can be edited by the owner of the Dashboard.

Users with the Dashboard, Configure Module and Dashboard, Distribute Module Access Grants can configure and distribute the Default Dashboard to all users.

Configuring Reports

Mercury Change Management includes a pre-defined set of HTML-based reports accessed through a Web browser. The reports allow users to view the current detailed status of their Mercury IT Governance data at any point in time.

Change Management also provides a Reporting Meta Layer, which allows users to build their own custom reports using third-party reporting tools.

For a full list of available reports and information on configuring them, see *Reports Guide and Reference*.

Chapter 11 Rolling Out Your Deployment Process

This chapter provides checklists and instructions for rolling out the deployment process to the users. This includes information on using Migrators, enabling user access, and training the users to use the system.

This chapter covers the following topics:

Test the Deployment System - Checklists

This section provides a series of high-level checklists to help validate the system before rolling it out to users.

The following topics are discussed:

- *Cross-Entity Checklist*

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

	Entity Defined?	Notes
	Workflow	One or more Workflows that will be used to process the Packages (deploy your objects) must be available. See the following sections for details on Workflow construction: • *"Mapping your Process into a Workflow"* on page 65 • *"Advanced Workflow Topics"* on page 255
	Object Types	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 sections for details on Object Type and Command construction: • *"Constructing the Object Type"* on page 123 • *"Validations"* on page 287 • *Commands and Tokens Guide and Reference*
	Environments	Define the source and destination Environments for the objects being deployed. See the following sections for details on Environment definition: • *"Defining your Environments"* on page 153
	Security Groups / User Access	Define the Security Groups used to control different aspects of the deployment process: Package creation, Package processing, and deployment system configuration. See the following sections for details on Security Group and User Participant definition: • *"Integrating Participants into Your Deployment System"* on page 187 • *Security Model Guide and Reference* Work with the instance administrator to configure User and Security Group definitions.

Table 11-1. General Configuration Checklist [continued]

	Entity Defined?	Notes
	Dashboard / Portlets	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 sections for details on Portlet construction and default Dashboard creation: • *"Setting Up Communication Paths"* on page 207 • *Configuring the Dashboard*

Workflow Checklist

Table 11-2. Workflow configuration checklist

	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 section for details: • *"Building the Workflow Skeleton - Overview"* on page 65
	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 sections for details: • *"Create the Required Step Source"* on page 68
	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 sections for details: • *"Creating a Decision Type Step"* on page 73 • *"Create an Execution Type Step"* on page 77
	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 sections for details: • *"Adding Transitions Between Steps"* on page 109
	Manual transitions are set	Ensure that the step has a transition path for each available decision result. See the following sections for details: • *"Adding Transitions Between Steps"* on page 109
	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 sections for details: • *"Create an Execution Type Step"* on page 77

Table 11-2. Workflow configuration checklist [continued]

	Workflow Check Item	Notes
	Notifications are set on appropriate Workflow Steps	You need to configure notifications to be sent at specific points in the process. See the following sections for details: • *"Adding Notifications to Workflow Steps"* on page 208
	Includes a Close step.	The process should conclude with a "Closed" Package at all exit points.
	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 *Table 11-3*.

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.'

Table 11-3. Workflow Logical Guidelines [continued]

Guideline	Returns	Reason
An immediate Execution Step should not have a transition to itself on 'Success' or 'Failure.'	Error	The Workflow could loop indefinitely.
'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

	Object Type Check Item	Notes
	Fields defined	Fields are required to define the object. Ensure that the correct parameters describe the object to be deployed. See the following sections for details: • *"Creating Object Type Fields"* on page 125 • *"Validations"* on page 287
	Commands defined	All commands needed to process and deploy the object have been constructed. See the following sections for details: • *"Creating Object Type Commands"* on page 144 • *Commands and Tokens Guide and Reference*
	Conditions set in commands	Conditions to steps within the command that dictate when the specific command steps run have been added. See the following sections for details: • *Commands and Tokens Guide and Reference*

Environments Checklist

Table 11-5. Environment definition checklist

	Environment Check Item	Notes
	Define the “source” Environment	See the following sections for details: • *“Defining Environments”* on page 154
	Define the “destination” Environment	See the following sections for details: • *“Defining Environments”* on page 154
	Select the appropriate connection protocol	See the following sections for details: • *“Selecting the Environment’s Connection Protocol”* on page 157
	Select the appropriate transfer protocols	See the following sections for details: • *“Selecting the Environment’s Transfer Protocol”* on page 158
	Define Environment Groups	See the following sections for details: • *“Creating Environment Groups”* on page 166
	Verify the Environment Definitions	See the following sections for details: • *“Environment Maintenance and Utilities”* on page 180

Security / User Access Checklist

Table 11-6. Security / User Access Configuration Checklist

	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 sections for details: • *"Establishing Security Groups"* on page 188 • *Security Model Guide and Reference*
	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 sections for details: • *"Establishing Security Groups"* on page 188 • *Security Model Guide and Reference*
	Set security on Package Creation	All available options for restricting who can create and submit Packages have been set. See the following sections for details: • *"Setting Package Creation Security"* on page 194
	Set security on Package processing	All available options for restricting who can process Packages have been set. See the following sections for details: • *"Setting Package Processing Security"* on page 197
	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, etc. See the following sections for details: • *"Setting Configuration Security"* on page 201

Dashboard / Portlet Checklist

Table 11-7. Dashboard / Portlet Configuration Checklist

	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 sections for details: • *"Configuring Your Dashboard"* on page 233
	Enable Portlets for use on the Dashboard	See the following sections for details: • *Configuring the Dashboard* • *Security Model Guide and Reference*
	Specify which users can add use certain Portlets	See the following sections for details: • *Configuring the Dashboard* • *Security Model Guide and Reference*
	Create a default Dashboard or distribute a Dashboard to your users.	See the following sections for details: • *Configuring the Dashboard*

Cross-Entity Checklist

Table 11-8. Cross Entity Configuration Checklist

	Entities	Configuration Considerations
	Workflow and Object Type	The following items should be coordinated between the Workflow and Object Type: • 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	The following items should be coordinated between the Workflow and Environments: • Specify the source and destination Environments (or Environment Groups) on the appropriate Workflow Execution Steps.
	Workflow and Security Groups	The following items should be coordinated between the Workflow and Security Groups: • 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	The following items should be coordinated between the Object Types and Environments: • Specify any Environment overrides in the Object Type commands.
	Security Groups and other entities (Object Types, Environments, etc.)	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.

Migrate Your Configuration Data into Production

"Developing Your Configurations (Using Migrators)" on page 23 discusses using multiple Mercury Change Management instances to configure and deploy your configuration data. 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

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. For detailed instructions on using the migrators, see *Migrators Guide and Reference*.

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.

Enable Entities and User Access

Each entity used in the deployment process includes an Enabled parameter. This parameter needs to 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 the educating the users on the following activities:

- Basic product use: creating, processing, and reporting on Packages
- Process-specific training:
  Ensure that each user understands the deployment process. Consider holding a formal roll-out meeting or publish 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.

Additional Resources for Education and Roll-Out

Consider using the following resources to assist in your education and roll-out activities:

Mercury Education and Online courses

Mercury Interactive provides a complete training curriculum to help you achieve optimal results using the Mercury IT Governance Center. For more information, visit the Education Services Web site at *http://www.merc-training.com/main/ITG*.

Online Help

The Online Help provides details on creating and processing Requests, Packages, Projects and Tasks. This Help system features information on using Mercury Change Management's main HTML interface and the Dashboard.

Power Users can access other product documentation from the Workbench. From the **Help** menu, select **Mercury IT Governance Library**. A single page opens and provides access to all user, configuration and administration documentation.

Appendix A

Advanced Workflow Topics

This chapter provides instructions for implementing advanced Workflow features. This includes configuring Subworkflows, integrating Workflows used in different Mercury IT Governance products, and using Workflow parameters.

This appendix covers the following topics:

Using Subworkflows

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 sub-processes.

A Subworkflow can be selected from the Workflow Step Sources window and dragged onto the **Layout** tab. When the Package, Request, or Release Distribution 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 Subworkflow 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 can also be generated by copying and renaming an existing Subworkflow.

This section covers the following topics:

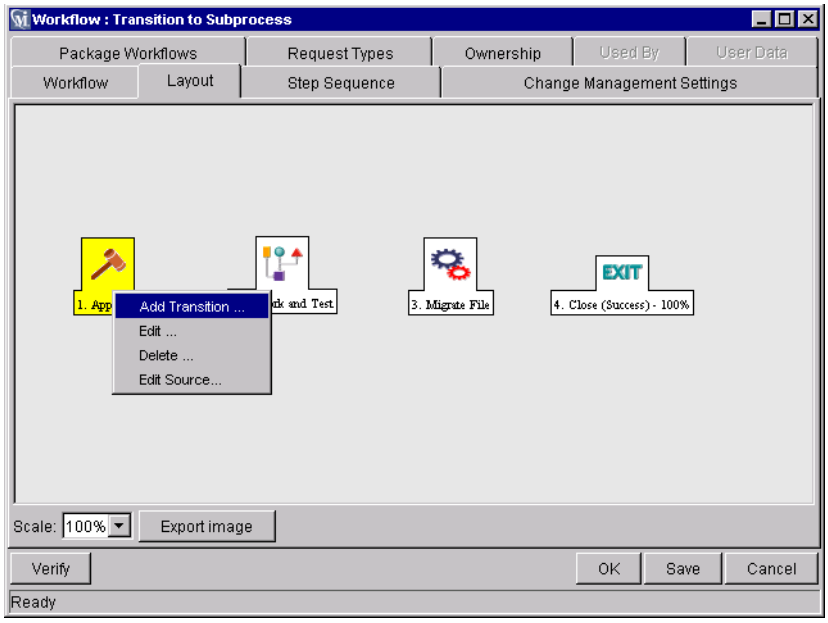
Transitioning to a Subworkflow

A transition to a Subworkflow Step is made in the same way as a transition to any other Workflow Step (Execution, Decision or Condition).

To transition to a Subworkflow:

1. Open a Workflow and click the **Layout** tab.
2. Right-click the source Workflow Step and select **Add Transition** from the pop-up menu.

   This creates an arrow from the source Workflow Step.

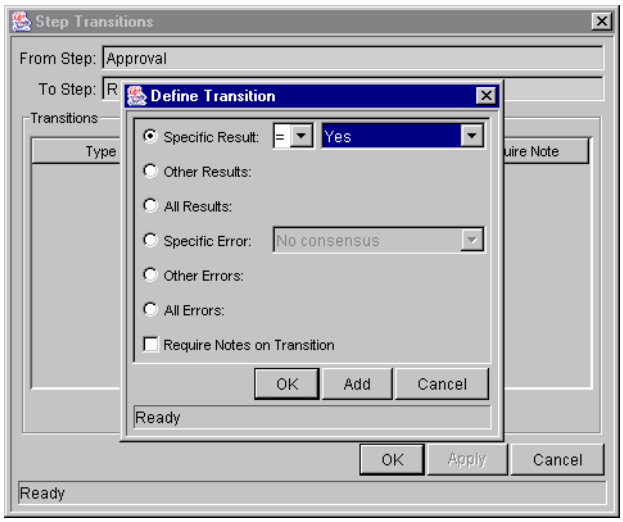


3. Drag the arrow to the destination Step and left-click.

   The Step Transitions window opens.

4. Click **New**.

   The Define Transition window opens.



5. Define the desired transaction result by:

   a. Clicking one of the radio buttons for results (Specific Result, Other Results, etc.).

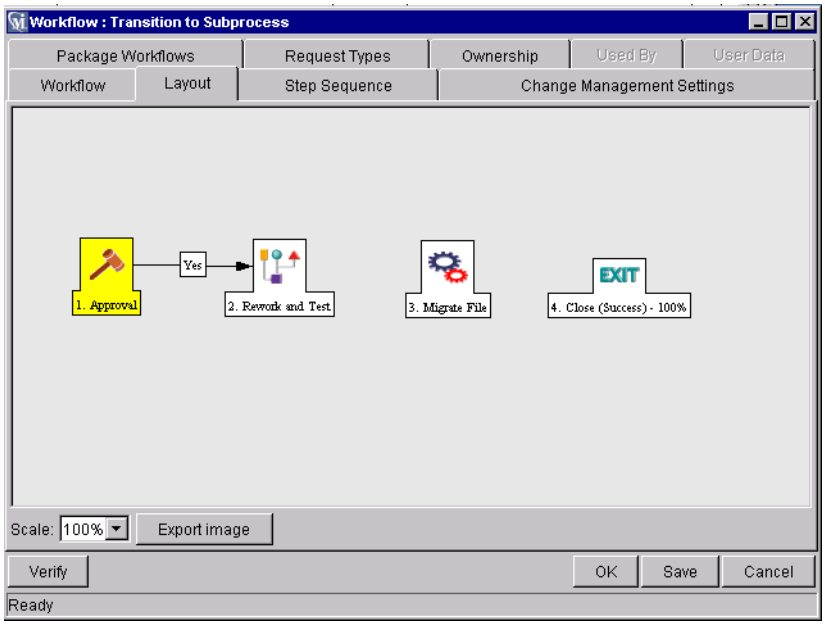
   b. Selecting **=** or **!=** (does not equal) from the drop down list.

   c. Selecting **Yes** or **No** from the drop down list.

   If the Workflow Step results in this value, the Request or Package proceeds along this path.

6. Click **OK** to close the Define Transition window.

7. Click **OK** to finalize the step transition definition.

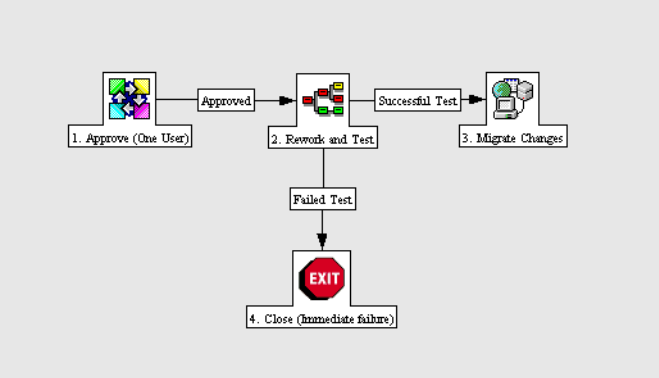


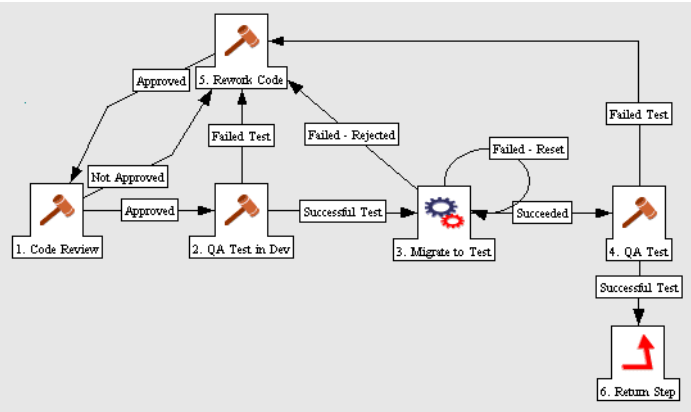
The transition is graphically represented by an arrow between the two steps. The Package Line or Request proceeds to the First Step designated in the Subworkflow definition.

Transitioning From a Subworkflow

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.





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.

Package - Request Workflow Integration

Request and Package Workflows 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 that the Package will not continue until the Request closes.

Two built-in Workflow Events facilitate this cross-product Workflow integration. These steps are **wf_jump** and **wf_receive**. Jump (**wf_jump**) and Receive (**wf_receive**) steps are used at the points of interaction between Workflows. Each Jump step must be coupled with a Receive step. Workflows can communicate through these Jump and Receive 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 in Mercury Change Management.

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 A-1*.

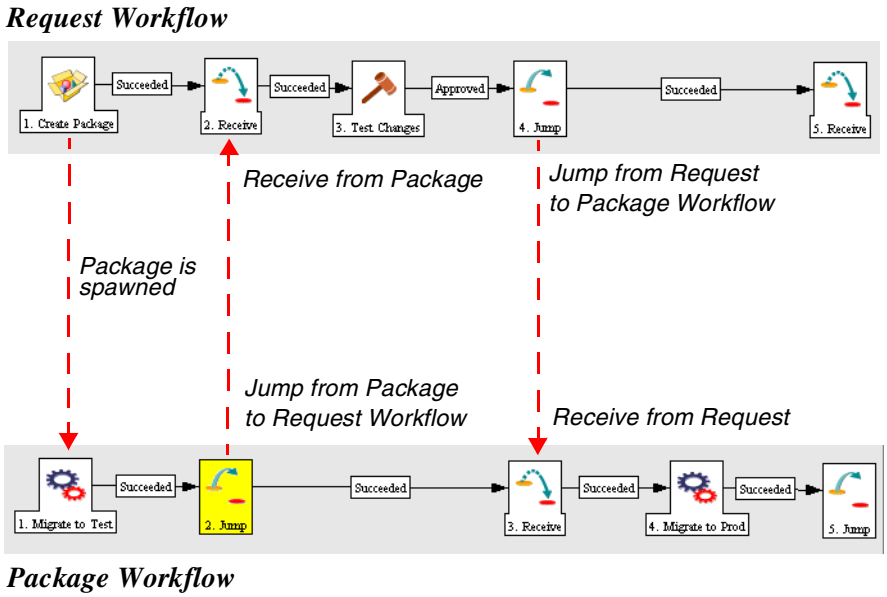


Figure A-1 Jump/Receive Workflow Steps

The Jump and Receive pair must be carefully coordinated. Each Jump step must have an associated Receive step, linked together by a common Jump/Receive Step Label defined in the Workflow Step window. The transition values for entering into and exiting the Jump and Receive steps must also be coordinated.

This section details the process for setting up a successful Request - Package Workflow integration.

To establish communication between Requests and Packages:

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 step. The selected Jump/Receive Step Label must match in the paired Jump and Receive Workflow Step windows. See *"Setting Up the 'WF - Jump/Receive Step Labels' Validation"* on page 262.

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

   See *"Generating a Jump Step Source"* on page 264.

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

   See *"Generating a Receive Step Source"* on page 266.

4. Verify that both the Jump and Receive steps specify the same Jump/Receive Step Label.

   See *"Including the Jump/Receive pair in Workflows"* on page 268.

5. Verify that the transitions exiting the Jump and Receive steps match the possible values entering the Jump step.

Setting Up the 'WF - Jump/Receive Step Labels' Validation

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

1. Click the **Configuration** screen group and click the **Validations** icon.

   The Validation Workbench opens.

2. On the **Query** tab, enter **WF - Jump/Receive Step Labels** in the Validation Name field.

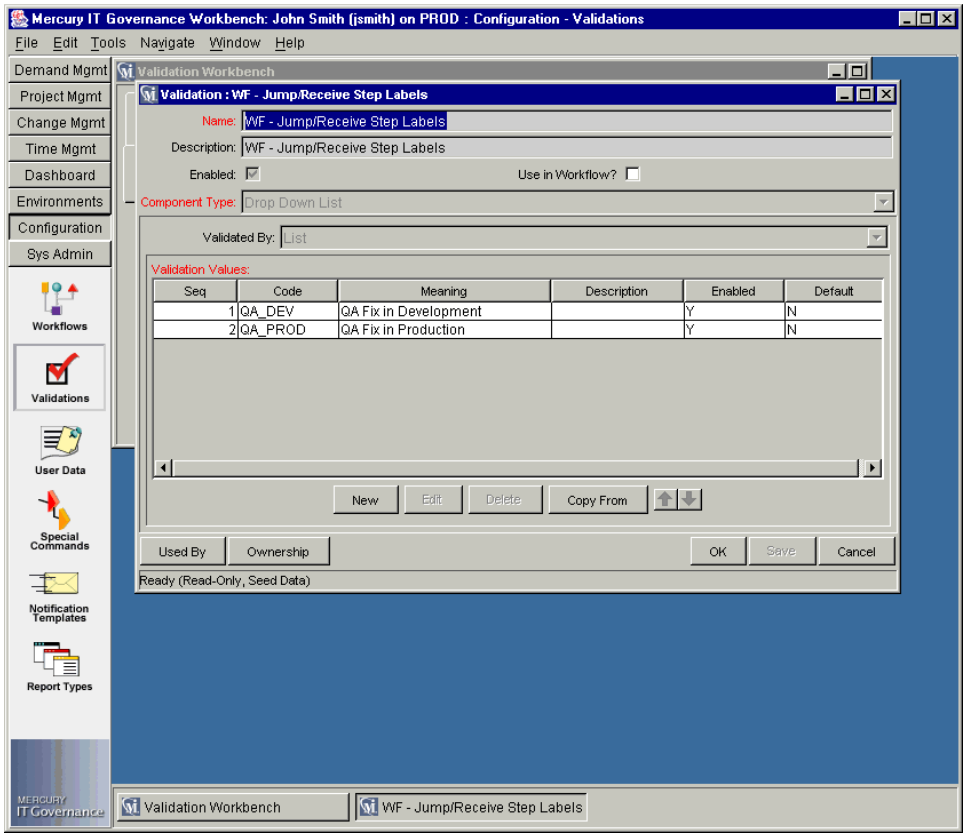
3. Click **List**.

4. Click the **Results** tab.

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

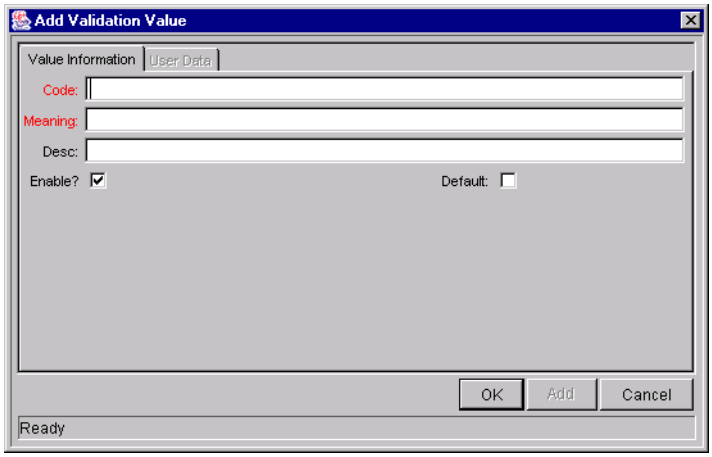
5. Click **Open**.

   The Validation window opens.



6. Click **New** to define a new Validation Value that is used to link two Workflows together.

   The Add Validation Value window opens.



7. Enter the desired Code, Meaning and Description in the appropriate fields.
8. Click **OK** to close the Add Validation Value window.
9. Click **Ownership** to select which Ownership Groups will have the ability to edit this Validation.
10. Click **OK** to close the Validation window.

The new Validation Value is now included in the Jump/Receive Step Label drop down list in the Workflow Step window.

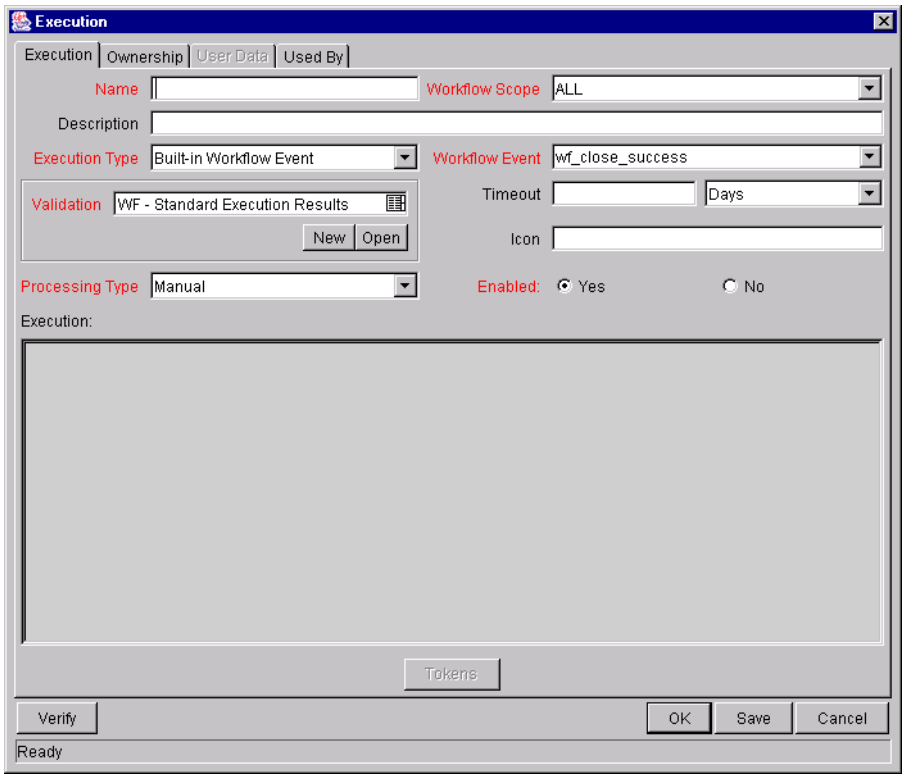
Generating a Jump Step Source

To create a Jump step using the wf_jump Built-in Workflow Event:

1. Click the **Configuration** screen group and click the **Workflows** icon.

   The Workflow Workbench and Workflow Step Sources window open.
2. Select the Workflow Step Sources window.
3. Select the Executions folder.
4. Click **New**.

   The Execution window opens.



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

   Package Level Subworkflows can not include jump and receive steps.

6. From the Execution Type drop down list, select **Built-in Workflow Event**.

7. From the Workflow Event drop down list, select **wf_jump**.

8. From the Validation drop down list, select or create a Validation which will be used to transition out of this Workflow Step.

Note

The Validation values exiting the Jump step must match the possible Validation values entering the Jump step.

9. Fill in any other required or optional information in the Execution window (such as Name, Description or Processing Type).

10. Click the **Ownership** tab to select which Ownership Groups will have the ability to edit this Execution step.

11. 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.

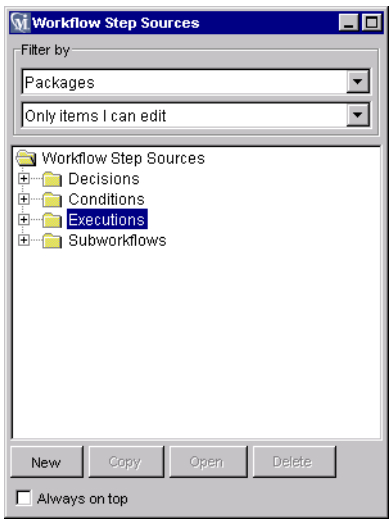
Generating a Receive Step Source

To create a Receive step using the wf_receive Built-in Workflow Event:

1. Click the **Configuration** screen group and click the **Workflows** icon.

   The Workflow Workbench and Workflow Step Sources window open.

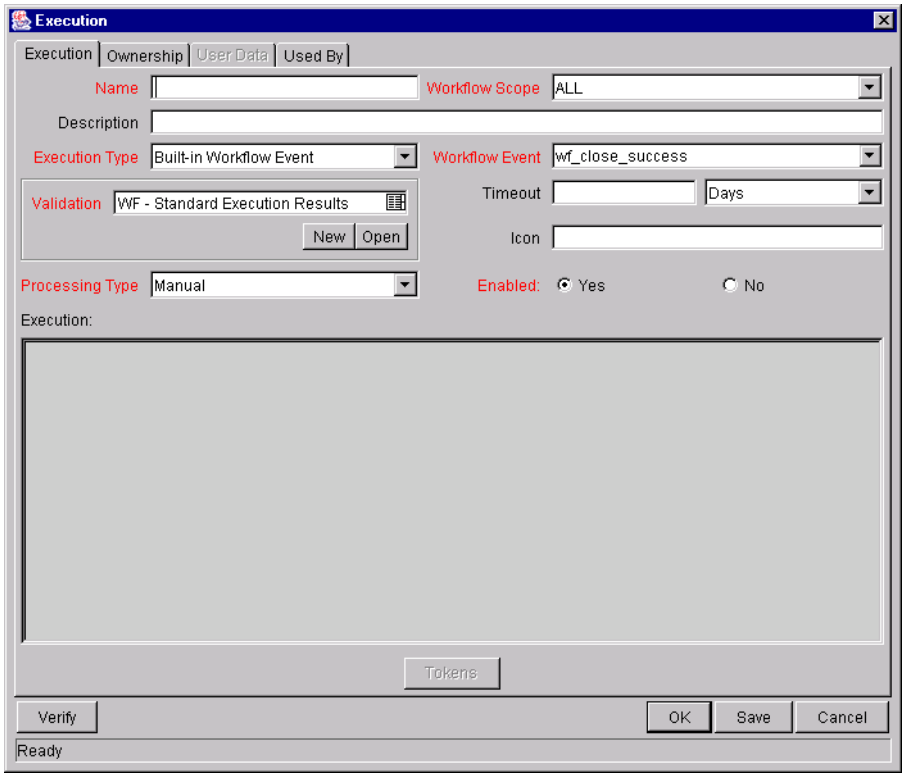
2. Select the Workflow Step Sources window.



3. Select the Executions folder.

4. Click **New**.

   The Execution window opens.



5. From the Workflow Scope drop down list, select either **Packages** or **Requests** depending on the desired application of the Workflow.

6. From the Execution Type drop down list, select **Built-in Workflow Event**.

7. From the Workflow Event drop down list, select **wf_receive**.

8. Select or create a Validation which will be used to transition out of this Workflow Step.

Note

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

9. Fill in any other required or optional information (such as Name, Description or Processing Type).

10. Click the **Ownership** tab to select which Ownership Groups will have the ability to edit this Execution step.

11. Click **OK**.

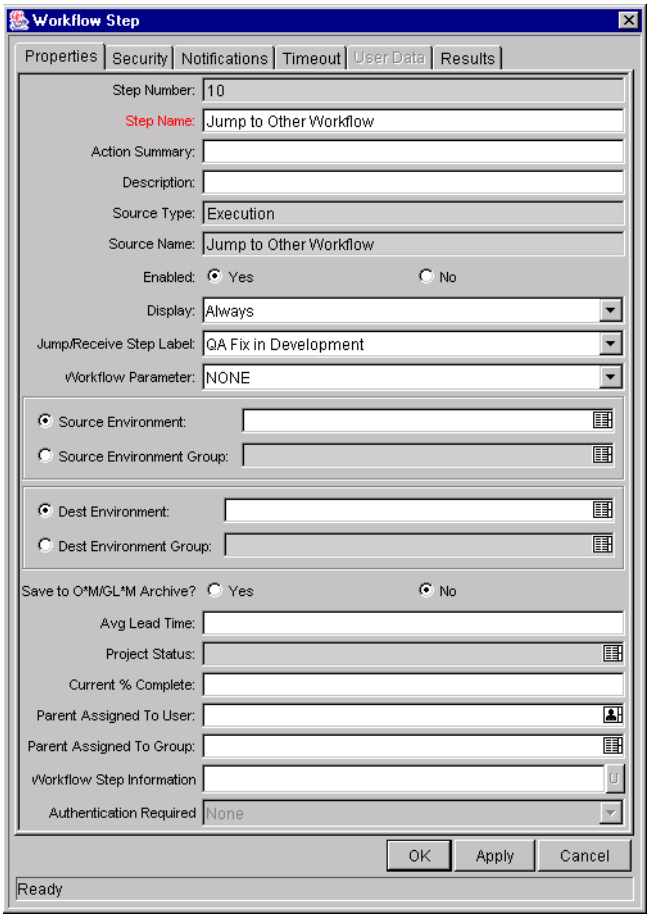
    The Workflow Step is added to the Workflow Step Sources window.

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 the Jump/Receive pair in Workflows

1. Drag either the Jump or Receive step from the Workflow Step Sources window into the Workflow's **Layout** tab.

   The Workflow Step window opens.



2. Select an item from the Jump/Receive Step Label drop down list.

   This item must be the same for a paired Jump/Receive 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.

3. Enter any additional Workflow Step information.
4. Click **OK.**
5. Repeat the above process for the other paired Workflow Step (Jump or Receive), depending on which one was configured first

Using Condition Steps

It is possible to perform complex routing based on the status of multiple Workflow Steps using Condition steps. *Table 0-1* lists the five available Condition steps:

Table A-1. Workflow Condition Steps

Condition	Description
AND	Waits until all parallel Workflow branches of a single Package Line or Request have reached this point before continuing processing.
FIRST LINE	Only used for Package Workflows and has a validation of True/False. Evaluates to true for the first Package Line to reach the condition. Used for multiple Package Lines.
LAST LINE	Only used for Package Workflows. Has a validation of True/False. Evaluates to true for the last Package Line to reach the condition. Used for multiple Package Lines.
OR	Allows the first branch of the Workflow that reaches this point to continue and terminates the other branches upon their arrival at this point so only one will continue.

Table A-1. Workflow Condition Steps [continued]

Condition	Description
SYNC	Only used for Package Workflows. This condition halts further processing of each Package Line until all Lines of the Package reach this step. This is used to synchronize all the Lines of a Package at one point. Used for multiple Package Lines.

The following sections provide examples of the following:

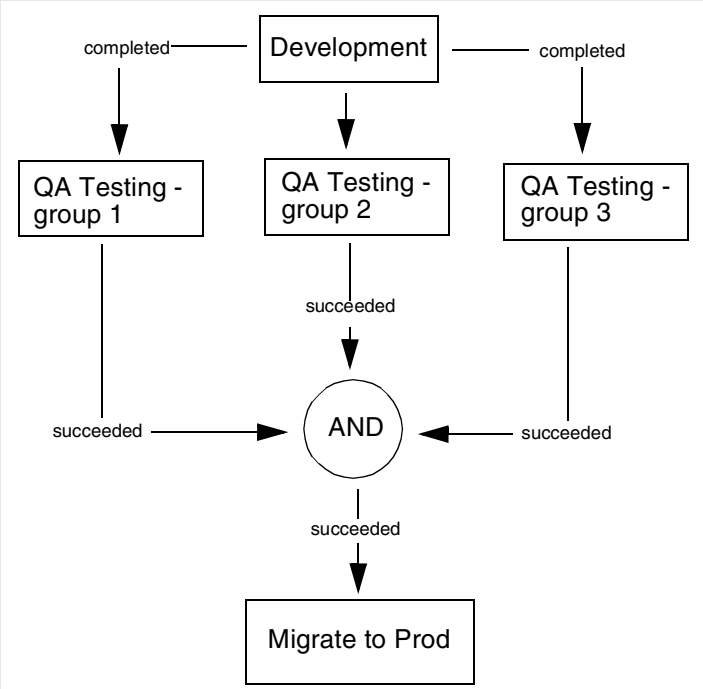
- *AND*
- *OR*
- *SYNC*
- *FIRST LINE*
- *LAST LINE*

AND

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

Example

The AND step becomes successful only if ‘QA Testing - group 1,’ ‘QA Testing - group 2’ and ‘QA Testing - group 3’ are successful. At that point, the following step ‘Migrate to Prod’ becomes eligible.

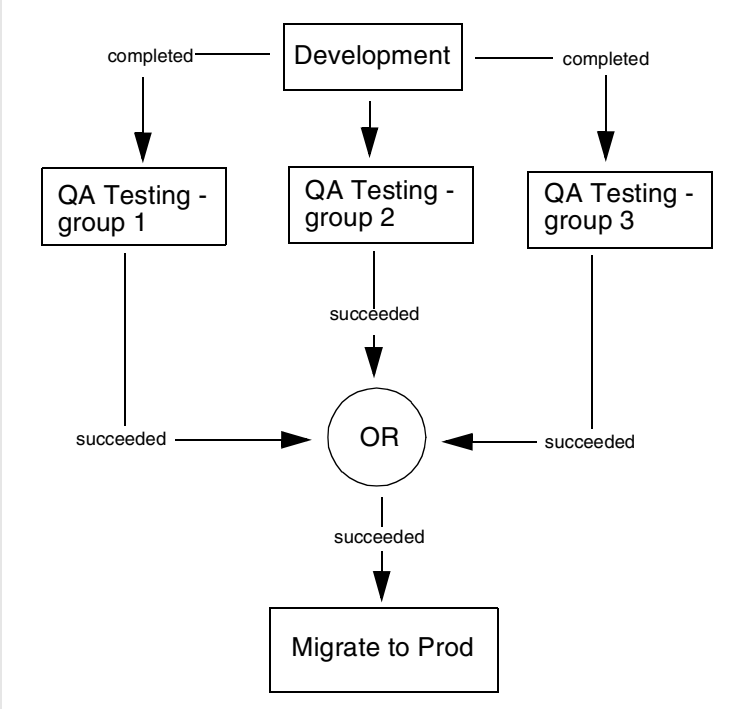


OR

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

Example

The OR step becomes successful if any one of ‘QA Testing - group 1,’ ‘QA Testing - group 2’ and ‘QA Testing - group 3’ is successful. At that point, the following step ‘Migrate to Prod’ becomes eligible.



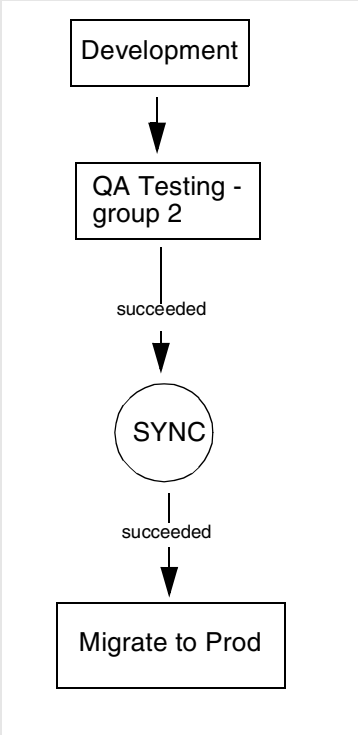
SYNC

A SYNC step is valid only for Mercury Change Management Packages. A SYNC 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.

Example

Consider the business process outlined in the following flow chart. According to the flow chart, when ‘QA Testing’ is successful for all Package Lines, SYNC becomes successful and the next step, ‘Migrate to Prod’ becomes eligible.

This business process could be part of a software development life cycle. Consider a case where three Java files are being processed on three respective Package Lines in a single Package. By including a SYNC step, even if the first two Java files pass ‘QA Testing,’ they must wait for the third Java file to succeed ‘QA Testing’ before ‘Migrate to Prod’ becomes eligible for any of these Package Lines.



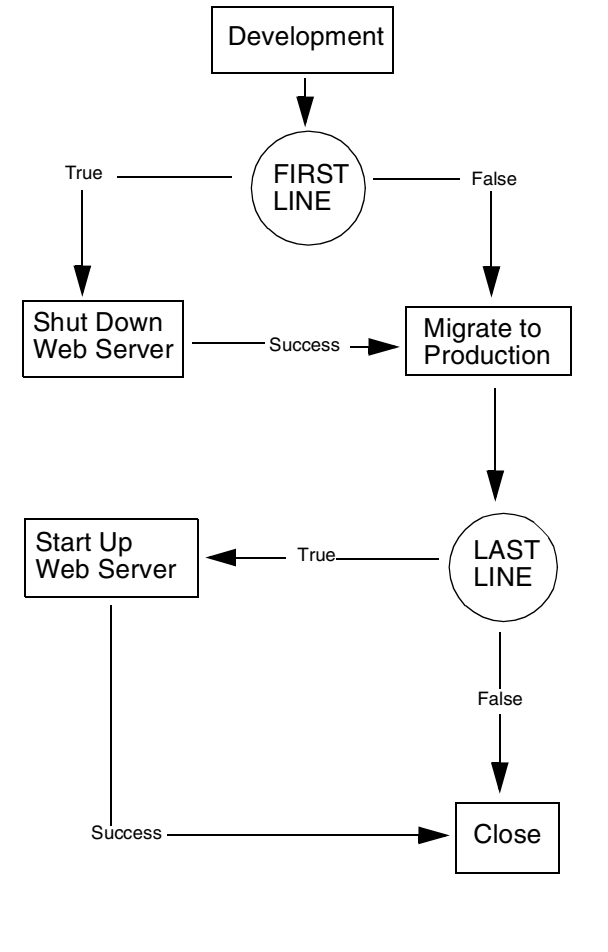
FIRST LINE

A FIRST LINE step is valid only for Mercury Change Management Packages. Only the first line to reach the Condition step takes the ‘True’ transition. All successive lines take the ‘False’ transition.

Example

Consider the business process outlined by the following flow chart. 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 step, 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.



LAST LINE

A LAST LINE step is valid only for Mercury Change Management. Only the last active line to reach the Condition step takes the 'True' transition. All previous lines take the 'False' transition. See the example of a LAST LINE step shown in the previous flow chart.

Setting the Reopen Step for Request Workflows

Closed Requests can be re-opened by users with the proper access grants. A re-opened Request begins at the pre-defined Reopen Step in its Workflow, and begins processing normally.

The Reopen Step is defined from the Workflow window.

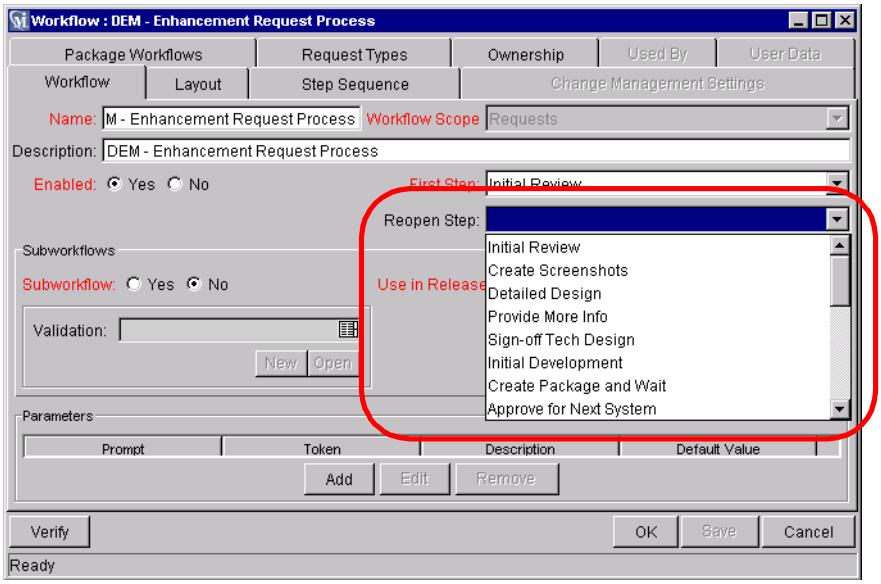


Figure 0-1 Workflow Window Reopen Step Drop Down

To specify the Reopen Step for the Workflow, select the desired step from the Reopen Step drop down list.

Modifying Workflows in Use

Workflows can be modified while they are going through their Workflow steps in a Package Line or Request that 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.

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 0-1*.

Table 0-1. 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, and Distributions.

Changes to a Workflow can have undesirable effects on Requests or Packages currently in progress and are using that Workflow.

The information included here also applies when migrating Workflows between installations (instances) of Mercury IT Governance products.

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 Step. While no one solution covers all situations, the following sections describe possible solutions for common problems when modifying Workflows:

- *Copying and Testing the Workflow*
- *Moving Requests Out of a Step*
- *Disabling a Workflow Step*
- *Setting Up Execution Steps*
- *Verifying Workflow Logic*

Copying and Testing the Workflow

To modify a Workflow that is being used, make a copy of the original Workflow in a Development environment. Then modify the copied version of the Workflow. Test the copied version of the Workflow to make sure it works correctly.

After verifying that the modified Workflow functions as it is supposed to, make the same changes to the original Workflow and move it through the same cycle of Development -> Test -> Production environments.

Moving Requests Out of a Step

If the Requests 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 have flowed out of the step, delete the transition again.

To determine when the Requests have flowed out of the 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.

Disabling a Workflow Step

As mentioned in *Table 0-1*, 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 the Workflow

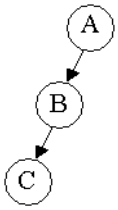
When disabling a Workflow step that is currently 'Eligible' for user action, the Requests or Package Lines in that step will become stuck. Since the step is now disabled, the user cannot take action on it and will not be able to progress any further through the Workflow.

To determine which steps are currently Eligible, remove the incoming transition to the step that will be deleted and run the Packages Pending Report in Mercury Change Management or the Workflow Detail Report in Mercury Demand Management. The reports will indicate if the step to be deleted is 'Eligible' for action by Package Lines or Requests.

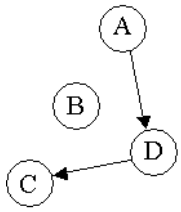
The outgoing transition to be deleted is still intact, so the eligible Package Lines and Requests will eventually be acted upon and flow out of the Workflow step.

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 below.



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 the sequence shown below.



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.

Setting Up Execution Steps

When setting up Execution steps in a Workflow Step, be sure to include Workflow Events 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.

Modifying Workflow Step Security – Performance Consideration

Updating an existing Workflow's step security with a specific configuration can impact system performance. When adding dynamic security to a step (for example, based on a Standard or User Defined Token) in the Workflow Step window on the **Layout** tab, product database tables are updated to handle this new configuration. Due to the scope of these database changes, the Database Statistics need to be re-run on your database. Instructions for this operation are included in the *System Administration Guide*. Contact the System 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.

Verifying Workflow Logic

A Workflow can also become stuck if the logic behind it is faulty. Plan the steps of the Workflow process carefully before actually defining it. After configuring the Workflow, click the **Verify** button in the Workflow window to ensure that the logic of the Workflow is correct. Any mistakes in the Workflow's logic will be highlighted.

Using Workflow Parameters

Use Workflow parameters to store the result of a workflow step. This value can then be used later to define a transition.

This section covers the following topics:

- *Creating a Workflow Parameter*
- *Example: Building a Loop Counter Using Workflow Parameters*

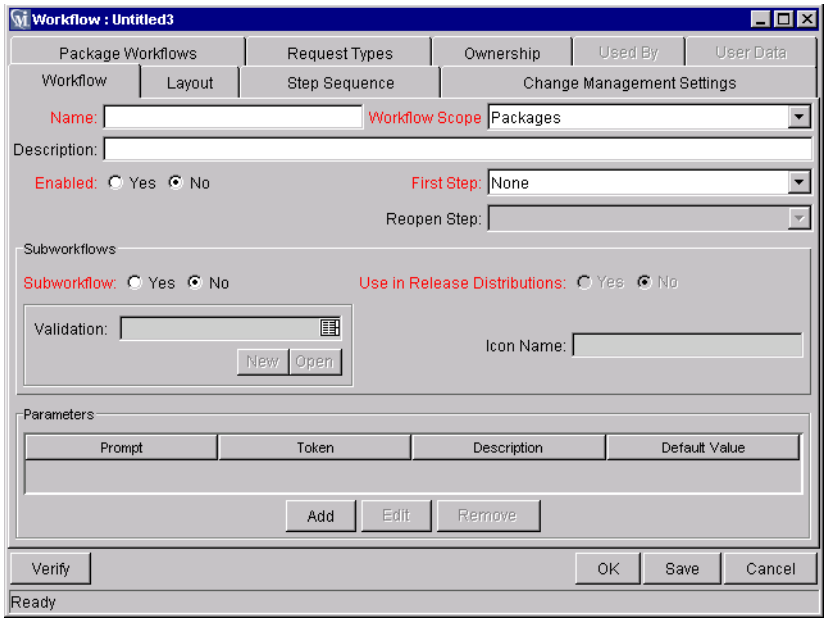
Workflow Parameters:

- Can be referenced using the WFI.P Token prefix.
- Can be used in PL/SQL and SQL Workflow Step executions.

Creating a Workflow Parameter

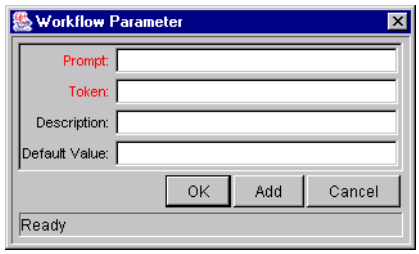
To create a Workflow parameter:

1. From the Workflow Workbench, query and open the Workflow to be modified.



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

   The Workflow Parameter window opens.



3. Enter information in the required fields.

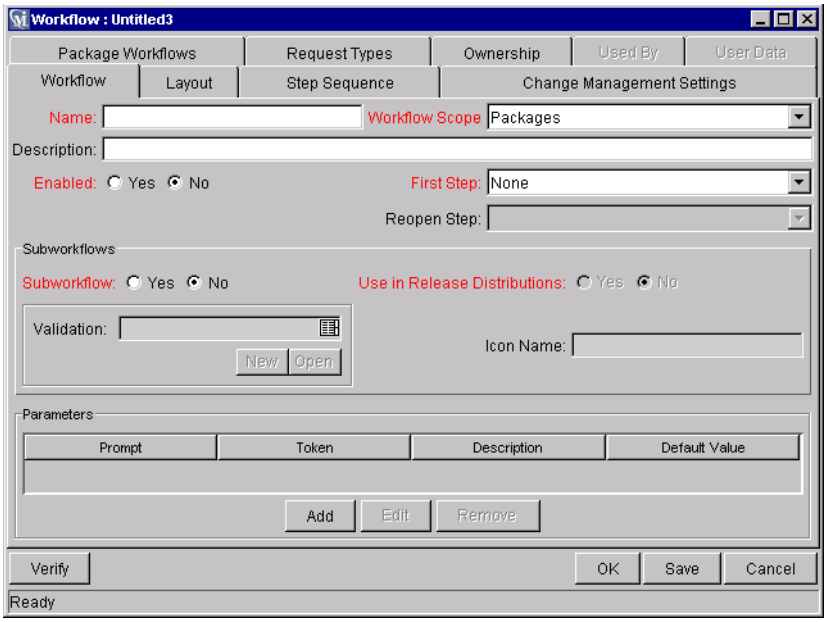
4. Click **OK.**

Example: Building a Loop Counter Using Workflow Parameters

Workflow parameters can be used to generate a counter for the number of times a Workflow Step is in a certain state.

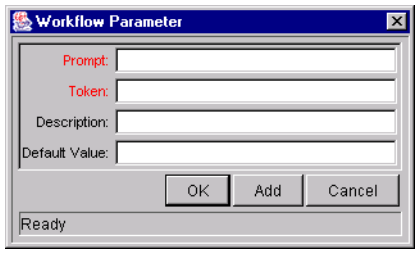
To build a loop counter using Workflow parameters:

1. From the Workflow Workbench, open the Workflow to which the loop counter is to be added.



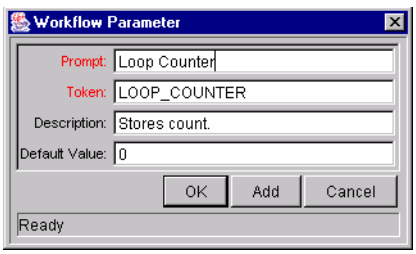
2. In the **Workflow** tab, click **Add**.

   The Workflow Parameter window opens.



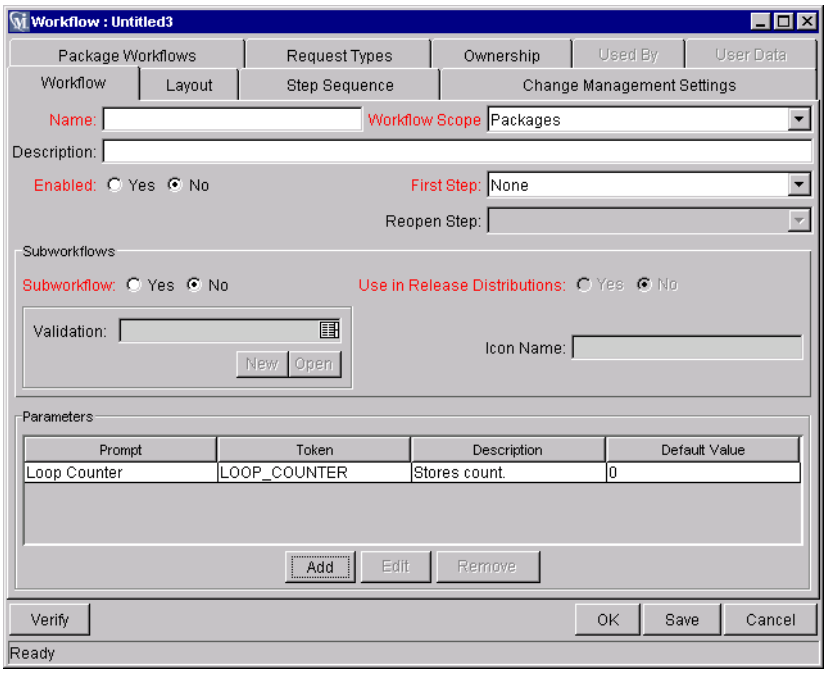
3. Generate the Workflow parameter by entering information in the fields of the Workflow Parameter window.

   In this example, the parameter is named **LOOP_COUNTER**.



4. Click **OK**.

   The **Loop Count** parameter is added to the Workflow window.

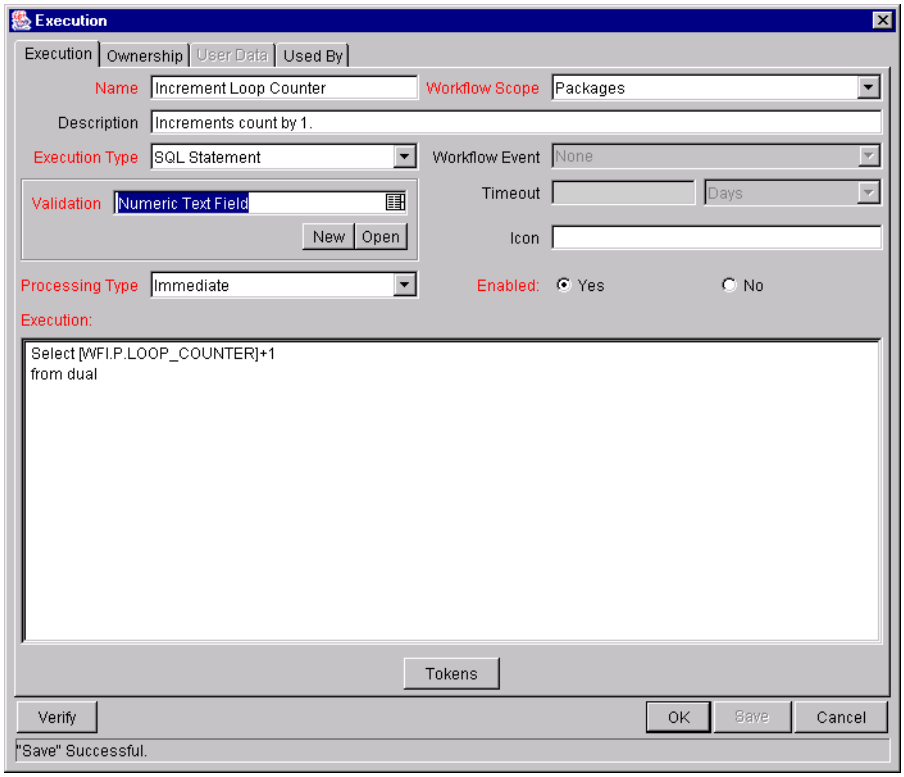


5. Generate a new Immediate SQL Execution Step.

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

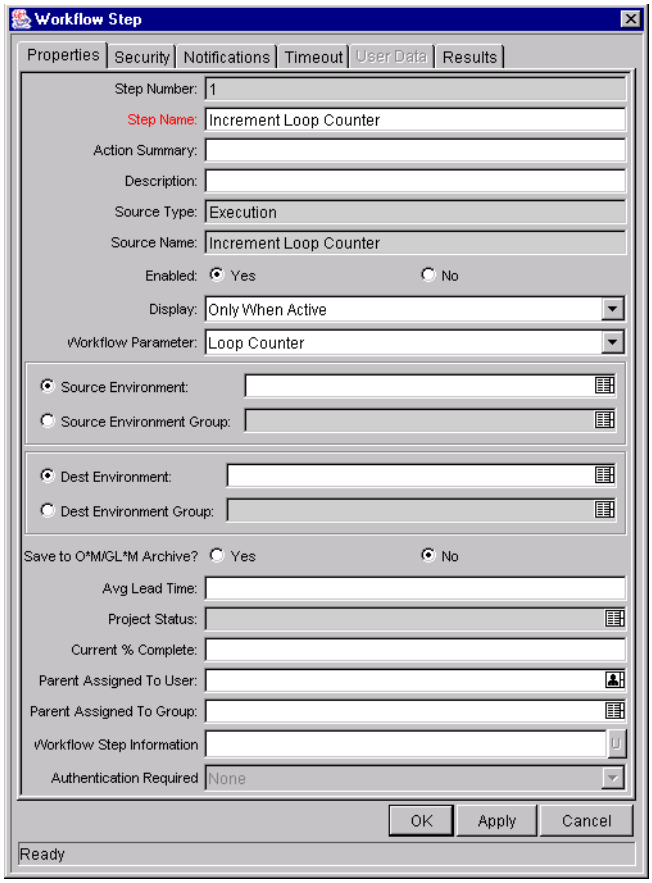
   - The result of the SQL Execution step returns the result **LOOP_COUNTER + 1**. This return value is linked back into the parameter when the Workflow Step is generated on a Workflow.

   - A Validation for a **Numeric** Text Field is used. This allows <=, <, >=, and > comparisons to be used in transitions off this step.



6. 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.

Example

The loop counter can be incremented 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.

Appendix B
Validations

This appendix provides an overview for how to use Validations in your Mercury IT Governance system. Validations determine the acceptable input values for user-defined fields (such as Object Type or Request Type fields). Validations also determine the possible results that a Workflow step can return.

This appendix covers the following topics:

What are Validations

Validations are used for two primary functions:

- Fields:
  Validations determine the field's component type (text field, drop down list, etc.) and the fields possible values. Fields can be created for a number of product entities: Object Types, Request Types, Request Header Types, and User Data.
- Workflow step results:
  Validations determine the possible results exiting a Workflow step. For example, the validation WF - Standard Execution Results contains the possible execution step results of **Succeeded** or **Failed**.

Pre-seeded (system) Validations are included with every product installation or upgrade. When configuring your system, you can select to use these system Validations. If no Validation exists that meets your specific requirements, you can create a new Validation using the Validation Workbench. See *"Creating a Validation"* on page 291 for details.

Validation Component Types - Overview

The following table summarizes the available types of field components. Note that only certain component types can be used in a Workflow step source's Validation.

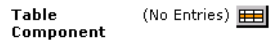
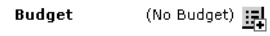
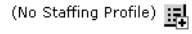
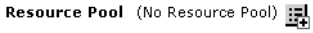
Table B-1. Component Types

Component Type	Use In Workflow?	Example**	Description
Text Field	Yes		Text entry fields displayed on a single line. Text fields can be configured to display the data according to a certain format. For example, you can configure a text field to accept and format a hyphenated nine-digit social security number or a ten digit telephone number.
Drop down list	Yes		Field showing a column of choices.

Table B-1. Component Types

Component Type	Use In Workflow?	Example**	Description
Radio Button	No		Field providing a Yes/No input.
Auto-complete list	Yes		Field showing list of choices with multiple columns.
Text Area	No		Text entry field that can span multiple lines.
Date Field	No		Supports a variety of date and time formats: long, medium, and short.
Web Address (URL)	No		Text entry field for entering a URL. Pressing the U button opens a browser window to the specified web address.
File Chooser	No		Used only in Object Types. Requires that two fields be defined with the following Tokens: P_FILE_LOCATION and P_SUB_PATH. See *"Using Directory and File Choosers"* on page 338 for configuration details.
Directory Chooser	No		Used only in Object Types. Requires that a parameter field be defined with the Token P_FILE_LOCATION.
Attachment	No		Field for indicating file attachments. Comes with buttons for locating files for previewing contents of the selected file.
Password field	No		Field for capturing passwords.

Table B-1. Component Types

Component Type	Use In Workflow?	Example**	Description
Table Component	No		Used to enter multiple records into a single component. The table component can be configured to include multiple columns of varied data types. Additionally, this component supports rules for populating elements within the table and provides functionality for capturing column totals. See *"Configuring the Table Component"* on page 343 for details. Fields of this component can only be added to Request Types, Request Header Types and Request User Data.
Budget	No		Field that can be added to the Request Type to enable access to view, edit or create Budgets associated with a Request or Project. Fields of this component can only be added to a Request Type.
Staffing Profile	No		Field that can be added to the Request Type to enable access to view, edit or create Staffing Profiles associated with a Request or Project. Fields of this component can only be added to a Request Type.
Resource Pool	No		Field that can be added to the Request Type to enable access to view, edit or create Resource Pools associated with a Request or Project. Fields of this component can only be added to a Request Type.

Creating a Validation

Generating certain Workflow steps may require specific validations to ensure that business procedures are being followed. It is necessary to have both the Validation Editor and the Validation Values Editor access grants to add a new validation. See *Security Model Guide and Reference* for a discussion of security groups and access grants.

To define a new Validation:

1. Click **New Validation** on the Validation Workbench or select **File > New > Validation** from the menu.

   The Validation window opens.

2. Enter the name of the new Validation in the Name field.

3. Enter a description of the new Validation in the Description field.

4. Select whether the Validation is enabled or not in the Enabled check box.

5. In the Use in Workflow checkbox, specify whether or not this Validation can be used in a Workflow step source.

   You can only use Text Field, Drop Down List and Auto-Complete component types within Workflow step sources.

6. Select the desired type of Validation from the Component Type drop down list.

   Choices are **Text Field**, **Drop Down List**, **Radio Buttons (Y/N)**, **Auto Complete List**, **Text Area**, **Date Field**, **Web Address (URL)**, **File Chooser**, **Directory Chooser**, **Password Field**, **Attachment**, **Table Component**, Budget, **Staffing Profile**, and **Resource Pool**. Selecting a value from this field will dynamically update the Validation window to display fields used to configure that type of Validation.

7. Enter any additional information required for the component type selected.

8. Click **Ownership** to select which users will be able to edit, copy and delete this validation.

9. To save changes to the Validation without closing the window, click **Save**. To save changes and close the window, click **OK**

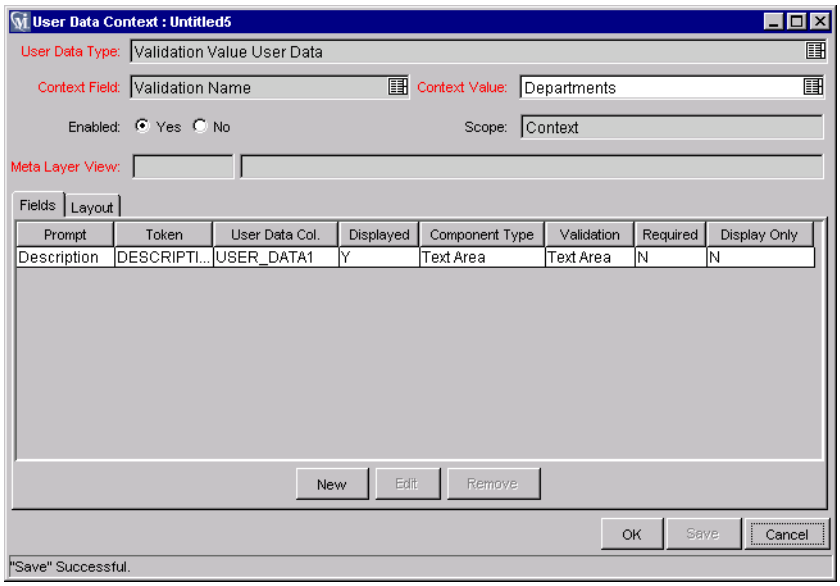
User Data on the Validation Value

You can enable the **User Data** tab to capture more information related to an individual Validation value within a specific Validation. For example, you can create a Description user data field that is associated with the Departments Validation. When you add new values to the validation, you can click on the **User Data** tab and enter a description for that value.

The **User Data** tab can only be used when creating a drop down or an auto-complete validated by a list.

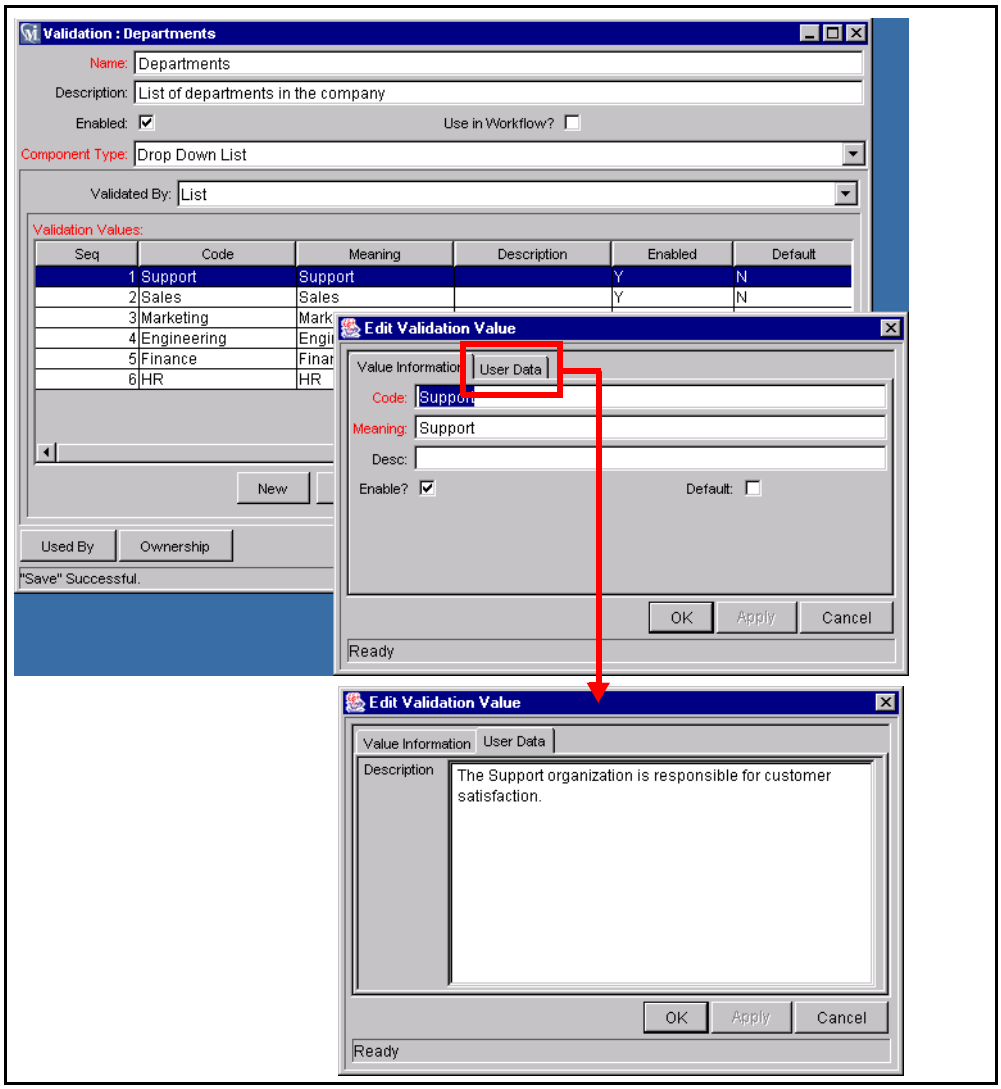
To enable the User Data tab in the Edit Validation Value **window:**

1. Create the Validation and note its name.
2. Open the User Data workbench.
3. Click **New User Data Context**.
4. Select **Validation Value User Data** from the User Data Type field.
5. Click **New** to create a User Data field.



6. Save the settings in the User Data window.
7. On the Validation window, add or edit a Validation value.

The **User Data** tab is now enabled. You can select the tab and enter information in the newly defined user data field.



Editing Validations

You can open and edit Validations using the Workbench. You should exercise caution when editing Validations that are currently used by fields or Workflow

step sources. Both field and Workflow step validations can be tied to Workflow logic. Changing the Validation values can invalidate a process.

For example, ACME changes the Priority field Validation to include a new value **Very Easy**. ACME uses a deployment system Workflow that has an Evaluate Priority step that routes the Package based on the value in the Priority field (using a Token execution type). ACME, however, did not update the Workflow to enable a transition out of the step for the case when Priority = **Very Easy**. When a **Very Easy** Package enters the Evaluate Priority step, it will get stuck.

The following restrictions apply to editing Validations:

- User must have the following Access Grants:
    - Edit Validations
    - Edit Validation Values
- User must be a member of the Ownership Group for the Validation
- You can not change which Validation is associated with a Workflow step source after a Package has traversed that step. You can, however, still edit the values within that Validation.

Creating a URL to Open the Validation Window

You can create a URL that opens a specific Validation in the Workbench. This can provide a quick link to the configuration screen for a Validation that is expected to change frequently. This URL can be included on your internal or external Web pages or a list of browser Favorites to provide convenient access to the Validation's definition.

Use the following URL format to access a specific Validation window:

```
http://host:port/kintana/servlet/SmartURL?screen=VAL&pkname=<Va
lidationName>
```

The following URL opens the Validation window for the Validation named "Development Priorities."

```
http://host:port/kintana/servlet/SmartURL?screen=VAL&pkname=
   Development+Priorities
```

Deleting Validations

Validations can be deleted from the Workbench. To delete a Validation, you must be a member of the Validation's Ownership Group and have the Edit Validations access grant.

A Validation can not be deleted when:

- It is a system Validation (a Validation that is delivered with the product as seed-data)
- It is being used by a Workflow step source. Validations referenced by Workflow step sources can only be disabled. A disabled Validation continues to function in existing Workflow steps, but can not be used when defining a new step source.
- It is being used by a field in a product entity (Object Type, Request Type, User Data, Report Type, or Project Template field). Validations referenced by entity fields can only be disabled. A disabled Validation continues to function in existing fields, but can not be used when defining a new field.

Although you may not be able to delete a custom Validation in all cases, you can disable it. This will allow the Validation to be used in any active Workflows or product entities, but will keep it from being used in any new Workflow or entity definitions.

Static List Validations

You can create Validations that provide a static list of options to the user. For example, ACME, Inc. can create a Validation for their engineering teams. They create a Validation called Engineering Teams, consisting of the following values: **New Product Introduction**, **Product One**, and **Product Two**.

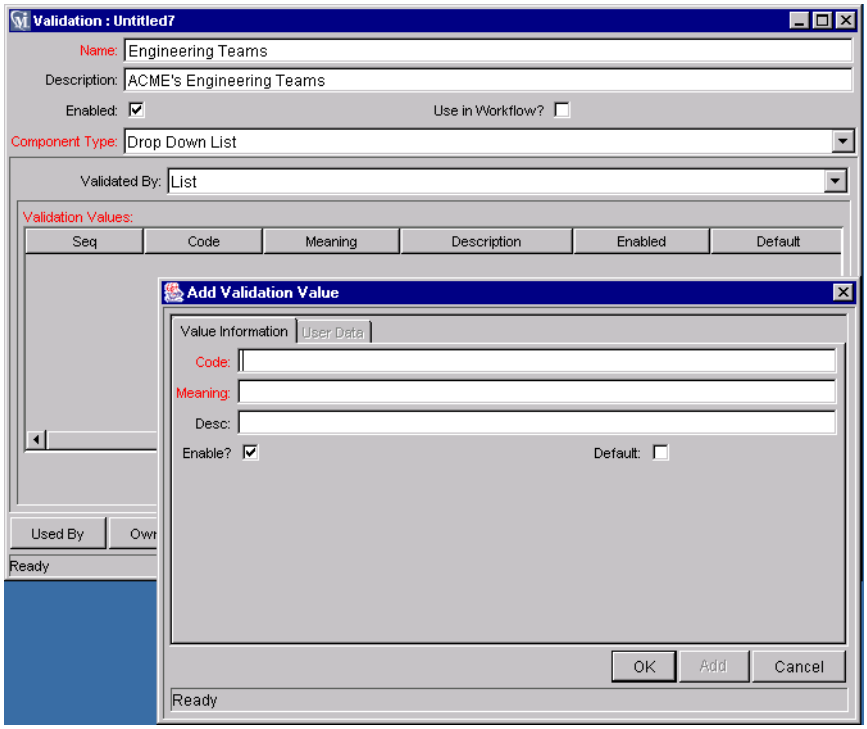
A static list validation can be a drop down or an auto-complete list component.

To add values to the Validation list:

1. In the Validation window, select **Drop Down List** or **Auto Complete List** from the Component Type field.
2. Select **List** from the Validated By field.

3. Click **New** and add a value.

   The Add Validation Window opens.



4. Enter information for the Validation value as described in the following table.

Field	Definition
Code	The underlying code for the Validation value. The code is the value stored in the database or passed to any internal functions, and is rarely displayed.
Meaning	The displayed meaning for the Validation value in the drop down list or auto-complete.
Default	The default value for the list. This value is initially displayed in drop down lists (this is not used for auto-complete lists). There can be only one default value per list.

5. Optionally set the Validation value as the default by checking the Default field.

The default option is only available for drop down lists.

6. Click **OK** to close the window and add the value to the Validation. Click **Add** to add the value and keep the Add Validation Value open.

Validation values can be re-ordered using the up and down arrow buttons. The sequence of the Validation values determines the order that the values are displayed in the list.

You can copy existing values defined in other Validations using the **Copy From** button. Click **Copy From** and query an existing list-validated Validation and choose any of the Validation values. Click **Add** or **OK** in the Copy From window and the selected value or values are added to the list.

Be careful when creating Validations (drop down lists and auto-complete lists) that are validated by lists. Each time the set of values changes, you will be forced to update the Validation. Consider, instead, validating using a SQL query or PL/SQL function to obtain the values from a database table.

Dynamic List Validations

You can create Validations that provide a dynamic list to the user. This is often a better approach than defining static list validations. Each time a static list Validation needs to be updated, a manual update has to occur. Dynamic list Validations can often be constructed in such a way as to automatically pick up and display the altered values.

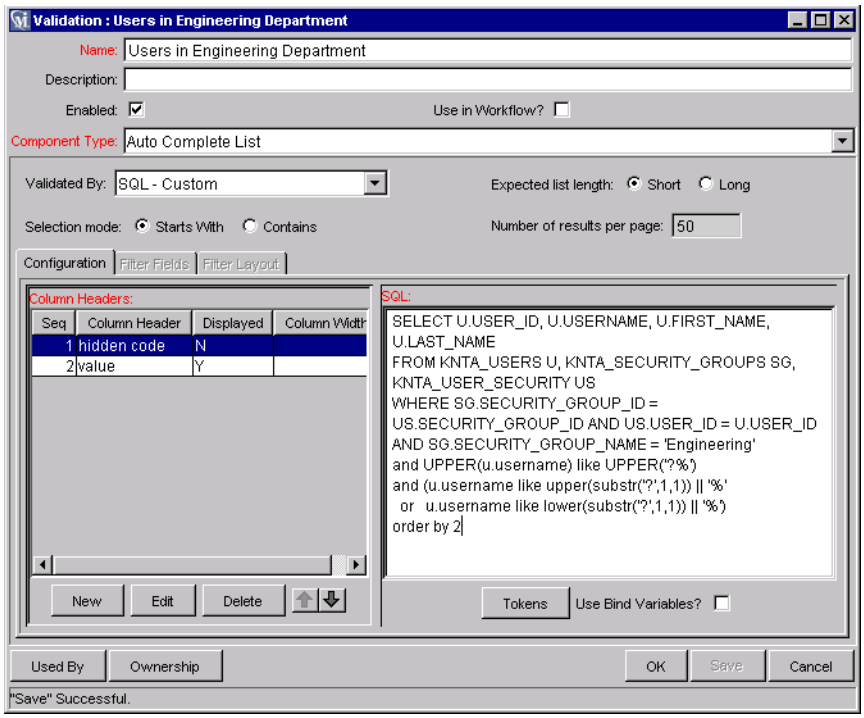
For example, ACME needs a field validation that will list all users who are on their Support Team. They could construct a Validation that is validated by a list of users, but any time the Support Team changed (members join or leave the department) the list would have to be manually updated. ACME decides instead to create a dynamic list validation. They create an auto-complete list validation that is validated by a SQL statement. The SQL statement returns all users who are a member of the **Support Team** Security Group. When the Security Group membership is altered, the Validation is automatically updated with the correct values.

A dynamic list validation can be created using a drop down or an auto-complete list component. The lists are dynamically generated using either:

- *SQL Validation*
- *Command Validation*

SQL Validation

You can use a SQL statement to generate the values in a Validation. SQL can be used as a validation method for drop down lists and auto-complete lists. To define a dynamic list of choices, set a drop down list or auto-complete list to Validated By - **SQL**. Then in the SQL area, enter the Select statement that queries the necessary database.



If an auto-complete list is being used, you can define headers for the selected columns. These column headers are used in the window that opens when a value from an auto-complete list is selected. Click **New** under Column Headers. *Table 0-2* shows the fields that can be entered for a column header. If a column header is not defined for each column in a SQL query, a default name is used.

Table 0-2. Column Headers

Field	Definition
Column Header	The name of the column that is displayed in the auto-complete window.
Display	Determines whether or not the column is displayed. The first column is never displayed and the second column is always displayed.

ACME, Inc. creates an auto-complete field that lists all users in the "Engineering" department. They choose to validate the list by SQL.

```
SELECT U.USER_ID, U.USERNAME, U.FIRST_NAME, U.LAST_NAME
FROM KNTA_USERS U, KNTA_SECURITY_GROUPS SG,
   KNTA_USER_SECURITY US
WHERE SG.SECURITY_GROUP_ID = US.SECURITY_GROUP_ID AND
   US.USER_ID = U.USER_ID
AND SG.SECURITY_GROUP_NAME = 'Engineering'
and UPPER(u.username) like UPPER('?%')
and (u.username like upper(substr('?',1,1)) || '%'
  or   u.username like lower(substr('?',1,1)) || '%')
order by 2
```

When a new user account is created and is added to the "Engineering" Security Group, that user will automatically be included in the auto-complete list.

A Validation may already exist that meets your process requirements. If it does, consider using that Validation in your process. Also consider copying and modifying Validations that are similar to the desired Validation. See *"System Validations"* on page 359 for a complete list of Validations that are delivered with the product.

SQL Validation Tips

The following guidelines are helpful when writing a SQL statement for a SQL-validated Validation:

- The SQL statement must query at least two columns. The first column is a hidden value which is never displayed, and is often stored in the database or passed to internal functions. The second column is the value that is displayed in the field. All other columns are for information purposes and

are only displayed in the auto-complete window. Extra columns are not displayed for drop down lists.

- When something is typed into an auto-complete list field, the values in the auto-complete window that appear are constrained by what was first typed in the field. Generally, the constraint is case insensitive. This is accomplished by writing the SQL statement to query only values that match what was typed.

  Before the auto-complete window is displayed, all question marks in the SQL statement are replaced by the text that the user typed. In general, if the following conditions are added to the WHERE clause in a SQL statement, the values in the auto-complete window are constrained by what the user typed.

```
where UPPER(<displayed_column>) like UPPER('?%')
and (<displayed_column> like upper(substr('?',1,1)) || '%'
or   <displayed_column> like lower(substr('?',1,1)) || '%')
```

  Any column aliases included directly in the SQL statement are not used. The names of the columns, as displayed in auto-complete lists, are determined from the Column Headers. Drop down lists do not have column headers

Command Validation

An auto-complete list can contain command line executions that return and display a list of values. To define a dynamic list of choices, set an auto-complete list to Validated By - **Command with Delimited Output** or **Command with Fixed Width Output**. Then enter commands the Commands area. See *"Configuring the Auto-Complete Values"* on page 315 for detailed instructions.

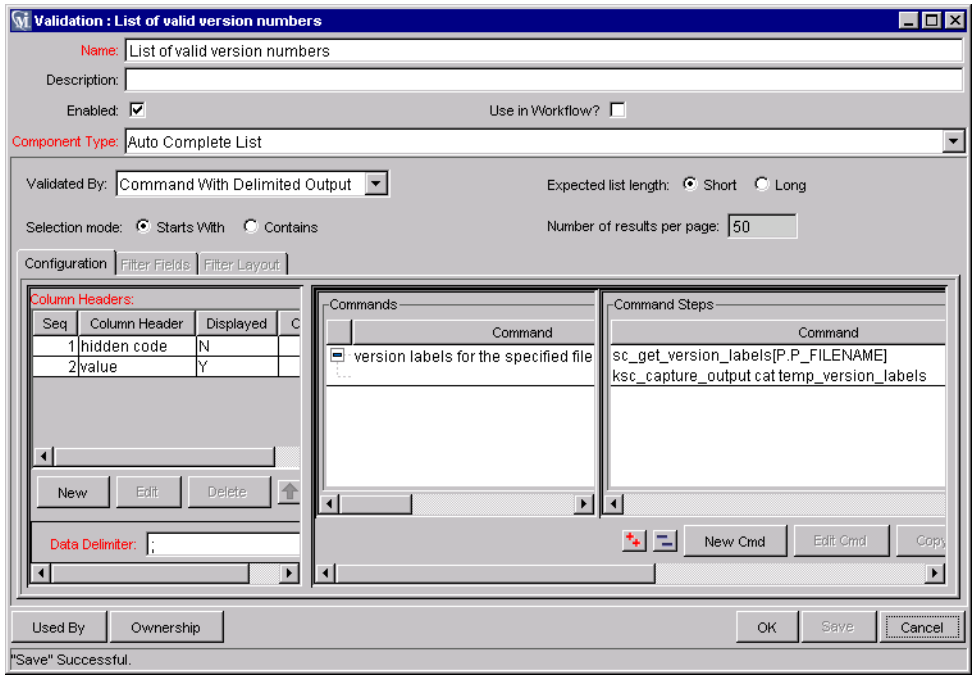


Figure B-1 Auto Complete Using Command Validation

Configuring Auto-Complete Validations

Auto-complete fields are used throughout the Mercury IT Governance Center to provide users with an efficient way to select field values from a set of valid choices. Configuring auto-complete fields consists of two activities:

- Specifying general auto-complete behavior
- Configuring the validation values

Configuring General Auto-complete Behavior

Auto-complete fields can be used for validations with a small or large number of choices. The auto-complete can be configured to behave differently depending on the expected number of values. For example, if you expect a large number of entries, the auto-complete window will include an interface that allows you to page through your results. Additionally, you can configure how the “auto-complete” feature of the field behaves. For example, you can

configure the auto-complete field to automatically complete entries that either start with or contain a text string.

This section covers the following topics related to configuring auto-complete field behavior:

- *Configuring Short List Auto-Complete Fields*
- *Configuring Long List Auto-Complete Fields*
- *Configuring the Automatic Value Matching and the Interactive Select Page*
- *Adding Search Fields to the Auto-Complete Window*

Configuring Short List Auto-Complete Fields

Auto-complete fields configured to display a short list of entries, displaying all of the values on a single page. *Figure B-2* shows the Select window for a short list auto-complete field.

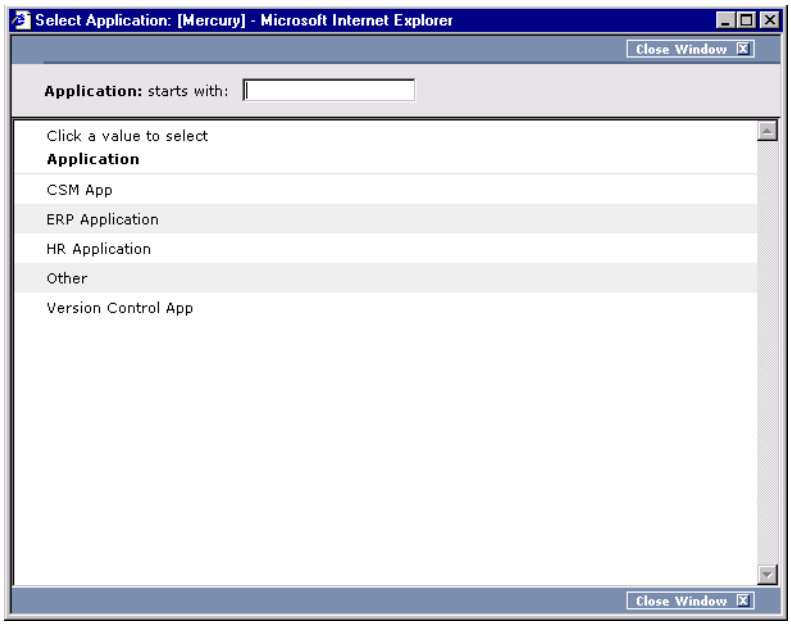


Figure B-2 Short List Auto-Complete

Auto-completes configured as short lists will load all values when the window is opened. This can lead to a slower load time and an unfavorable user experience. For fields with many possible values, consider formatting the auto-complete using the long list format.

To configure a short list auto-complete field:

1. Create a new Validation or open an existing Validation.

   The Validation window opens.

2. From the Component Type field, select **Auto Complete List**.

3. In the Expected list length field, select **Short**.

4. Click **Save**.

Configuring Long List Auto-Complete Fields

Auto-complete fields configured to display a long list of entries, dividing the results between multiple pages. By default, 50 results are shown per page. End users can page through the results or further limit the results by specifying text in one of the available filter fields at the top of the page. *Figure B-3* shows the Select window for a long list auto-complete field.

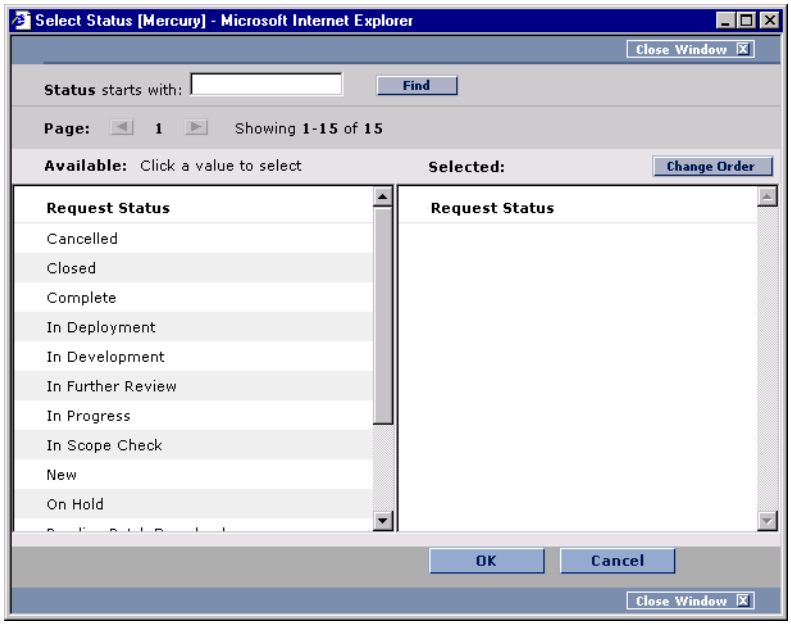


Figure B-3 Long List Auto-Complete

Tip

Auto-completes configured as long lists only load a limited set of values when the window is opened. For extremely long lists or lists that are at risk of loading slowly (for example the values are obtained from an alternate database), consider using the long list format.

To configure a long list auto-complete field:

1. Create a new Validation or open an existing Validation.

   The Validation window opens.

2. In the Expected list length field, select **Long**.

3. Click **Save**.

Note

All auto-completes that are validated by **SQL - User** are required to use the long list auto-complete format. This selection is automatically defaulted when the user selects **SQL - User** from the Validated By field on the Validation window.

Configuring the Automatic Value Matching and the Interactive Select Page

This section provides instructions for configuring auto-complete fields to filter a list of possible values based on a matching character string. It also provides instructions for configuring the automatic value-limiting that occurs on the auto-complete's Select page. *Figure B-4* shows an auto-complete field that has opened to display matching values.

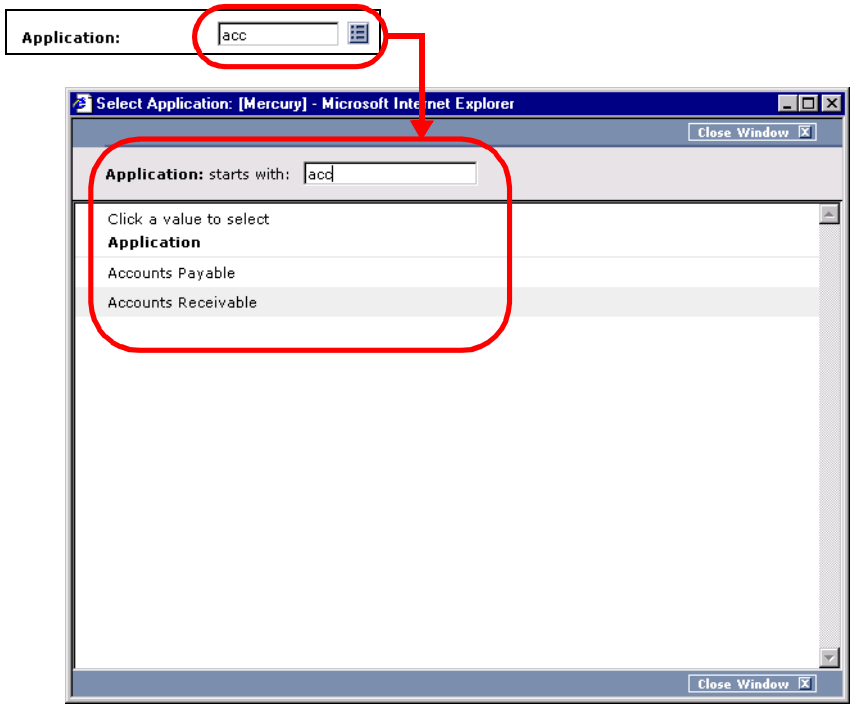


Figure B-4 Auto-complete field and matching values in the Select page

This section discusses the following topics:

- *Functional Overview: Matching for "Starts with" or "Contains"*
- *Configuration Instructions*
- *Configuration Tips*

Functional Overview: Matching for "Starts with" or "Contains"

Auto-complete field behavior can be divided into the following areas:

- Field behavior—A user types a character in the field and presses the Tab key. If an exact match is not available, the Select page opens.

- The Select page behavior—For lists that are configured appropriately, when a user types a character or characters into the field at the top of the page, the results are automatically limited to display only matching entries.

For both the field and Select page behaviors, automatic value matching can be based on either "starts with" character matching or "contains" character matching. The following table summarizes this behavior:

Table 0-3. Automatic character matching field behavior

Character matching mode	Description of Behavior
Starts with	Type characters and press tab. The selection window will open and list entries that begin with the specified characters.
Contains	Type characters and press tab. The selection window will open and list entries that contain the specified character string. This is the same behavior as a wild card search, which uses the % character at the beginning of the search text.

Table 0-4. Automatic character matching Select page behavior

Character matching selection mode	Description of Behavior
Starts with	Type characters and the list will automatically be filtered for entries that begin with the specified characters.
Contains	Type characters and the list will automatically be filtered for entries that contain the character string.

Configuration Instructions

The field and the Select page behavior are configured distinctly in the Validation window for the specific auto-complete list. This section provides instructions for configuring the "starts with" and "contains" functionality in the field and Select page, as described in *"Functional Overview: Matching for "Starts with" or "Contains""* on page 305.

To configure "starts with" matching from the auto-complete window to the selection window, add the following to the SQL WHERE clause:

```
UPPER(value) like UPPER('?%') and (value like
upper(substr('?',1,1)) || '%' or value like
lower(substr('?',1,1)) || '%')
```

To configure "contains" matching from the auto-complete window to the selection window, add the following to the SQL WHERE clause:

```
UPPER(value) like UPPER('%?%') and (value like '%' ||
upper(substr('?',1,1)) || '%' or value like '%' ||
lower(substr('?',1,1)) || '%')
-
```

To configure "starts with" matching within the interactive selection window:

1. Open the auto-complete's Validation window.
2. From the Expected list length field, select **Short**.

   This feature is only available for short lists.
3. From the Selection mode radio button, select **Starts With**.
4. Save the Validation.

This setting only controls the matching in the Select page. Matching in the auto-complete field is controlled by including specific clauses in the auto-complete's SQL. See above for details.

To configure "contains" matching within the interactive selection window:

1. Open the auto-complete's Validation window.
2. From the Expected list length field, select **Short**.

   This feature is only available for short lists.
3. From the Selection mode radio button, select **Contains**.
4. Save the Validation.

This setting only controls the matching in the Select page. Matching in the auto-complete field is controlled by including specific clauses in the auto-complete's SQL. See above for details.

Configuration Tips

Consider the following tips when configuring the “starts with” versus “contains” functionality for auto-complete fields and the Select page.

Auto-completes should be configured such that the field matching behavior works the same way as the Select page matching behavior. Specifically, if the auto-complete field uses the "starts with" clauses in the SQL, then the selection window should use the "Starts With" Selection Mode. See *“Configuration Instructions”* on page 306 for details.

Consider using the "Contains" Selection Mode for fields with multi-word values. For example, consider the possible values for the Request Type auto-complete field:

```
Development Bug
Development Enhancement
Development Issue
Development Change Request
IS Bug
IS Enhancement
IS Issue
IS Change Request
Support Issue
Support Change Request
```

Using "contains" can be useful here. The user knows that he needs to log a bug against one of the IS-supported Financial applications. The user types "bug" into the auto-complete field and presses the Tab key. The following items are returned:

```
Development Bug
IS Bug
```

The user selects "IS Bug." Without the "contains" feature enabled, typing "bug" would have returned the entire list. He might have also typed "Financial," thinking that there might be a separate Request Type used for each type of supported application. This, too, would have returned the entire list. At that point, the user would be forced to try another "starts with" phrase or simply read the entire (potentially long) list.

Adding Search Fields to the Auto-Complete Window

Auto-completes with a long list of values can be configured to display additional filter fields in the Select window. These fields can be used to search other properties than the primary values in the list. Users can enter values in

the filter fields and click **Find** to display only the values that match the search criteria. *Figure B-3* shows the Select window with additional filter fields.

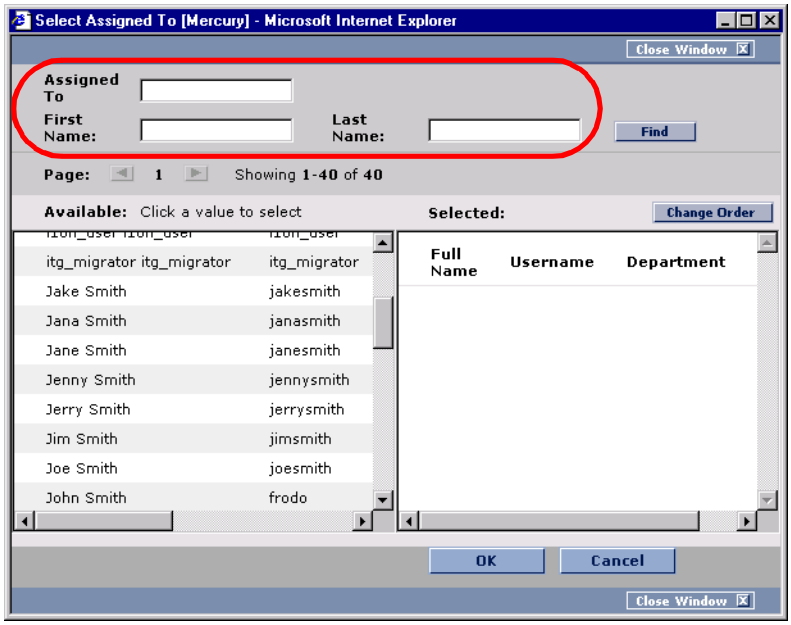


Figure B-5 Filter Fields in the Auto-Complete Select Window

Note

Filter fields can not be configured when validating your list by **List**, **Command With Delimited Output** or **Command With Fixed Width Output**.

To add a filter field to the auto-complete validation:

1. Open the Validation for the auto-complete.

   Auto-complete validations must display **Auto Complete List** in the Component Type field.

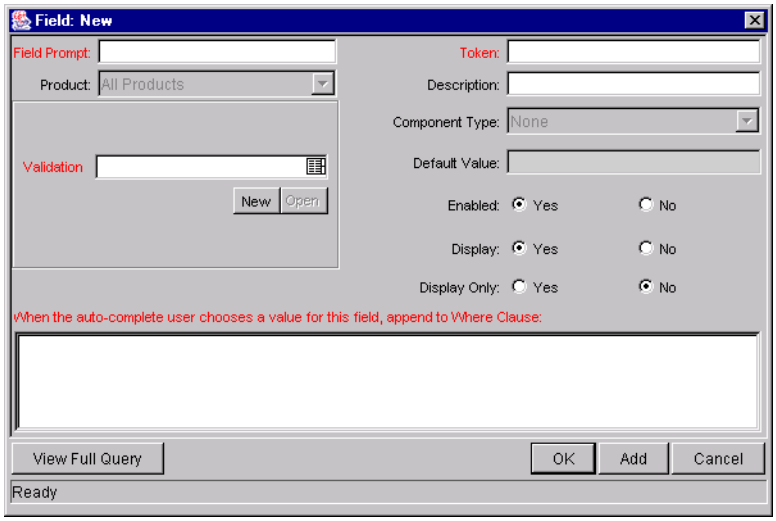
2. In the Expected list length field, select **Long**.

   Only long formatted auto-complete lists can include filter fields.

3. Click the **Filter Fields** tab.

4. Click **New**.

The Field: New window opens.



5. Enter the required information.

   Table B-2 defines all of the fields on this window.

Table B-2. Fields in the Fields:New Window

Field	Description
Field Prompt	The name that is displayed for the field in the auto-complete Select window.
Product	
Validation	The Validation for the filter field. You can select any type of Validation, except for auto-complete type Validations. The values accepted by this Validation will be appended to the WHERE clause in the SQL query that determines the ultimate auto-complete list display.
New	Opens the Validation window where you can construct a new Validation for the filter field. Note that you can not use an auto-complete type Validation for the filter field.
Open	Opens the Validation window and displays the definition of the Validation specified in the Validation field.
Token	The Token for the field value. The Token value will be appended to the WHERE clause in the SQL query that determines the ultimate auto-complete list display.

Table B-2. Fields in the Fields:New Window [continued]

Field	Description
Description	The description of the filter field.
Component Type	The component type for the filter field, determined by its Validation.
Default Value	The default value for the filter field, determined by its Validation.
Enabled	Determines whether the filter field is enabled.
Display	Determines whether the filter field is visible to the user in the auto-complete's Select window.
Display Only	Determines whether the filter field is updatable. When Display Only is set to **Yes**, the field can not be updated.
When the auto-complete user chooses a value for this field, append to WHERE clause:	The AND clause that is appended to the portlet's WHERE clause if the user enters a value in this filter field. Each filter field will append its term to the portlet query when a value is entered by the end user in the Select window. For example, if the filter field uses the CRT-Priority-Enabled Validation and a filter field Token of P_PRIORITY, enter the following into this field: `AND R.PRIORITY_CODE = '[P.P_PRIORITY]'` Note: The value in this field must start with "AND".
View Full Query	Opens a window showing the full query.

6. Click **OK**.

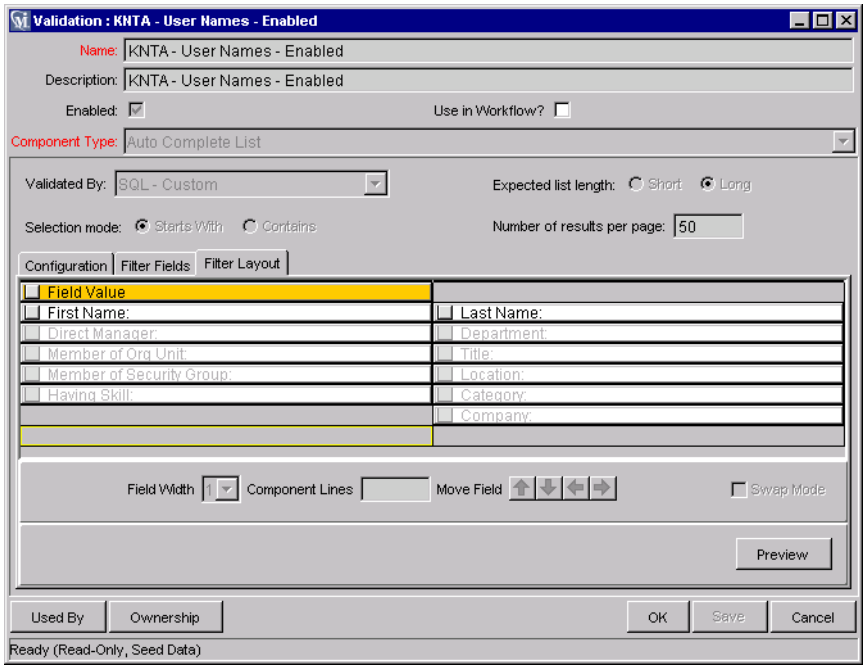
Filter fields can offer a powerful method for enabling users to efficiently locate specific values in large lists. When adding filter fields to an auto-complete Validation, consider the following tips:

- Ensure that the filter fields are functionally related to the list of values. For example, a validation that provides a list of Request Types can include a filter field for a specific Department associated with the Request Types.
- Consider reusing (copying) an auto-complete Validation and modifying the filter fields to display a subset of the entire list. Using the Displayed, Display Only, and Default fields in the Filter Field window, you can configure the auto-complete values to automatically limit the results.
- Performance can degrade if joining tables over database links.
- Only use this functionality for complex fields.

To modify the filter field layout:

1. Open the auto-complete Validation that includes filter fields on the **Filter Fields** tab.
2. Click the **Filter Layout** tab.

   The tab lists the primary field and all of the filter fields that have been defined for the auto-complete. The primary field is named Field Value. This is the field that holds the eventual selected value.



3. Select the field that you would like to move.

   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 (i.e. the Ctrl-Select functionality is not supported).

4. Use the arrow buttons to move the fields to the desired location in the layout builder.

Note

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.

5. 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 you want to 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.

6. 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:

   - 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.

Special Case: Configuring an Auto-Complete List of Users

User auto-completes or Validations (Validated by: SQL-User) have three filter fields by default:

- Primary field—this field takes the name of the auto-complete field
- First Name
- Last Name

The user auto-complete always appears in the long list format, which uses the paging interface to display the items. Additionally, user auto-completes display a different icon ( ) in the auto-complete field.

To configure a user auto-complete Validation:

1. Create a new Validation.

   The Validation window opens.

2. From the Component Type field, select **Auto Complete List**.

3. From the Validated By field, select **SQL - User**.

4. Configure the SQL query that will determine the users listed in the Validation.

   See *"Configuring the Auto-Complete Values"* on page 315 for details.

5. Click **Save**.

Configuring the Auto-Complete Values

The values in an auto-complete list can be specified in the following ways. In the Validate By field, select one of the following:

- **List**: Used to enter specific values.
- **SQL**: Uses a SQL statement to build the contents of the list.
- **SQL - User**: Identical to SQL configuration, but includes a few additional preconfigured filter fields.
- **Command With Delimited Output**: Uses a system command to produce a character-delimited text string and uses the results to define the list.
- **Command With Fixed Width Output**: uses a system command to produce a text file and parses the result on the basis of the width of columns, as well as the headers.

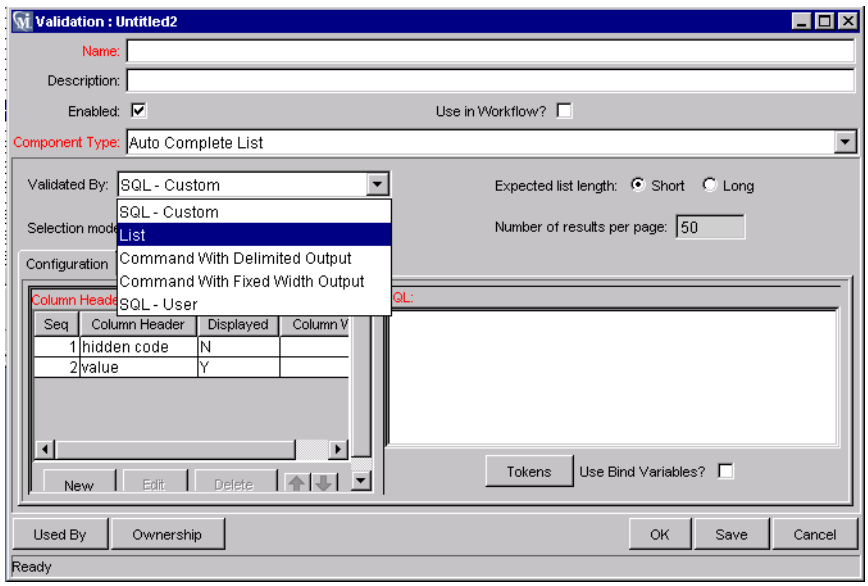


Figure B-6 Auto-Complete List

The following sections discuss the following topics:

- *Validation by Command With Delimited Output*
- *Validation by Command With Fixed Width Output*
- *User-Defined Multi-Select Auto-Complete Fields*
- *Example: Token Evaluation and Validation by Command with Delimited Output*

For more information on creating auto-completes validated by List or SQL, refer to the following sections:

- *"Static List Validations"* on page 295
- *"Dynamic List Validations"* on page 297

Validation by Command With Delimited Output

Validations that are validated by commands with delimited output can be used to get data from an alternate source, and use that data to populate an auto-complete field. This functionality provides additional flexibility when designing auto-complete lists.

Many enterprises need to use alternate sources of data within their applications. Examples of these sources are a flat file, an alternate database source, or output from a command line execution. Special commands may be used in conjunction with these alternate data sources, in the context of a Validation, to provide a list of values.

To configure a validation by command with delimited output:

1. In the Validation Workbench, under Validated By, choose **Command With Delimited Output** and input the delimiting character.

2. Under New Command, enter in the command steps to be executed.

   These can include Mercury ITG Special Commands. Your commands should include the Special Command `ksc_capture_output`, which captures and parses the delimited command output. If the `ksc_capture_output` Special Command is surrounded by the `ksc_connect` and `ksc_disconnect` commands, the command will be run on the remote system. Otherwise, the command will be run locally on the Mercury ITG server (similar to `ksc_local_exec`).

The simple example below uses a comma for a delimiter and has the validation values red, blue and green. The script places the validations into the newfile.txt file, and then uses the Special Command ksc_capture_output to process the text of the file.

```
ksc_begin_script[AS.PKG_TRANSFER_PATH]newfile.txt
red,red
blue,blue
green,green
ksc_end_script
ksc_capture_output cat[AS.PKG_TRANSFER_PATH]newfile.txt
```

Table B-3 shows the Validation window for Command with Delimited Output.

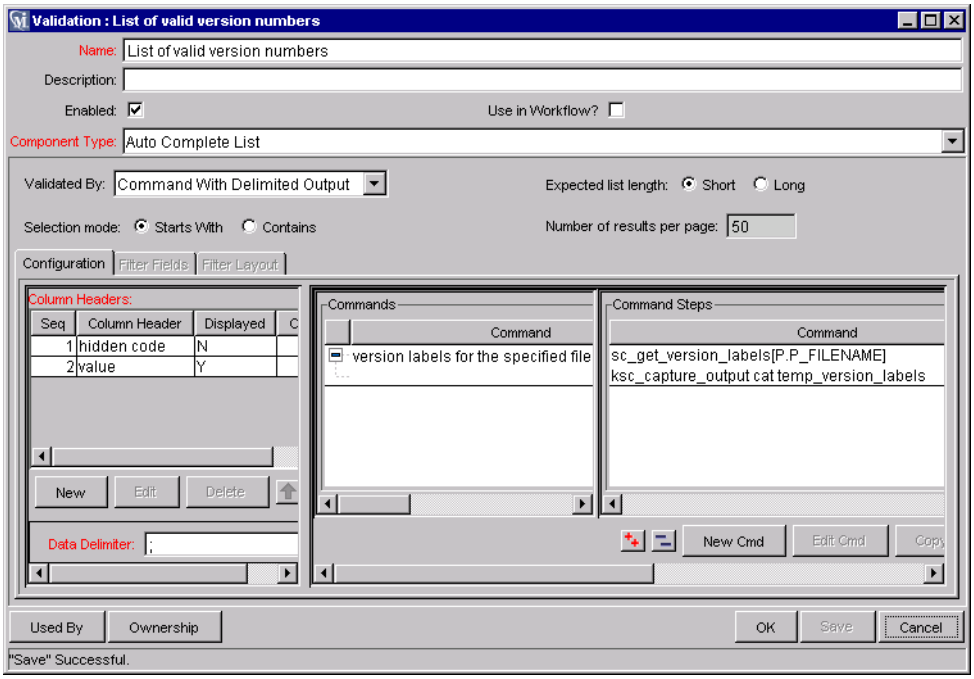


Figure B-7 Validation by Command with Delimited Output

Table B-3. Validation by Command With Delimited Output

Field	Definition
Command Panel	Panel where new commands can be added to capture Validation values.
Data Delimiter	Indicates the character or key by which the file will be separated into the Validation columns.

Headers can also be defined for the columns selected. These column headers are used in the window that opens when a value is selected from an auto-complete list. To define a new header, click **New** under Column Header. *Table B-4* shows the fields that can be entered for a column header. If a column header is not defined for each column in a Command, a default name is used.

Table B-4. Column Headers

Field	Definition
Column Header	The name of the column that is displayed in the auto-complete window.
Display	Determines whether or not the header is displayed in the Validation.

Validation by Command With Fixed Width Output

Validations by Command with Fixed Width Output can be used to obtain data from an alternate source, and use that data to populate an auto-complete field. This functionality provides additional flexibility when designing auto-complete lists.

Many enterprises need to use alternate sources of data within their applications. Examples of these sources are a flat file, an alternate database source, or output from a command line execution. Special commands may be used in conjunction with these alternate data sources, in the context of a Validation, to provide a list of values on the fly.

In the Validation Workbench, under Validated By, choose **Command With Fixed Width Output** and input the appropriate width information.

Then, under New Command, enter in the command steps to be executed. These can include Special Commands. Your commands should include the Special Command `ksc_capture_output`, which captures and parses the delimited command output. If the `ksc_capture_output` Special Command is surrounded by the `ksc_connect` and `ksc_disconnect` commands, the command will be run on the remote system. Otherwise, the command will be run locally on the Mercury ITG server (similar to `ksc_local_exec`).

The example below has the validations red, blue and green. The column width is set to a value of 6. The script places the validations into the newfile.txt file.

```
ksc_begin_script[AS.PKG_TRANSFER_PATH]newfile.txt
red    red
blue   blue
green  green
ksc_end_script
ksc_capture_output cat[AS.PKG_TRANSFER_PATH]newfile.txt
```

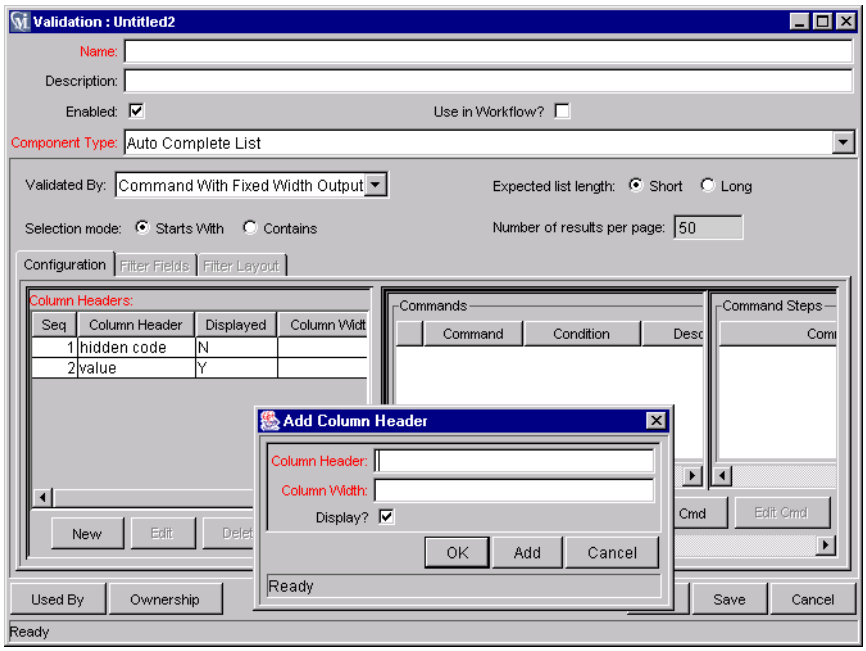


Figure 0-2 Validation by Command with Fixed Width Output

Table B-5. Validation by Command With Fixed Width Output

Field	Definition
Command Panel	The panel where new commands can be added to capture Validation values.

Headers can also be defined for the columns selected. These column headers are used in the window that opens when a value is selected from an auto-complete list. To define a new column header, click **New** under Column Header. *Table B-6* shows the fields can be entered for a column header. If a column header is not defined for each column in a Command, a default name is used.

Table B-6. Column Headers

Field	Definition
Column Header	The name of the column that is displayed in the Auto Complete dialog.

Table B-6. Column Headers

Field	Definition
Display	Whether or not the column is displayed. The first column is never displayed and the second column is always displayed.
Column Width	The number of characters in each column of the output generated as a result of the command.

User-Defined Multi-Select Auto-Complete Fields

A number of auto-complete fields in the Workbench have been pre-configured to allow users to open a separate window for selecting multiple values from a list. Users can also define custom auto-complete fields to have multi-select capability when creating various product entities.

The user-defined multi-select capability is supported for:

- User Data fields
- Report Type fields
- Request Type fields
- Project Template fields

The user-defined Multi-Select capability is **not** supported for:

- Request Header Types
- Object Types

In order to use this feature when creating a new entity, users must:

- Select a Validation for the new entity that has **Auto-Complete List** as the Component Type. This enables the Multi-Select Enabled field in the Field: New window.
- In the Field: New window, users must click **Yes** for the Multi-Select Enabled radio button.

The step-by-step procedure for defining multi-select capability in User Data, Report Type, Request Type Project Template fields is very similar. The procedure for enabling this capability for Request Type field is shown below as an example.

To define a multi-select auto-complete field for a Request Type:

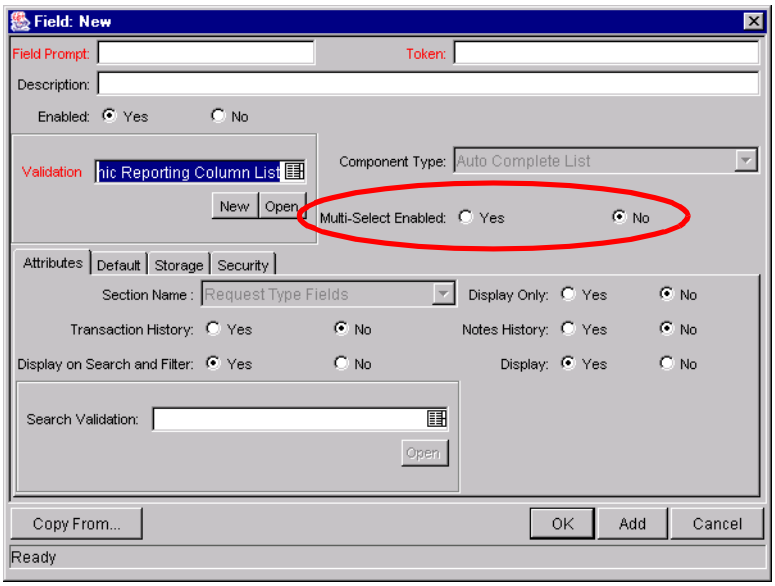
1. Click the **Create** screen group and click the **Request Types** screen.

   The Request Type Workbench opens.

2. Click **New Request Type**.

   The Request Type window opens.

3. Click **New**. The Field: New window opens.



4. Click the auto-complete icon for the **Validation** field.

   The Validate window opens.

5. In the Validate window, select a Validation that has **Auto-Complete List** as the Component Type.

6. Click **OK** in the Validate window.

   The Validate window closes. The **Validation** field is populated with the selection from the Validate window. The Multi-Select Enabled option is now enabled.

7. Click the **Yes** radio button for the Multi-Select Enabled option.

   The Possible Conflicts window opens. It warns you not to use a multi-select auto-complete for Advanced Queries, Workflow Transitions and Reports.

If this field is not going to be used in Advanced Queries, Workflow Transitions or Reports, click **Yes** to continue.

8. Configure the other options in this window for the new Request Type.
9. Click **OK**.

The field is now enabled for multi-select auto-complete.

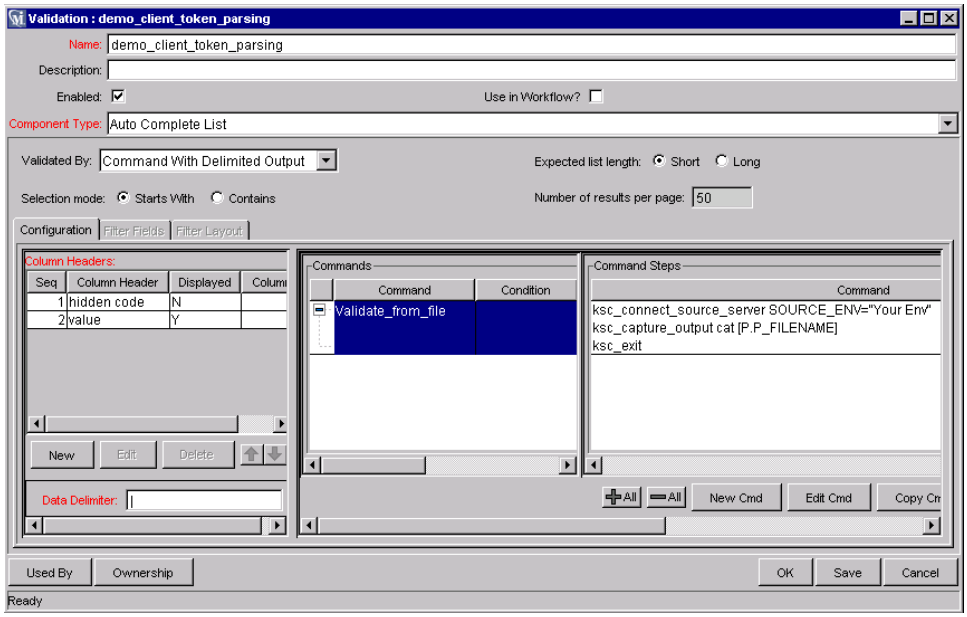
Example: Token Evaluation and Validation by Command with Delimited Output

The Validation functionality can be extended to include field dependent token evaluation. Validations can be configured to dynamically change, depending on the client-side value entered in another field.

To use field dependent token evaluation, it is necessary to configure a Validation in conjunction with an Object Type, Request Type, Report Type, Project Template, or User Data definition. Consider the following example for setting up an Object Type using field dependent tokens.

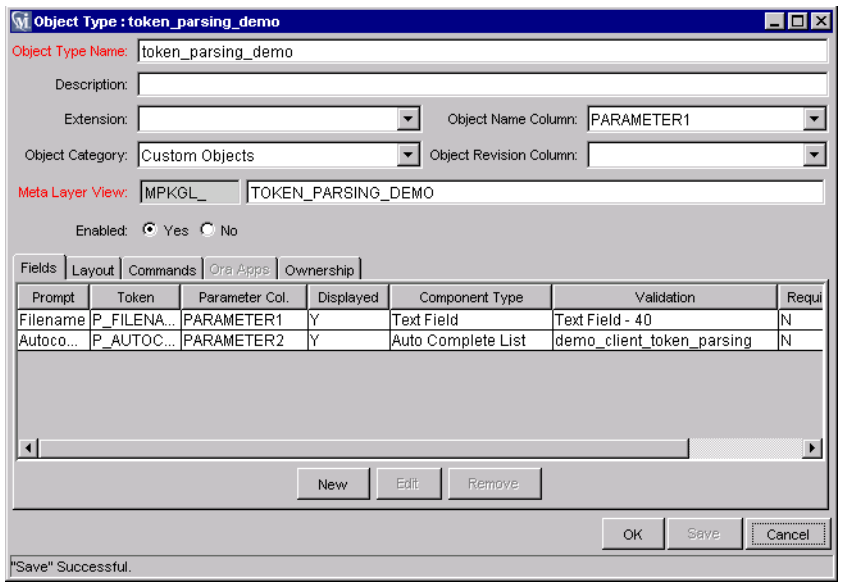
1. Generate a Validation and set the following parameters as shown:
   a. Name: **demo_client_token_parsing**
   b. Component Type: **Auto Complete List**
   c. Validated By: **Command With Delimited Output**
   d. Data Delimiter: | (bar)
   e. Command
      - Command: **Validate_from_file**
      - Steps

```
ksc_connect_source_server SOURCE_ENV="Your Env"
ksc_capture_output cat [P.P_FILENAME]
ksc_exit
```



When called, this Validation will connect to an Environment called 'Your Env' and retrieve data from a file specified by the token P_FILENAME. The file should be located in the directory specified in the Base Path in the Environment window.

2. Generate an Object Type named **token_parsing_demo**.



a. Generate a new field with the following parameters:

- Name: **Filename**
- Token: **P_FILENAME**
- Validation: **Text Field - 40**

b. Generate a new field with the following parameters:

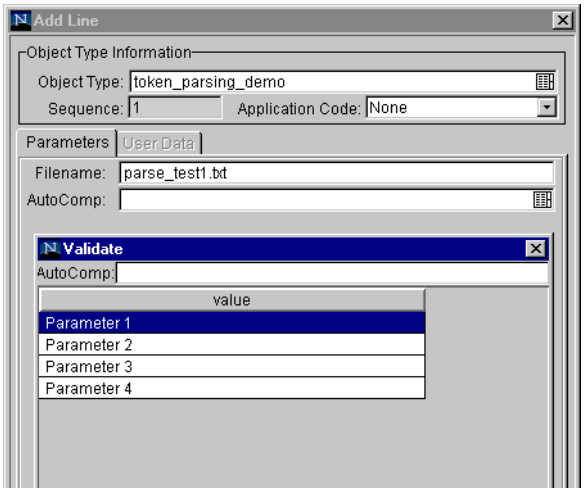
- Name: **AutoComp**
- Token: **P_AUTOCOMP**
- Validation: **demo_client_token_parsing** (this is the Validation that was defined above)

3. For this example to return any values in the auto-complete, a file must be generated in the directory specified in the Base Path in the Environment Detail of 'Your Env' Environment. Generate a file named 'parse_test1.txt' with the following delimited data:

```
DELIMITED_TEXT1|Parameter 1
DELIMITED_TEXT2|Parameter 2
DELIMITED_TEXT3|Parameter 3
DELIMITED_TEXT4|Parameter 4
```

The Object Type 'token_parsing_demo' is now enabled to use this token evaluation. To test the above configuration sample:

1. Generate a new Package.
2. Select a Workflow and click **Add Line.**
3. Select **token_parsing_demo** from the Object Type drop down list. The following fields are displayed:
   - Filename
   - AutoComp
4. Type 'parse_test1.txt' in the Filename field.
5. Click on the auto-complete box in the AutoComp field. The following Validation window opens, displaying the contents of the 'parse_test1.txt' file.



Configuring Text Fields

Text fields displayed on a single line. Text fields can be configured to display the data according to a certain format. For example, you can configure a text field to accept and format a ten digit telephone number or display a specific number of decimal places for a percentage.

This section covers the following topics:

- *Creating a Text Field Validation Overview*
- *Available Text Data Masks*
- *Customizing the System Text Data Masks*
- *Creating a Custom Data Mask*

Creating a Text Field Validation Overview

To create a text field Validation:

1. Open the Validation window in the Workbench.
2. In the Name field, enter the name of the Validation.

3. From the Component Type field, select **Text Field**.
4. From the Data Mask field, select the data mask that represents the desired format for the field.

   See *"Available Text Data Masks"* on page 326 for additional details.
5. (Optional) Configure the selected data mask.

   See *"Customizing the System Text Data Masks"* on page 328 for additional details.
6. Click **OK**.

Available Text Data Masks

The Mercury IT Governance Center includes a number of preconfigured data masks that can be used when creating text field Validations. Each of these data masks can be configured to meet your specific data requirements. *Table B-7* defines the data masks delivered with Mercury ITG.

Table B-7. Data Mask Formats

Data Mask	Description
Alphanumeric	Field allows all alphanumeric characters. The maximum field length for fields using this Validation can be specified.
Alphanumeric Uppercase	Field allows alphanumeric characters and formats all characters as uppercase text. The maximum field length for fields using this Validation can be specified.
Numeric	Field allows only numeric characters. The following characteristics can be specified for this data mask: • Range of values (maximum and minimum) allowed for this field • Whether or not a zero is displayed when data is not entered into the field • Whether or not group separators (such as a comma) are used within large numbers • How negative numbers are displayed • Number of decimal places See *"Customizing the Numeric Data Mask"* on page 328 for details.

Table B-7. Data Mask Formats [continued]

Data Mask	Description
Currency	Field allows only numeric characters and is used to display currency data. The following characteristics can be specified for this data mask: • Range of values (maximum and minimum) allowed for this field • Whether or not a zero is displayed when data is not entered into the field • Whether or not group separators (such as a comma) are used within large numbers • How negative numbers are displayed • Number of decimal places See *"Customizing the Currency Data Mask"* on page 330 for details.
Percentage	Field allows only numeric characters and is used to display percentages. The following characteristics can be specified for this data mask: • Range of values (maximum and minimum) allowed for this field • Whether or not a zero is displayed when data is not entered into the field • Whether or not group separators (such as a comma) are used within large numbers • How negative numbers are displayed • Number of decimal places See *"Customizing the Percentage Data Mask"* on page 332 for details.
Telephone	Field allows only numeric characters and is used to display telephone numbers. The following characteristics can be specified for this data mask: • Format—specify how many digits are included, and what delimiter should be used between groups of numbers. For example, you can select to use dashes (-) rather than periods (.) between numbers: 555-555-5555 or 555.555.5555. • Maximum and minimum number of digits See *"Customizing the Telephone Data Mask"* on page 334 for details.

Table B-7. Data Mask Formats [continued]

Data Mask	Description
Custom	Field allows a range of custom inputs. You can customize the field to accept digits, letters, spaces, and custom delimiters. See *"Creating a Custom Data Mask"* on page 336 for details.

Customizing the System Text Data Masks

Each data mask that is included in Mercury ITG can be customized. The following sections provide additional details on modifying the data masks:

- *Customizing the Numeric Data Mask*
- *Customizing the Currency Data Mask*
- *Customizing the Percentage Data Mask*
- *Customizing the Telephone Data Mask*

Customizing the Numeric Data Mask

The numeric data mask allows only numeric characters. When creating a validation using this data mask, the following characteristics can be specified:

- Range of values (maximum and minimum) allowed for this field
- Whether or not a zero is displayed when data is not entered into the field
- Whether or not group separators (such as a comma) are used within large numbers
- How negative numbers are displayed
- Number of decimal places

Figure B-8 shows the fields that can be configured for this data mask. *Table B-8* defines these fields.

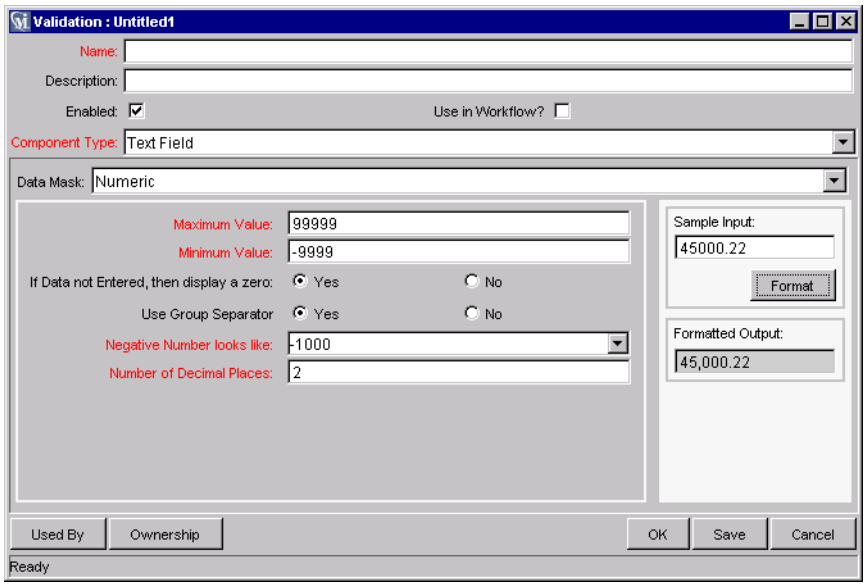


Figure B-8 Validation window for the numeric data mask

Table B-8. Fields for configuring the numeric data mask for text fields

Field	Description
Maximum Value	Largest value allowed for this field. This can be a positive or negative number.
Minimum Value	Smallest value allowed for this field. This can be a positive or negative number.
If Data not Entered, then display a zero	Determines if the field should display a zero when no data is entered.
Use Group Separator	Determines if the field should use a group separator (such as a comma) to divide characters within large numbers. For example: 1000000 versus 1,000,000. The character used for the separator defaults based on the machine's local, but can be configured in the Regional Settings dialog in the Workbench. Select **Edit > Regional Settings** to access this window.

Table B-8. Fields for configuring the numeric data mask for text fields

Field	Description
Negative Number looks like	Determines the appearance of negative numbers. There are four options available: • (1000)—parenthesis and black text • (1000)—parenthesis and red text • -1000—minus sign (-) and black text • -1000—minus sign (-) and red text
Number of Decimal Places	Determines the number of allowed decimal places. Users will only be able to enter up to this number of digits beyond the decimal place.

To view your customized data mask:

1. In the Sample Input field, enter the digits that you would like to see formatted.
2. Click **Format**.

The digits are formatted according to your settings and displayed in the Formatted Output field.

Customizing the Currency Data Mask

The currency data mask allows only numeric characters and is used to display currency data. When creating a validation using this data mask, the following characteristics can be specified:

- Range of values (maximum and minimum) allowed for this field
- Whether or not a zero is displayed when data is not entered into the field
- Whether or not group separators (such as a comma) are used within large numbers
- How negative numbers are displayed
- Number of decimal places

Figure B-9 shows the fields that can be configured for this data mask. *Table B-9* defines these fields.

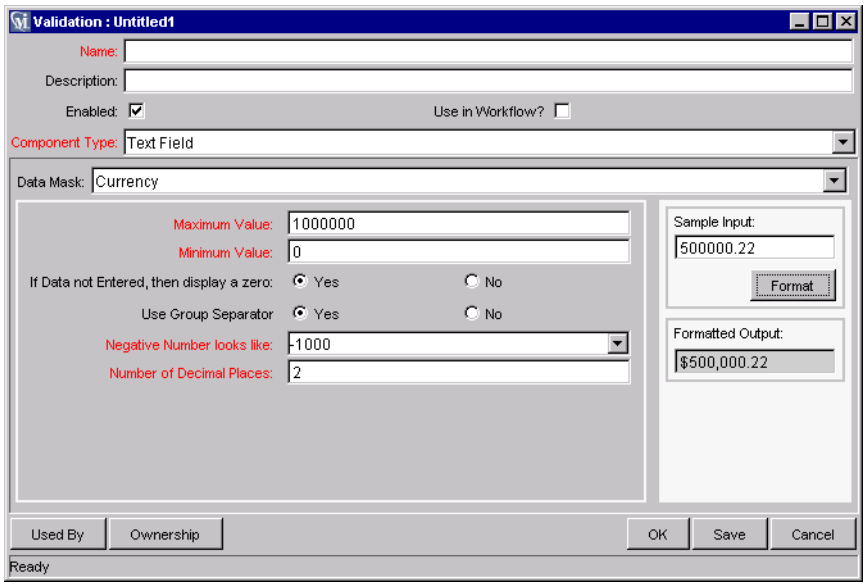


Figure B-9 Validation window for the currency data mask

Table B-9. Fields for configuring the currency data mask for text fields

Field	Description
Maximum Value	Largest value allowed for this field. This can be a positive or negative number.
Minimum Value	Smallest value allowed for this field. This can be a positive or negative number.
If Data not Entered, then display a zero	Determines if the field should display a zero when no data is entered.
Use Group Separator	Determines if the field should use a group separator (such as a comma) to divide characters within large numbers. For example: 1000000 versus 1,000,000. The character used for the separator defaults based on the machine's local, but can be configured in the Regional Settings dialog in the Workbench. Select **Edit > Regional Settings** to access this window.

Table B-9. Fields for configuring the currency data mask for text fields

Field	Description
Negative Number looks like	Determines the appearance of negative numbers. There are four options available: • (1000)—parenthesis and black text • (1000)—parenthesis and red text • -1000—minus sign (-) and black text • -1000—minus sign (-) and red text
Number of Decimal Places	Determines the number of allowed decimal places. Users will only be able to enter up to this number of digits beyond the decimal place.

To view your customized data mask:

1. In the Sample Input field, enter the digits that you would like to see formatted.
2. Click **Format**.

The digits are formatted according to your settings and displayed in the Formatted Output field.

Note

The INSTALLATION_CURRENCY server parameter dictates which currency symbol is displayed in the field. This parameter also dictated the position of the text in the field. For example:

```
INSTALLATION_CURRENCY=$;RIGHT
```

will right-align the text using a dollar sign.

Contact your system administrator for help with changing this setting.

Customizing the Percentage Data Mask

The percentage data mask allows only numeric characters and is used to display percentages. When creating a validation using this data mask, the following characteristics can be specified:

- Range of values (maximum and minimum) allowed for this field
- Whether or not a zero is displayed when data is not entered into the field

- Whether or not group separators (such as a comma) are used within large numbers
- How negative numbers are displayed
- Number of decimal places

Figure B-10 shows the fields that can be configured for this data mask. *Table B-10* defines these fields.

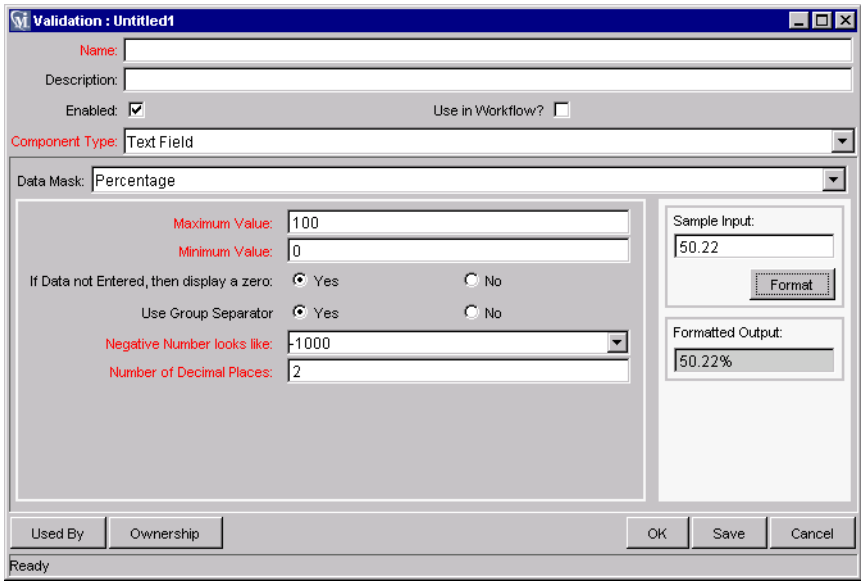


Figure B-10 Validation window for the percentage data mask

Table B-10. Fields for configuring the percentage data mask for text fields

Field	Description
Maximum Value	Largest value allowed for this field. This can be a positive or negative number.
Minimum Value	Smallest value allowed for this field. This can be a positive or negative number.
If Data not Entered, then display a zero	Determines if the field should display a zero when no data is entered.

Table B-10. Fields for configuring the percentage data mask for text fields

Field	Description
Use Group Separator	Determines if the field should use a group separator (such as a comma) to divide characters within large numbers. For example: 1000000 versus 1,000,000. The character used for the separator defaults based on the machine's local, but can be configured in the Regional Settings dialog in the Workbench. Select **Edit > Regional Settings** to access this window.
Negative Number looks like	Determines the appearance of negative numbers. There are four options available: • (1000)—parenthesis and black text • (1000)—parenthesis and red text • -1000—minus sign (-) and black text • -1000—minus sign (-) and red text
Number of Decimal Places	Determines the number of allowed decimal places. Users will only be able to enter up to this number of digits beyond the decimal place.

To view your customized data mask:

1. In the Sample Input field, enter the digits that you would like to see formatted.
2. Click **Format**.

The digits are formatted according to your settings and displayed in the Formatted Output field.

Customizing the Telephone Data Mask

The percentage data mask allows only numeric characters and is used to display telephone numbers. When creating a validation using this data mask, the following characteristics can be specified:

- Format—specify how many digits are included, and what delimiter should be used between groups of numbers. For example, you can select to use dashes (-) rather than periods (.) between numbers. For example, 555-555-5555 or 555.555.5555.
- Maximum and minimum number of digits

Figure B-11 shows the fields that can be configured for this data mask. *Table B-11* defines these fields.

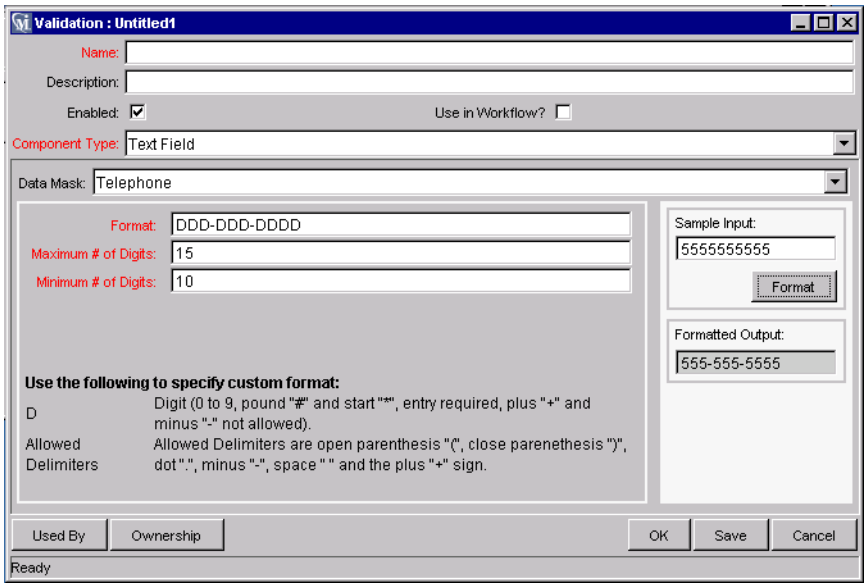


Figure B-11 Validation window for the telephone data mask

Table B-11. Fields for configuring the telephone data mask for text fields

Field	Description
Format	The rule that dictates how the digits are formatted, including any spaces or delimiters. The following delimiters are allowed in the format definition: • Open and close parentheses ( ) • Period (.) • Dash (-) • Space • Plus sign (+) See *Table B-12 on page 336* for a few examples of different Telephone formats.
Maximum # of Digits	Largest number of digits that will be accepted in this field.
Minimum # of Digits	Smallest number of digits that will be accepted in this field. If the user enters fewer than this number of digits in the field and then tries to move from the field, he will receive an error.

Table B-12. Sample telephone data mask formats

Format Rule	Text Entered By User	Sample Formatted Output
D-DDD-DDD-DDDD	15555555555	1-555-555-5555
DDD DDD DDDD	5555555555	555 555 5555
(DDD) DDD-DDDD	5555555555	(555) 555-5555

To view your customized data mask:

1. In the Sample Input field, enter the digits that you would like to see formatted.
2. Click **Format**.

The digits are formatted according to your settings and displayed in the Formatted Output field.

Note

Special behavior applies to the extra characters, if your format is defined to allow a range of entries. Extra characters will always be grouped with the first set of characters. For example, if the telephone data mask is configured with a minimum of 10 characters and a maximum of 15 characters, then the following behavior is expected:

```
Format: DDD-DDD-DDDD
Min: 10
Max: 15
```

Input: `1234567890`
Output: `123-456-7890`

Input 2: `12345678901`
Output 2: `1234-567-8901`

Creating a Custom Data Mask

A custom data mask can be defined that will allow a range of inputs and format them to your specification. You can customize the field to accept digits, letters, spaces, and custom delimiters.

Figure B-11 shows the fields that can be configured for this data mask.

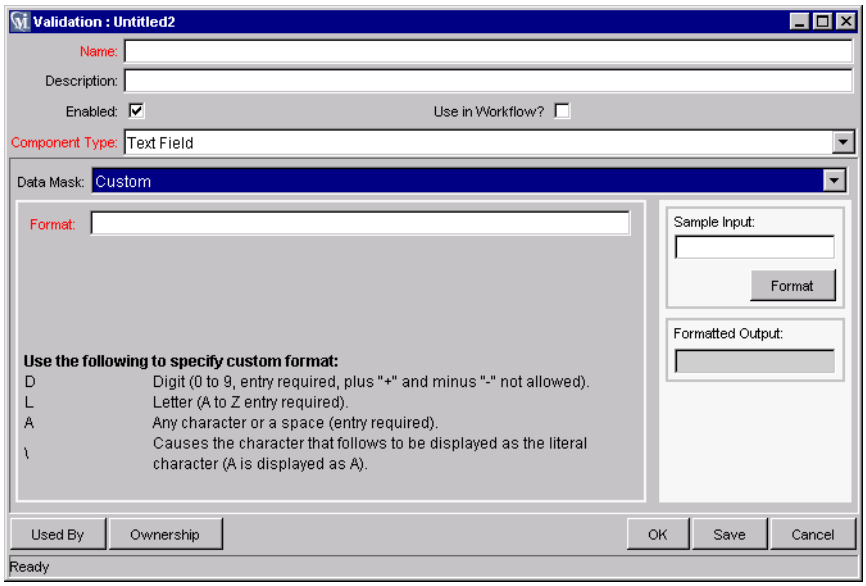


Figure B-12 Validation window for the custom data mask

To configure a custom format, enter a combination of symbols into the Format field. This field can accept the following entries:

- D—Specifies a required digit between 0 and 9.
- L—Specifies a required letter between A and Z.
- A—Specifies a required character or space.
- \ (backslash)—Causes the character that follows to be displayed as the literal character. For example: "\A" will be displayed as "A"

Table B-13 displays some examples of custom formats.

Table B-13. Sample custom data mask formats

Format Rule	Text Entered By User	Formatted Output
DDD\-DD\-DDDD	555555555	555-55-5555
AA\-DDD	BC349	BC-349

To view your customized data mask:

1. In the Sample Input field, enter the digits that you would like to see formatted.
2. Click **Format**.

The digits are formatted according to your settings and displayed in the Formatted Output field.

Using Directory and File Choosers

Directory and File Choosers are only used with Mercury Change Management Object Types. The following sections discuss them in more detail:

- *Directory Chooser*
- *File Chooser*

Directory Chooser

The Directory Chooser field can be used to select a valid directory from an Environment. Mercury Change Management connects to the first Source Environment on a Workflow and allows navigation through the directory structure and the selection of a directory from the list.

When implementing the Directory Chooser, note the following:

- The Directory Chooser field can only be used on an Object Type.
- On every Object Type that a Directory Chooser is chosen, it is also necessary to have a field whose token is **P_FILE_LOCATION** and whose validation is **DLV - File Location**. The possible values for this field are **Client** and **Server**. If **Client** is chosen, the Directory Chooser connects to the Client Base Path of the Source Environment. If **Server** is chosen, the Directory Chooser connects to the Server Base Path of the Source Environment.

File Chooser

A File Chooser field can be used by Object Types to select a valid file from an Environment. Mercury Change Management connects to the first Source Environment on a Workflow and provides the ability to view all files within a specific directory and select one from the list.

On every Object Type that a File Chooser is chosen, it is necessary to define the following fields:

- The first is a field for the File Location for the directory chooser, described in the previous section.
- The second is a field whose token is 'P_SUB_PATH'. This field is the directory from which the file is selected and is usually a Directory Chooser field.

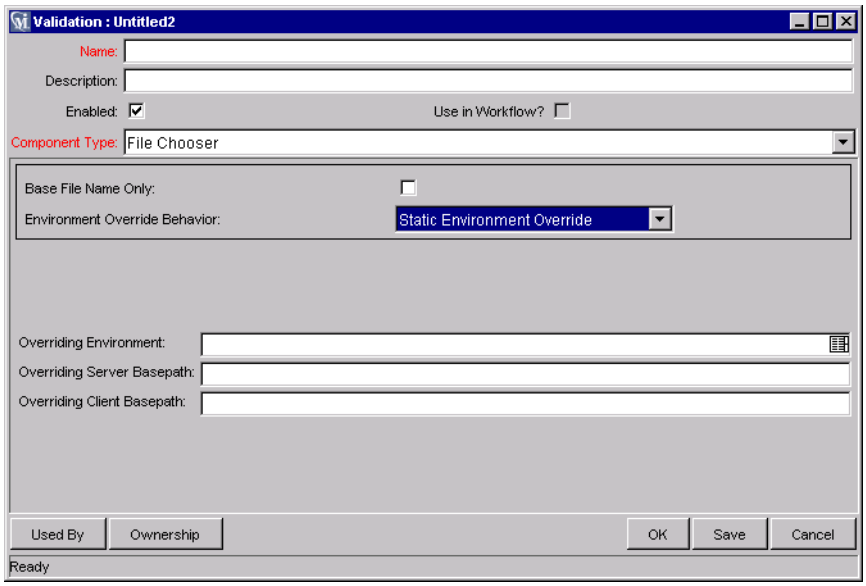


Figure B-13 Validation Window for Static Environment Override in File Chooser.

Table B-14. File Chooser Field

Field	Definition
Base File Name Only	Defines whether the base file name only (without its suffix) or the complete name is displayed.
Environment Override Behavior	Used to select files from a specific environment other than the default environment.

The Environment Override Behavior drop down list contains three options: **Default Behavior**, **Static Environment Override**, and **Token-Based Environment Override**.

Static Environment Override provides the ability to override one Environment at a time. The fields for Static Environment Override are pictured in *Figure B-13* and described in *Table B-15*.

Table B-15. Static Environment Override

Field	Definition
Overriding Environment	Selects the Environment to be overridden.
Overriding Server Basepath	The server basepath of the Environment may be overridden.
Overriding Client Basepath	The client basepath of the Environment may be overridden.

Token-based Environment Override provides the ability to select a token that will resolve to the overriding Environment. The fields for **Token-based Environment Override** are shown in *Figure B-14* and defined in *Table B-15*.

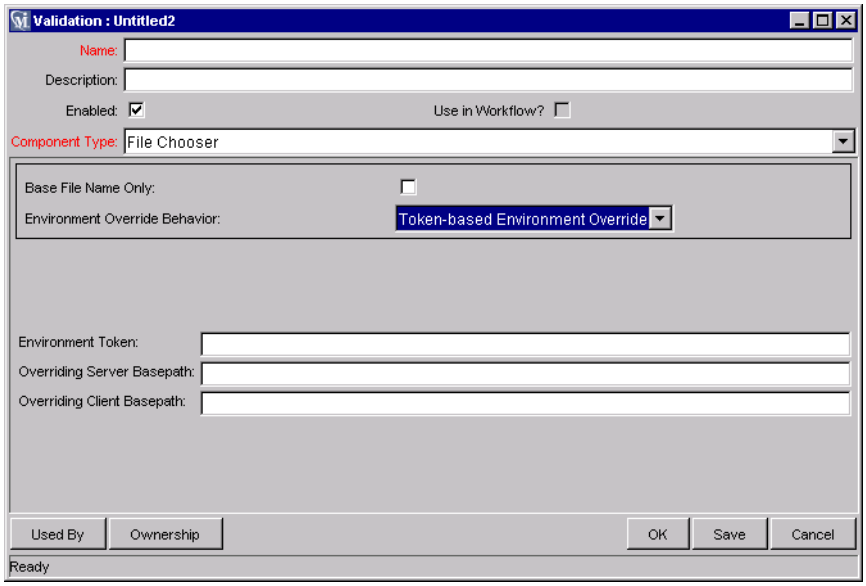


Figure B-14 Validation Window for Token-Based Environment Override in File Chooser.

Figure B-15 Token-Based Environment Override

Field	Definition
Environment Token	Select the token that will resolve to the overriding Environment.
Overriding Server Basepath	The server basepath of the Environment that is to be resolved by the token may be overridden.
Overriding Client Basepath	The client basepath of the Environment that is to be resolved by the token may be overridden.

Date Field Formats

Date fields can accept a variety of formats. The current date field Validations are separated into two categories: all systems, and systems using only the English language. These formats are defined in Table 3-14.

Table 0-5. Date Field

Field		Definitions
Name	Systems	
Date Format	All	The format for the date part of the field. Choices are: · Long - “January 2, 1999". · Medium - “02-Jan-99”. · Short - “1/2/99”. · None - no date is displayed.
Date Format	English Only	The format for the date part of the field. Choices are: · MM/DD/YY (6/16/99). · DD-MON-YY (16-Jun-99). · MONTH DD, YYYY (June 16, 1999). · Day, Month DD, YYYY (Monday, June 16, 1999). · DD-MON (16-JUN) - Defaults to current year. · DD-MON-YYYY (16-JUN-1999). · MM-DD-YYYY (06-16-1999). · MM-DD-YY (06-16-99). · DD [Defaults to the current month and year]. · MM/DD (06/16) - Defaults to current year. · MM/DD/YYYY (06/16/1999).

Table 0-5. Date Field

Field		Definitions
Name	Systems	
Time Format	All	The format for the time part of the field. Choices are: · Long - the time is displayed as “12:00:00 PM PST”. · Medium - the time is displayed as “12:00:00 PM”. · Short - the time is displayed as “12:00 PM”. · None - no time is displayed.

Creating 1800 Character Text Areas

Standard Text Areas are either 40 or 200 characters. You can, however, create a Text Area Validation with a character length of 1800.

To create a Validation with a character length of 1800:

1. Open the Validation Workbench.
2. Search for “**Text Area - 1800**.”
3. In the results tab, select **Text Area - 1800.**
4. Click **Copy**.
5. Rename the Validation.

The new Text Area Validation (with a length of 1800) can be used when defining a custom field in the product.

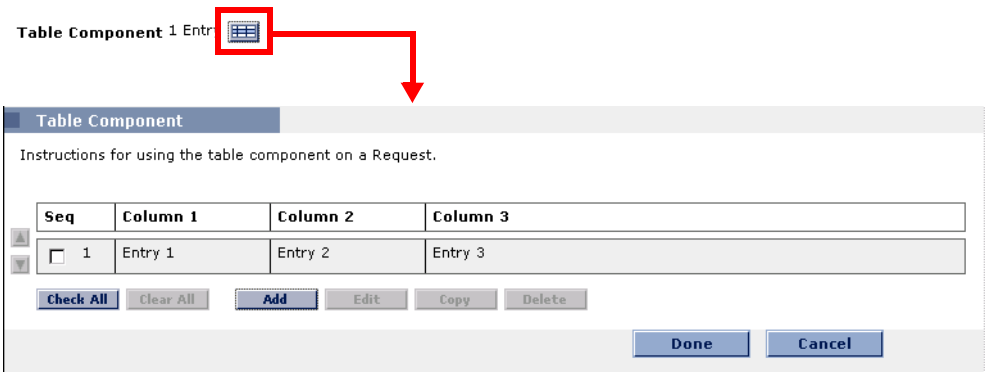
You can only create a Text Field or Area of length 40, 200, or 1800.

Configuring the Table Component

The table component is used to enter multiple records into a single field on a Request. The table component can be configured to include multiple columns of varied data types. Additionally, this component supports rules for populating elements within the table and provides functionality for capturing column totals.

1. Click the Table Component icon to open the Table Component entry page.

2. Add, edit, or delete entries in the list.



Fields of this component can only be added to Request Types, Request Header Types and Request User Data.

To configure and use a Table Component:

1. *Define the Table Component in the Validation Workbench*
2. *Add the Table Component to a Request Type*

Example

ACME creates a Request Type to request quotes and parts for hardware. Each entry of this type has four elements: Part, Sub-Type, Part Number, and Unit Price. ACME creates a Table Component field called Hardware Information to collect this information.

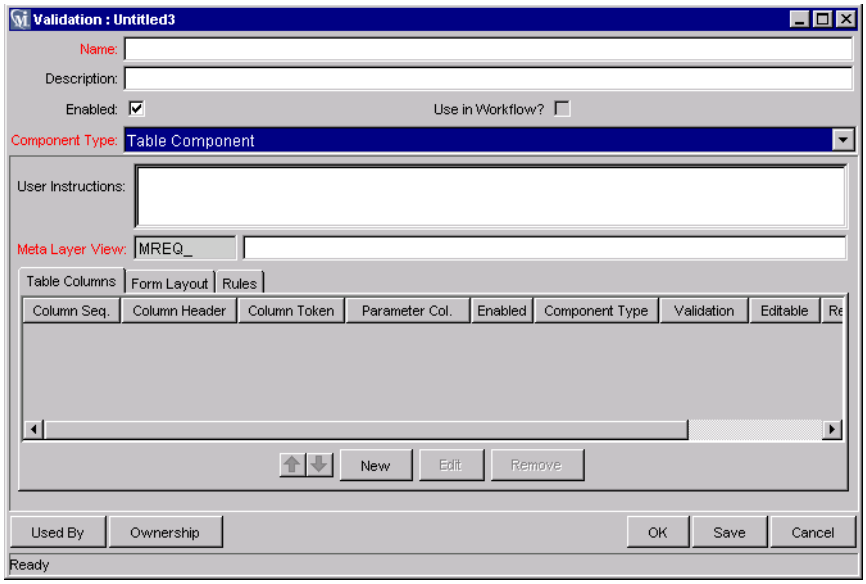
When the user logs a request for new hardware, the Request displays the Hardware Information field. The user opens the field. He selects a Part, which triggers a rule to populate the Part Number and Unit Price. He submits the Request, which now contains all of the information required to successfully order the hardware.

Define the Table Component in the Validation Workbench

To create a Table Component field:

1. Open the Validation Workbench in the Configuration screen group.
2. Click **New Validation**.

   The Validation window opens.
3. Select **Table Component** from the Component Type drop down list.



4. Enter a Validation Name and Description.
5. Enter any User Instructions.

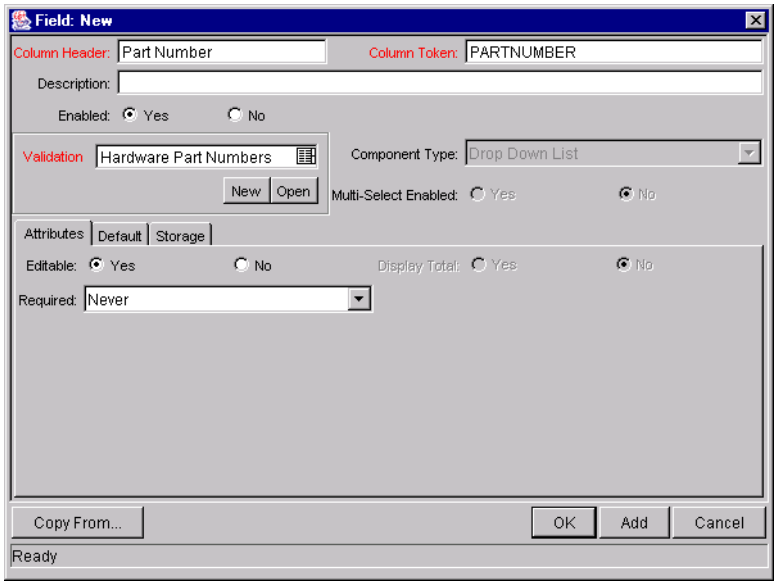
   This text will appear on the top of the table entry page.
6. Create the Table Columns.

   a. Click **New** in the **Table Columns** tab. The Field window opens.

   b. Define the type of information that will be stored in that column's entries. This may require you to create a new Validation for the column.

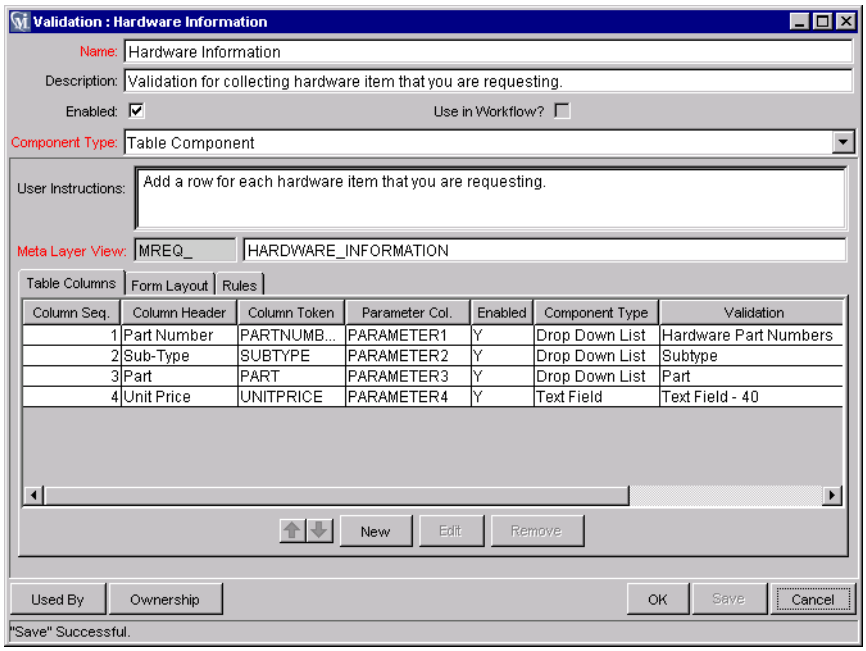
Note

File attachments can not be used in a Table component column.



c. Specify the **Attributes** (Editable or Required) and any **Default** behavior.

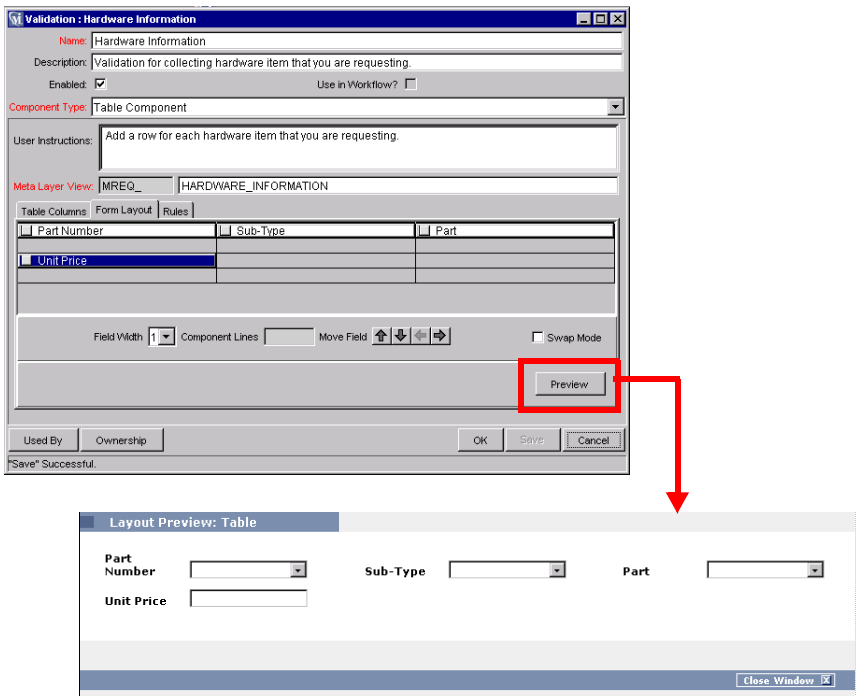
d. Click **Add** to save the column information and add another column. When you are finished adding columns, click **OK** to close the Field window.



7. Configure the Form Layout.

   a. Click the **Form Layout** tab.

   b. Select the fields and move their positions using the arrow buttons.



c. Click Preview to see a representation of the final positioning.

Note that the Preview loads a window in the Workbench, but the actual table component will only be available to users in the standard interface (HTML).

8. Configure any Table logic in the **Rules** tab.

Rules are used for advanced defaulting behavior and calculating column totals.

a. Click the **Rules** tab.

b. Click **New** to define a new rule.

See *"Creating a Table Rule"* on page 348 for detailed instructions.

9. Click **OK** to save the Validation.

The new Table Component field can be included on a Request Type, Request Header Type or Request User Data field.

Creating a Table Rule

Table rules are configured in the same manner as advanced Request Type rules. Essentially, you can configure fields (columns) in the table to default to certain values based on an event or value in another field in the table. Because the table component rules are configured using a SQL statement, you are given enormous flexibility for the data that is populated in the table cells.

Table rules are configured using the **Rules** tab on the Validation window.

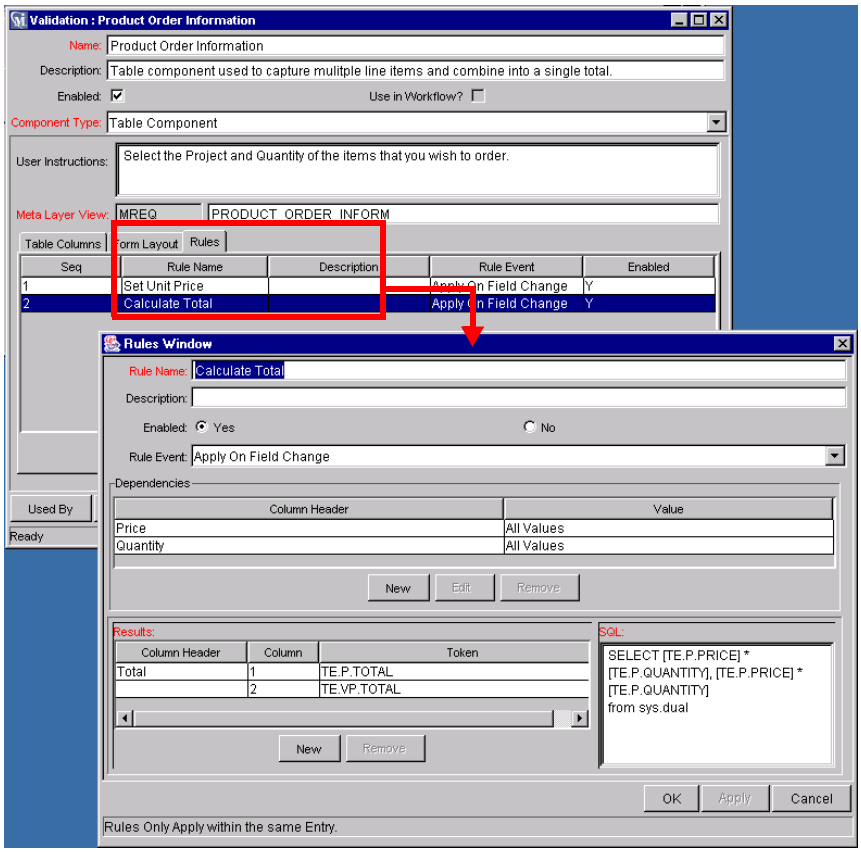


Figure B-16 Rules window accessed from the Rules tab

Example: Using a Table Component on an Order Form

The following example illustrates the table component rules functionality.

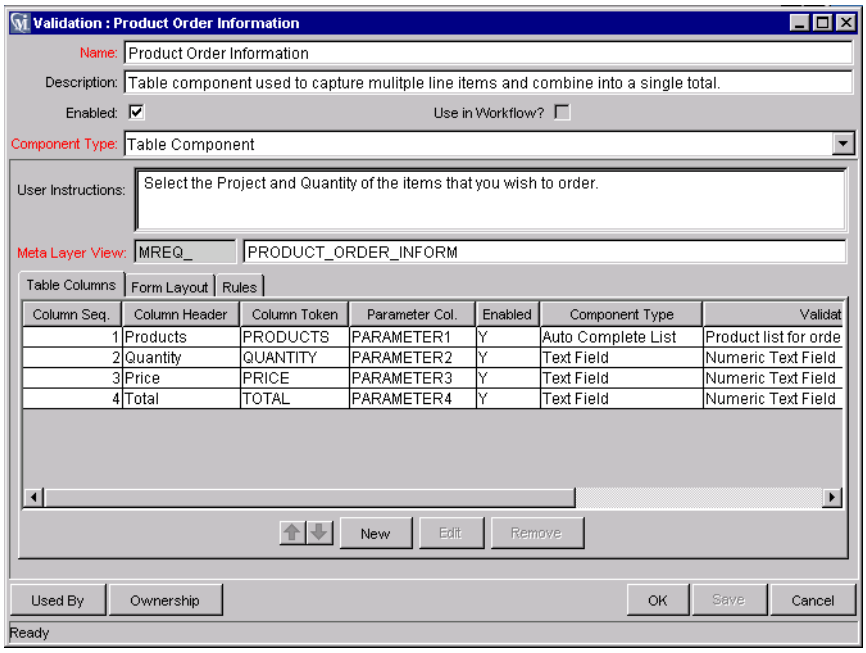
ACME, Inc. uses a Request for creating and tracking employee computer hardware equipment orders. ACME has included a table component field on their Request Type for gathering the order information. When the employee selects a Product, the Unit Price is automatically updated. Then, when they

update the Quantity, the total line cost is automatically calculated and displayed in the table.

To enable this functionality, ACME first has to configure a new Validation with the following specifications:

Table B-16. Example - Table Component Validation Settings

Setting	Value / Description
Validation Name	Product Order Information
Component Type	Table Component
Column 1	Column Header = Products Column Token = PRODUCTS Validation = Auto complete list with the following list values: PC, MOUSE, MONITOR, KEYBOARD
Column 2	Column Header = Quantity Column Token = QUANTITY Validation = Numeric Text Field
Column 3	Column Header = Price Column Token = PRICE Validation = Numeric Text Field
Column 4	Column Header = Total Column Token = TOTAL Validation = Numeric Text Field



Once the Validation's columns have been defined, the Rules can be configured:

Rule 1: Set Unit Price.

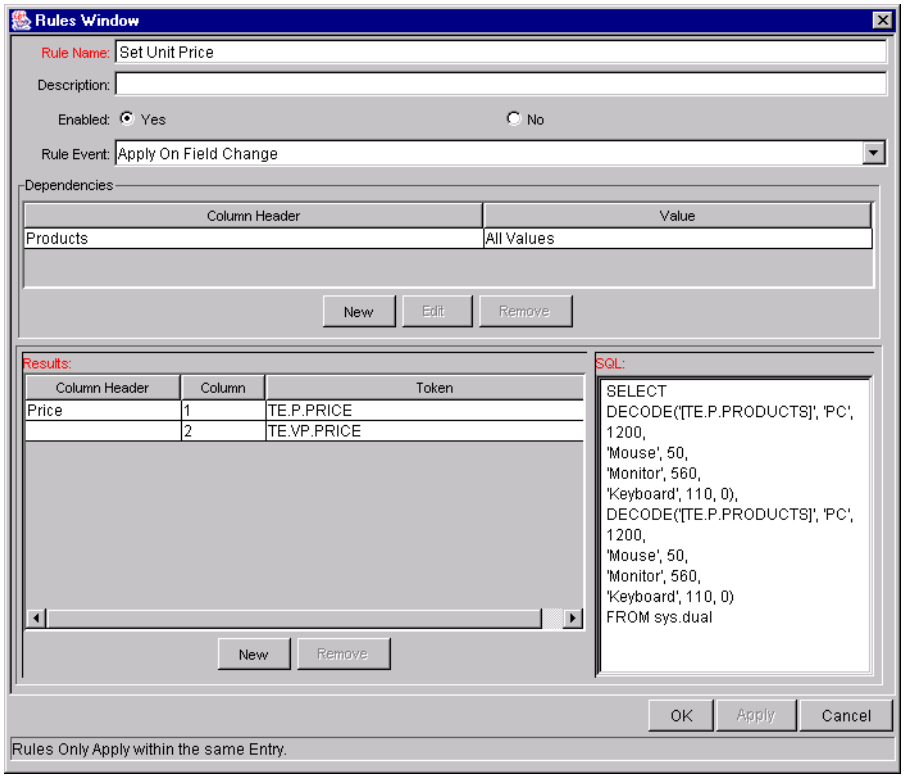
ACME uses the following rule to set the default unit price in the Price cell based on the Product selection.

Table B-17. Example - Set Unit Price Rule Settings

Setting	Value / Description
Rule Name	Set Unit Price
Rule Event	Apply on Field Change
Dependencies	Column = Products All Values = Yes
Results	Column Header = Price

Table B-17. Example - Set Unit Price Rule Settings

Setting	Value / Description
SQL	SELECT DECODE('[TE.P.PRODUCTS]', 'PC', 1200, 'Mouse', 50, 'Monitor', 560, 'Keyboard', 110, 0), DECODE('[TE.P.PRODUCTS]', 'PC', 1200, 'Mouse', 50, 'Monitor', 560, 'Keyboard', 110, 0) FROM sys.dual



Rule 2: Set Unit Price.

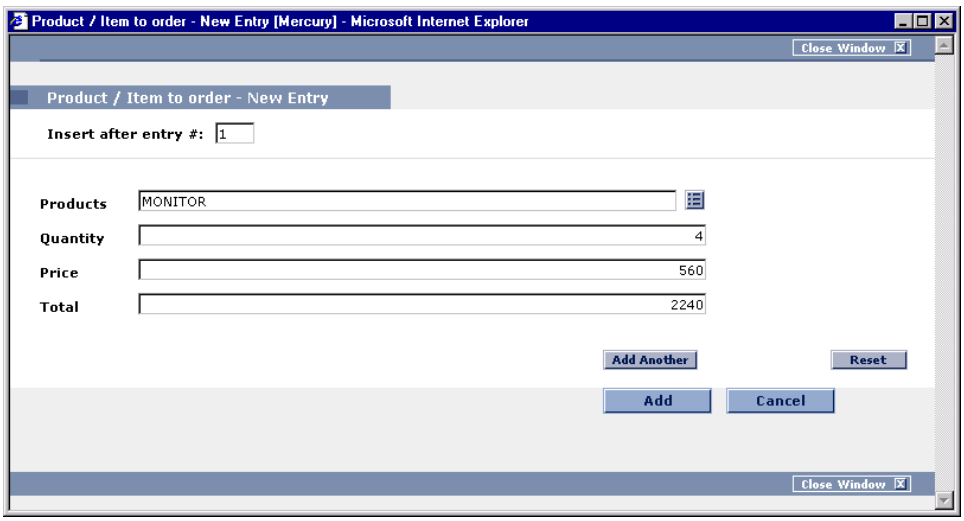
ACME uses the following rule to set the calculate and display the total line price in the Total column based on the values in the Products and Quantity cells.

Table B-18. Example - Calculate Total Rule Settings

Setting	Value / Description
Rule Name	Calculate Total
Rule Event	Apply on Field Change
Dependencies	Column = Price [All Values = Yes] Column = Quantity [All Values = Yes]
Results	Column Header = Total
SQL	SELECT [TE.P.PRICE] * [TE.P.QUANTITY], [TE.P.PRICE] * [TE.P.QUANTITY] from sys.dual

Using the Table Component

Add a field to a Request Type that is validated by this Table Component Validation. When a user opens the field to enter information, the table rules will be applied to each row that is created.



Tokens in the Table Components

Each column included in the table component has an associated Token. These Tokens can be used in the same manner as other field tokens, such as for commands, notifications or advanced field defaulting. See *Commands and*

Tokens Guide and Reference for details on referencing Tokens related to Table Components.

Calculating Column Totals

You can configure columns that are validated by a number to calculate the total for that column. This is configured in the Validation's Field window. The following example illustrates how to configure a column to calculate and display the column total.

ACME, Inc. uses a Request for creating and tracking simple employee equipment orders. ACME has included a table component field on their Request Type for gathering the order information. Employee enter the Purchase Items and Cost for each item. The table component automatically calculates the total cost for the Cost column.

ACME creates a Validation with the following settings:

- Component Type = **Table Component**.
- Column 1 = Purchase Item (text field)
- Column 2 = Cost (number). In the Field window for the Cost column, select Display Total = **Yes**. The Display Total field is only enabled if the field's validation is a number.

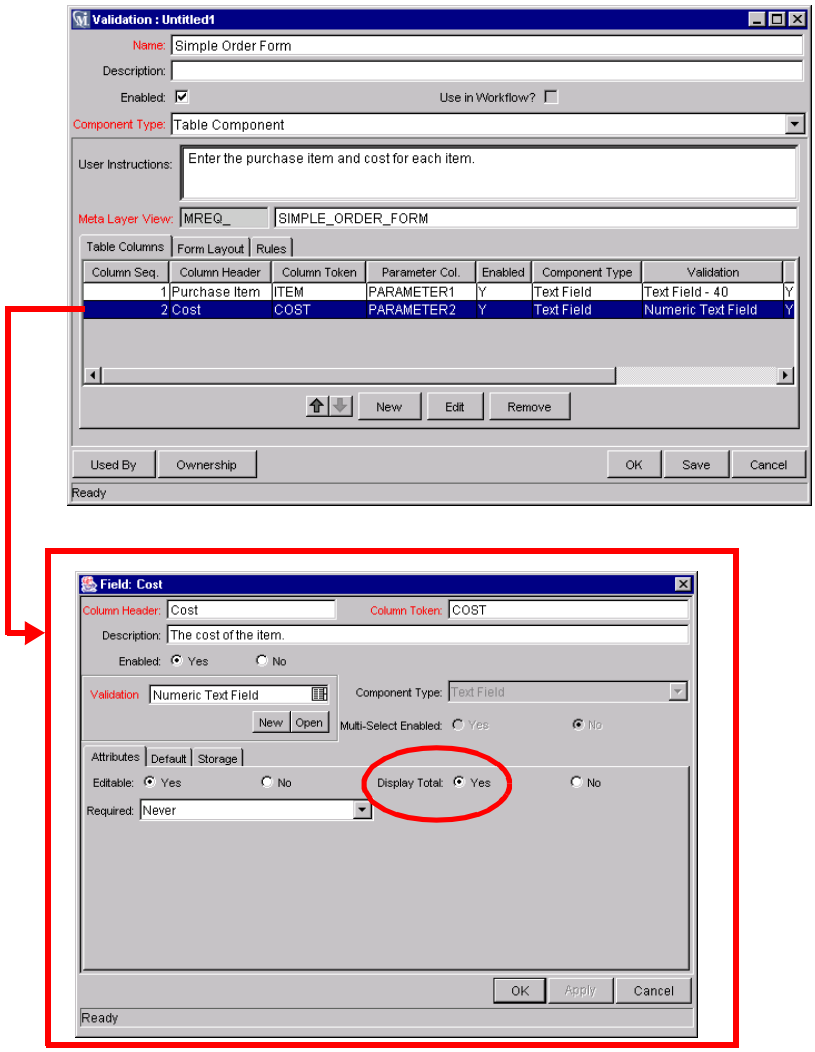


Figure B-17 Sample validation for a Simple Order table component.

ACME includes adds a field to their Order Request Type that uses this Validation. When a user creates a Request using that Request Type, he can click on the table component icon next to the field to open the order form. The total for the Cost column is displayed at the bottom of the table.

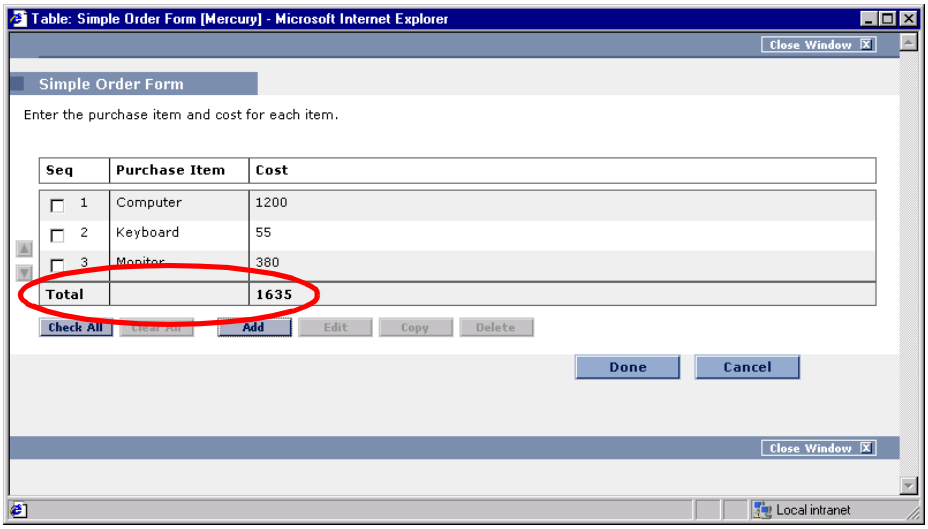


Figure B-18 Sample table component displaying a column total.

Add the Table Component to a Request Type

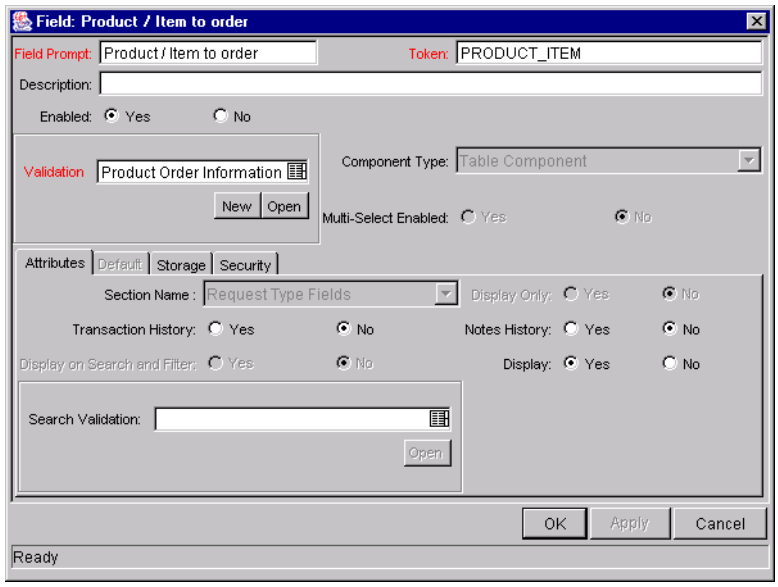
Table Component fields can be included on a Request Type, Request Header Type or Request User Data field.

To add a Table Component field to a Request Type:

1. Open the Request Type window.
2. Click New in the **Fields** tab.

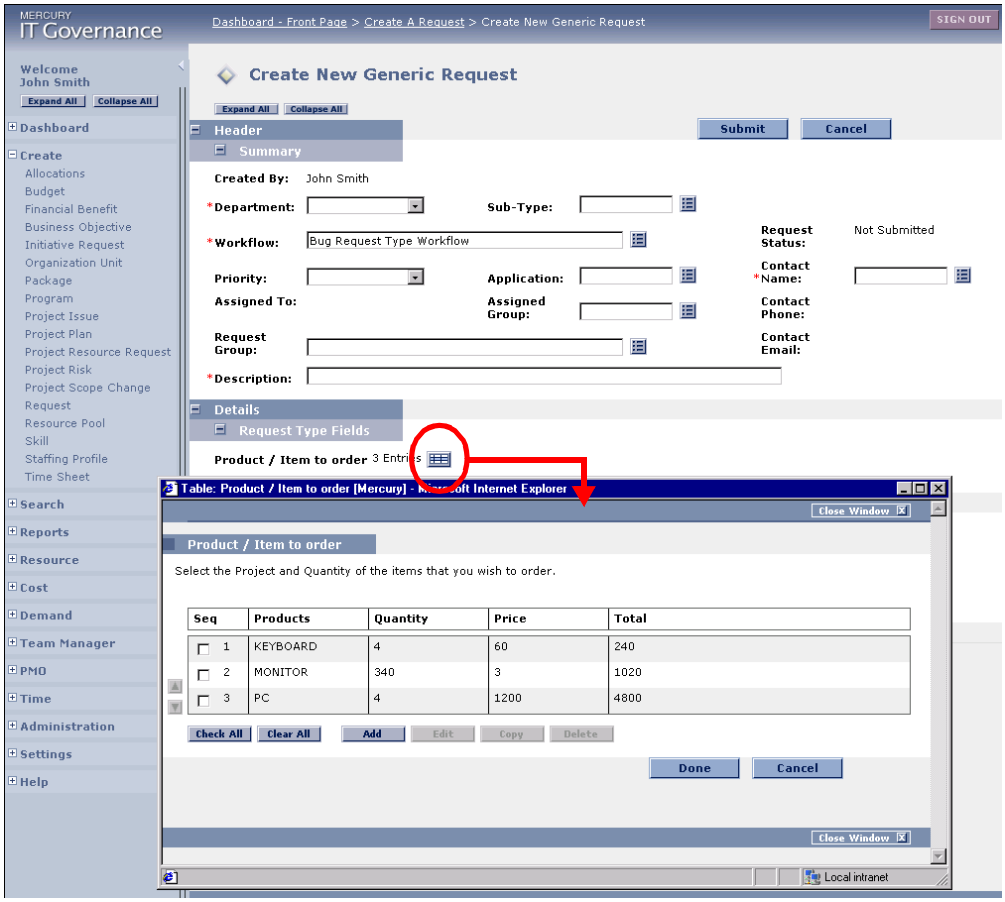
   The Field window opens.
3. Enter the Field Prompt, Token, and Description.
4. In the Validation field, select a table component Validation.

   If you have not created a table component Validation, click **New** to create one. See *"Define the Table Component in the Validation Workbench"* on page 344 for instructions.



5. Click **OK** to add the field to the Request Type.
6. Save the Request Type.

The table component field will now appear on Requests of this Request Type.



Package and Request Group Validations

Two particular entity-specific Validations can be accessed in the Workbench without entering the Validations screen group:

- *Package and Request Groups*
- *Request Type Category*

Package and Request Groups

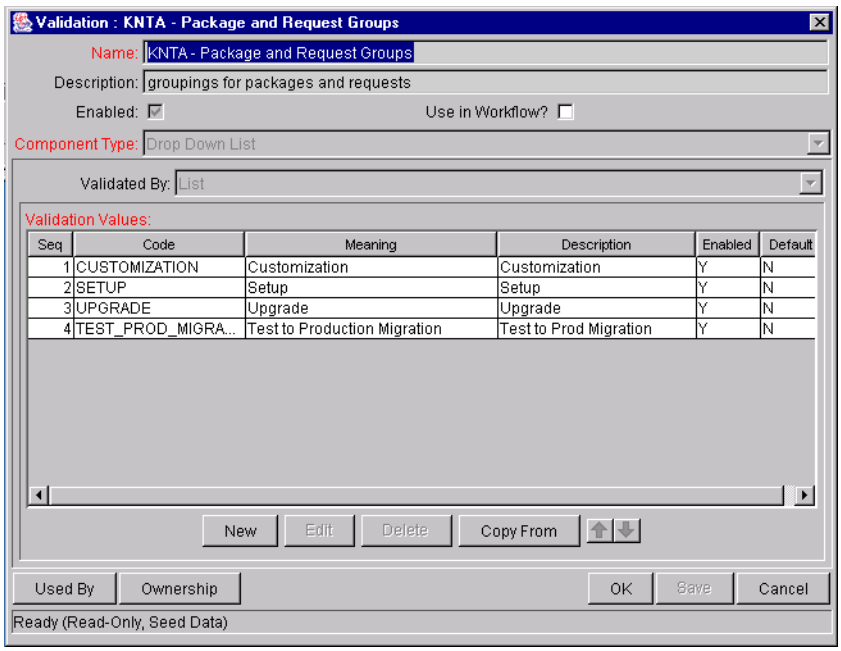
The KNTA-Package and Request Groups Validation can be accessed directly from the **Package** screen. To specify that a Package belongs to a new or unique Package Group that is not named in the auto-complete Validation list, it is not necessary to proceed through the Validation Workbench.

To access the KNTA-Package and Request Groups Validation window from the **Package** screen:

Select **New Package Group** from the **Package** menu. The Validation window will appear, listing the existing Mercury Change Management Package Groups.

Note

All users are granted read access to this screen, but only users with appropriate security privileges can alter the KNTA-Package and Request Groups Validation list.



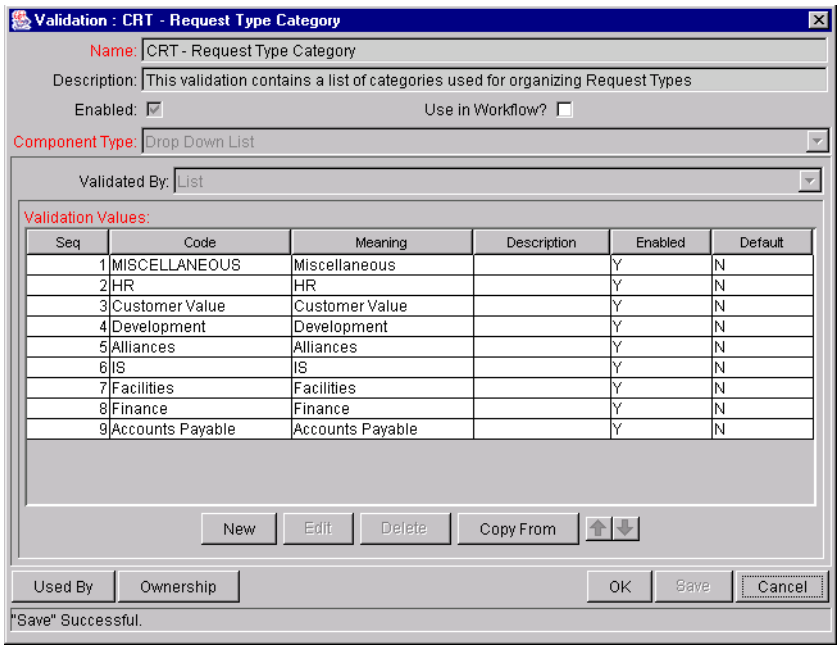
Request Type Category

The CRT - Request Type Category Validation can be accessed directly from the **Request Types** screen.

Access the CRT - Request Type Category Validation window from the **Request Types** screen by selecting **Request Type Category Setup** from the **Request Type** menu. The Validation window will appear, listing the existing Request Type Categories.

Note

All users are granted read access to this screen, but only users with appropriate security privileges can alter the CRT - Request Type Category Validation list.



Validation Special Characters

The Validation Name field for all Validations cannot contain a question mark (‘?’). The Workbench prevents this character from being entered into the field, but all previously configured Validation Names (Validations entered before Kintana release 4.5) should be checked and corrected.

System Validations

There are a number of Validations that are provided with Mercury IT Governance Center. Note that many of these validations may have been altered to better match your company’s specific business needs. Use the Validations

Report to get a list of all validations currently in your system. The report includes information on Validation values and commands.

Appendix C Tokens

While configuring certain features, it is often necessary to reference information that is undefined until the product is actually used a particular context. Instead of generating objects that are valid only in specific contexts, create variable objects that can be applied to a variety of contexts. These variables are called Tokens.

There are two types of Tokens: custom Tokens and standard Tokens. Standard Tokens are provided with the product. Custom Tokens are generated to suit specific needs. Each field of the following entities can be referenced as a custom Token:

- Object Types
- Request Types and Request Header Types
- Report Types
- User Data
- Workflow Parameters

In addition, numerous standard Tokens are available that provide other useful pieces of information related to the system. For example, there is one Token that represents the users currently logged onto the system.

For instructions on using Tokens and for a list of available system Tokens, see *Commands and Tokens Guide and Reference*.

Appendix D

User Data Creation and Processing

This appendix provides instructions for creating User Data fields for supported product entities. It also includes information on configuring advanced User Data field behavior and maintaining the User Data field definitions.

This appendix covers the following topics:

User Data Overview

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 every major entity, up to twenty User Data fields can be defined. These fields are displayed in the **User Data** tab for the specific 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.

User Data fields are available for each entity instance generated; the fields are available globally. For example, you can configure a Manager field to appear on the **User Data** tab in the User window, and then specify each user's manager when setting up their account.

For some entities, context-sensitive custom fields can be set up. For example, User Data fields could be defined for the Request entities that are only available when the priority of a Request is **Critical**.

The following entities support User Data functionality:

- Budgets
- Organizations
- Resource Pools
- Staffing Profiles
- Packages
- Package Lines
- Environments
- Environment Application
- Environment Refresh
- Requests
- Request Types
- Projects
- Tasks
- Security Groups
- Users
- Validation Values
- Workflows
- Workflow Steps
- Workflow Executions
- Workflow Decisions

Creating and Editing User Data

The following sections provide detailed instructions for creating and editing User Data:

- *Creating User Data Fields*
- *Copying a Field's Definition*
- *Editing User Data Fields*
- *Configuring User Data Field Dependencies*
- *Removing Fields*
- *Modifying the User Data Layout*

To configure User Data, you must have the **Config: Edit User Data** Access Grant.

Creating User Data Fields

To create a new User Data field:

1. Click the **Configuration** screen group and click the **User Data** screen.

   The User Data Workbench opens.

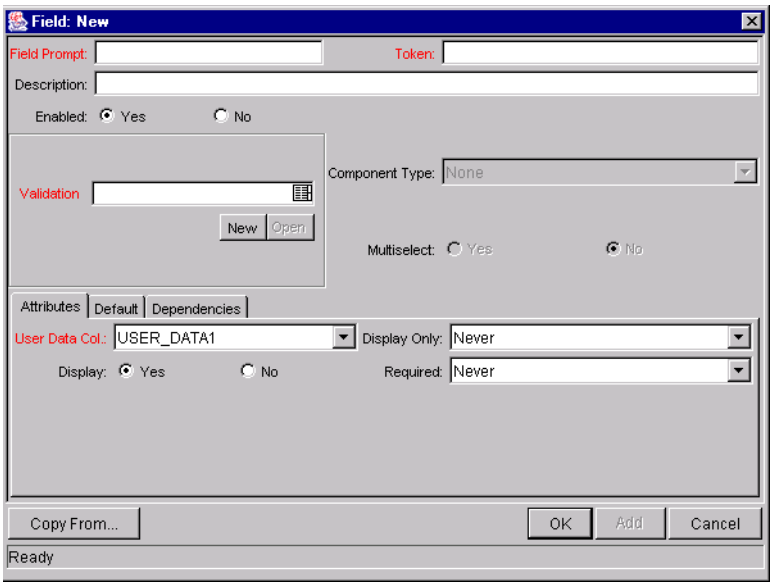
2. Search for the desired User Data Type from the **Query** tab and select it from the **Results** tab of the User Data Workbench.

3. Click **Open**.

   The User Data window opens.

4. Click **New** in the **Fields** tab.

   The Field: New window opens.



5. Enter the general information region fields for the User Data field.

   This includes entering a name in the Field Prompt field and selecting a Validation.

6. Click the **Attributes** tab to define the field's basic properties. This includes entering information described in the following table.

Field	Description
User Data Col	Determines 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	Determines whether the field is only displayed and cannot be updated. Select **Use Dependency Rules** to use the logic defined in the **Dependencies** tab.
Display	Determines if the user sees this field in the **User Data** tab.
Required	Determines 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.

Field	Description
Workbench Only	Determines whether or not the field is seen in the standard (HTML) interface.
Multi-Select Enabled	Determines whether or not the field allows users to select more than one entry. Only valid for fields with an auto-complete component for the Validation.
Display in Search and Filter Pages	Determines whether or not the field will be displayed in Search and Filter pages in the standard interface.

7. Click the **Defaults** tab to define the default value for that field. This includes entering information described 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.

8. Click the **Dependencies** tab to define the field dependent properties of the field. This includes entering information described 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** on the **Attributes** tab.

Field	Description
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** on the **Attributes** tab.

9. Click the **Security** tab to define which users can view or update this field. Only a few User Data types support the use of the **Security** tab.

   a. On the Security tab, click Edit.

   b. In the Edit Field Security window define the field access. This includes entering information described in the following table.

Field / Button	Description
Visible to all users	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. 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.
Editable by all users	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. 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.

Field / Button	Description
Enter a Security Group (drop down list)	To select the format for specifying users to grant visibility and editability 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. The choices are: • **Enter a Username** – select a specific user to grant visibility and/or editability to the field. The user must have an email address. • **Enter a Security Group** – select a specific Security Group to grant visibility and/or editability to the field. • **Enter a Standard Token** – select a standard token to grant visibility and/or editability to the field. • **Enter a User Defined Token** – select a user defined token to grant visibility and/or editability to the field. Selecting the **Enter a User Defined Token** format enables the Tokens button. Selecting an item from the Enter a Security Group drop down list dynamically updates the Security Group field.
Security Group	Provides a field for specifying the recipient. If the Enter a Security Group drop down list is**:** • **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.

10. Click **OK** to add the User Data field to the entity.

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. Click the **Configuration** screen group and click the **User Data** screen.

   The User Data Workbench opens.

2. Search for the desired User Data Type from the **Query** tab and select it from the **Results** tab of the User Data Workbench.

3. Click **Open**.

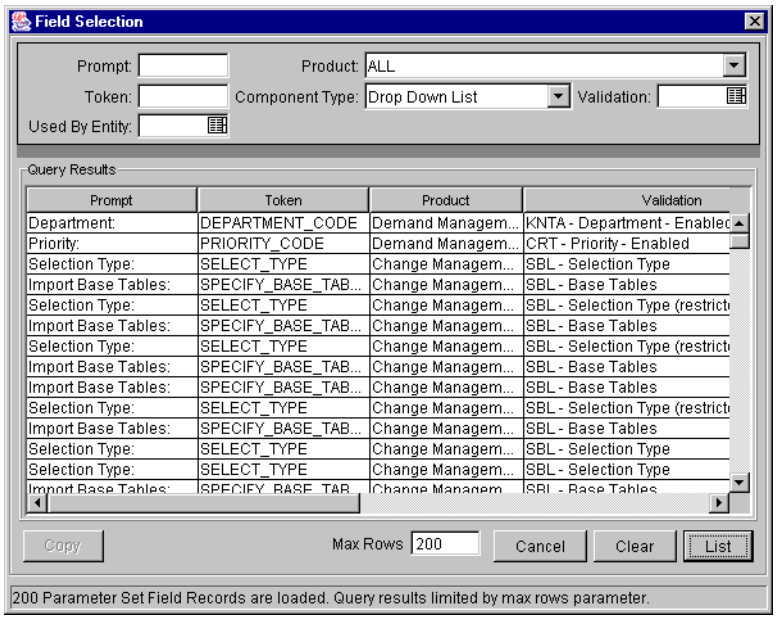
   The User Data window opens.

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

   The Field: New window opens.

5. Click **Copy From**.

   The Field Selection window opens.



6. Search for 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.

Due to the large number of fields in the Mercury IT Governance Center, limit the list of fields by one or more of the query criteria.

7. Select the desired field.
8. Click **Copy**.

   This closes the window and copies the definition of the selected field into the Field: New window.
9. Make any necessary modifications and click **OK**.

Editing User Data Fields

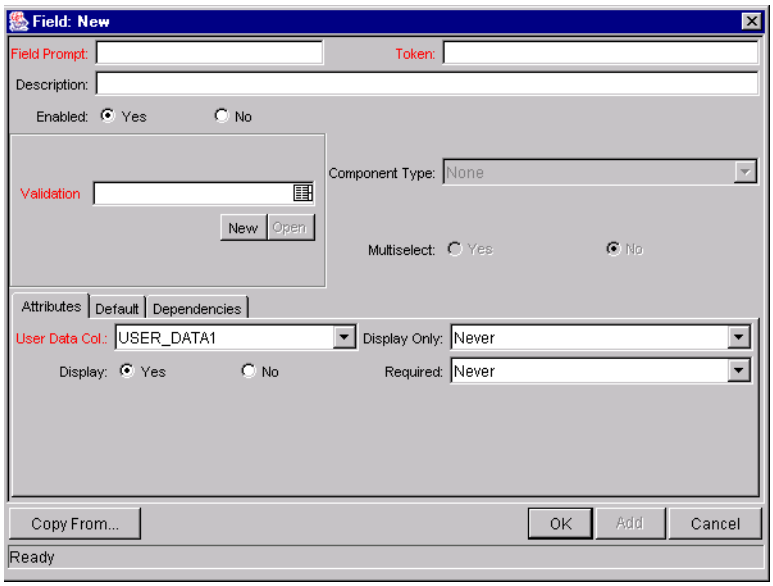
To edit an existing field:

1. Click the **Configuration** screen group and click the **User Data** screen.

   The User Data Workbench opens.
2. Search for the desired User Data Type from the **Query** tab and select it from the **Results** tab of the User Data Workbench.
3. Click **Open**.

   The User Data window opens.
4. Select the field to edit.
5. Either double-click on the Field in the **Fields** tab or select the field and click **Edit**.

   The Field window opens.



6. Make the desired changes in the header region, **Attributes** tab, **Default** tab, and **Dependencies** tab.

7. Click **Apply** to save the changes to the **Fields** tab without closing the Field window, or click **OK** to save the changes and close the Field window.

The field has been updated with the changes.

Configuring User Data Field Dependencies

Field behavior and properties can be linked to the value of other fields defined for that entity.

Example

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 0-6*.

- Become required when another field meets a logical condition, defined in *Table 0-6*.

Table 0-6. Field Dependency Logical Qualifiers

Logical qualifier	Definition
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. Click the **Configuration** screen group and click the **User Data** screen.

   The User Data Workbench opens.

2. Search for the desired User Data Type from the **Query** tab and select it from the **Results** tab of the User Data Workbench.

3. Click **Open**.

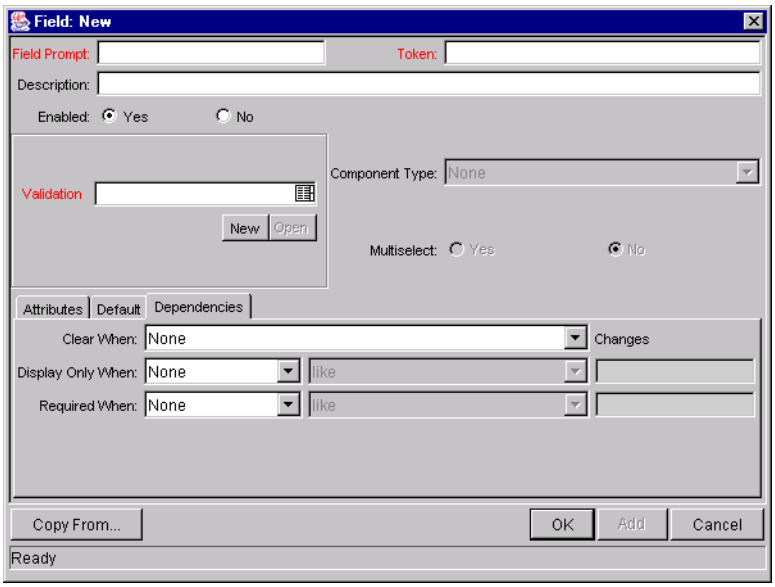
The User Data window opens.

4. Select the field you wish to edit.

5. Either double-click on the Field in the **Fields** tab or select the field and click **Edit**.

   The Field window opens.

6. Click the **Dependencies** tab.



7. 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.

8. Click **OK**.

Removing Fields

To remove a field permanently from a User Data Type:

1. Click the **Configuration** screen group and click the **User Data** screen.

   The User Data Workbench opens.

2. Search for the desired User Data Type from the **Query** tab and select it from the **Results** tab of the User Data Workbench.

3. Click **Open**.

   The User Data window opens.

4. In the **Fields** tab, select the field.

5. Click **Remove**.

6. To save the change to the database and close the window, click **OK**.

Modifying the User Data Layout

The layout of User Data fields can be changed in the **Layout** tab of the User Data window.

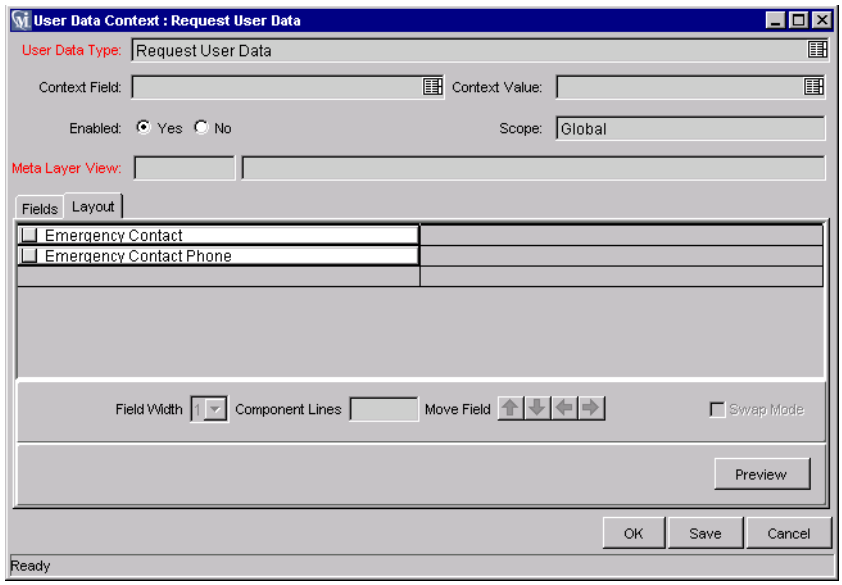


Figure 0-3 User Data Window - Layout Tab

The following sections discuss modifying User Data field layout in more detail:

- *Changing Column Width*
- *Moving a Field*
- *Swapping Positions of Two Fields*

Changing Column Width

To change the column width of a Field:

1. Click the **Configuration** screen group and click the **User Data** screen.

   The User Data Workbench opens.

2. Search for the desired User Data Type from the **Query** tab and select it from the **Results** tab of the User Data Workbench.

3. Click **Open**.

   The User Data window opens.

4. Click the **Layout** tab.

5. Select the Field.

6. The Field Width radio button, select either **1** or **2** in.

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.

Moving a Field

To move a field or a set of fields:

1. Click the **Configuration** screen group and click the **User Data** screen.

   The User Data Workbench opens.

2. Search for the desired User Data Type from the **Query** tab and select it from the **Results** tab of the User Data Workbench.

3. Click **Open**.

   The User Data window opens.

4. Click the **Layout** tab.

5. Select the field(s).

   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.

6. Use the directional arrow buttons 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. Those other fields must be moved out of the way first.

Swapping Positions of Two Fields

To swap the positions of two fields:

1. Click the **Configuration** screen group and click the **User Data** screen.

   The User Data Workbench opens.

2. Search for the desired User Data Type from the **Query** tab and select it from the **Results** tab of the User Data Workbench.

3. Click **Open**.

   The User Data window opens.

4. Click the **Layout** tab.

5. Select the first field.

6. Select the Swap Mode check box.

   This causes an **S** to appear in the check box area of the selected field.

7. 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.

To swap another set of fields, repeat the above procedure.

Previewing the Layout

To check what the layout will look like in actual use, click **Preview**. This opens a small window that shows the fields as they will appear in the window, shown in *Figure 0-4*.

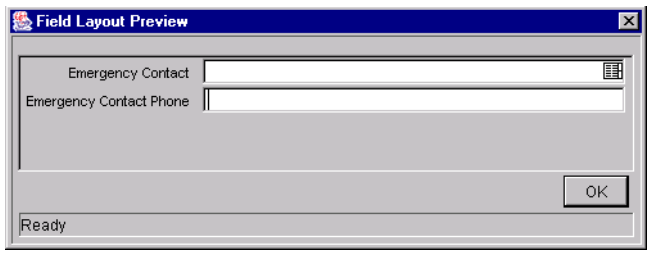


Figure 0-4 Preview Mode

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.

Creating and Editing Context Sensitive User Data

Certain User Data Types support an additional level of detail. The fields displayed in the entity's **User Data** tab can be dynamically linked to a field value in the information region of the respective entity's window. Context Sensitivity can be used with the following User Data Types:

- Package
- Request
- Validation Value

For example, if **Critical** was selected from the drop down list for the Priority field of a Request, then the Assigned To field could be automatically set to one of the top level support personnel, such as the manager of the technical support team.

The following sections provide detailed instructions for creating and editing Context Sensitive User Data:

- *Creating Context Sensitive User Data*
- *Editing Context Sensitive User Data*
- *Deleting Context Sensitive User Data*
- *Copying Context Sensitive User Data*
- *Example - Using Context Sensitive User Data for a Field in a Request Header Type*

Creating Context Sensitive User Data

Context Sensitive User Data can be defined for the Request, Package, and Validations (Validation value region) windows in the User Data Workbench.

To define Context Sensitive User Data:

1. Define a Context Field in the Global User Data scope.

   See *"Defining the Context Field"* on page 380.

2. Define a Context Value.

   See *"Defining a Context Value"* on page 382.

3. Define and configure the fields which appear under certain contexts.

   See *"Defining the Context Sensitive Fields"* on page 382.

When defining or editing Context Sensitive User Data fields for the same User Data Type, be sure to save any changes to the Global User Data fields before working on the Context User Data fields.

Defining the Context Field

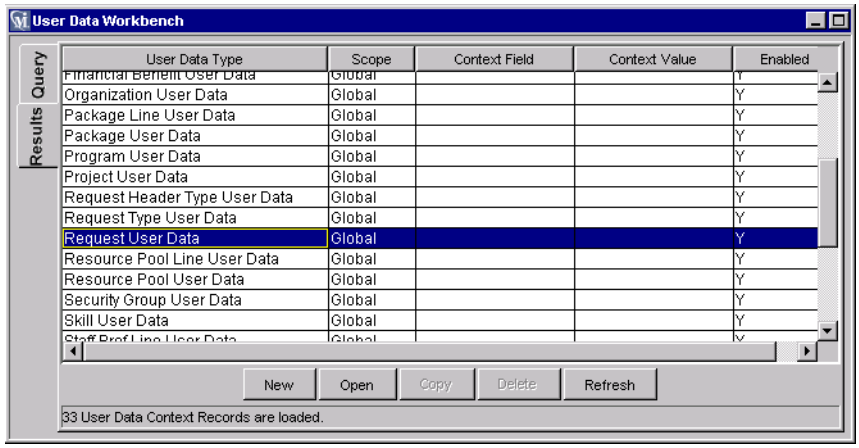
Only one field can be defined as the Context Field at any given time.

To specify the Context Field:

1. Click the **Configuration** screen group and click the **User Data** screen.

   The User Data Workbench opens.

2. Click **List** to display all of the existing User Data Types.



3. Select the desired User Data Type (Package, Request or Validation) with a **Global** scope.

4. Click **Open**.

   The User Data Context window opens.

5. Select the Context Field from the auto-complete list.

Note

The Context Field is disabled if any specific contexts for the User Data has been defined. In order to change the Context Field, all specific contexts for the User Data Type must first be deleted. For more information on editing Context Sensitive User Data, see *"Editing Context Sensitive Fields"* on page 384.

6. Verify that the global context is enabled (Enabled=**Yes**).

7. To save and close the window, click **OK**.

The Context Field has now been specified.

Note

The Context Field for the Validations Value User Data Type is always **Validation Name**.

Defining a Context Value

Context Values are the predefined possible values for the selected Context Field. Different User Data fields can be defined to be displayed for each of the possible Context Values.

To define a Context Value:

1. Click the **Configuration** screen group and click the **User Data** screen.

   The User Data Workbench opens.

2. Click **New** in the **Results** tab or **New User Data Context** in the **Query** tab.

   The User Data Context window opens.

3. Select a User Data Type from the auto-complete list.

   The Context Field is displayed as read-only.

Only User Data Types with a defined Context Field appear in the list. To define a Context Field for Request, Package, or Validation Values, see *"Defining the Context Field"* on page 380.

4. Select a Context Value from the drop down list or auto-complete list.

Defining the Context Sensitive Fields

User Data fields to be used with the specified Context Value are defined and configured just as in other areas of the product. For details on defining the field content and layout, see one of the following sections:

- *"Creating User Data Fields"* on page 365
- *"Editing User Data Fields"* on page 371
- *"Removing Fields"* on page 375

To define different fields based on a different Context Value, see *"Defining a Context Value"* on page 382.

Editing Context Sensitive User Data

Context Sensitive User Data can be edited for the Request, Package, Environment, and Validations (Validation value region) windows in the User Data Workbench. For details on editing Context Sensitive User Data, see one of the following sections:

- *Changing the Context Field*
- *Changing the Context Value*
- *Editing Context Sensitive Fields*

Changing the Context Field

In order to change the Context Field, all specific contexts for the User Data Type must first be deleted.

To change the Context Field for a particular User Data Type:

1. Click the **Configuration** screen group and click the **User Data** screen.

   The User Data Workbench opens.

2. To display all of the existing User Data Types, click **List**.
3. Locate the desired User Data Type.
4. Select the rows where the desired User Data Type's Scope=**Context**.
5. Click **Delete.**
6. Select the desired User Data Type.

   It will have a **Global** Scope.

7. Click **Open**.

   The User Data Context window opens.

8. From the auto-complete list, select the Context Field.
9. Verify that the Global Context is enabled (Enabled=**Yes**).
10. To save and close the window, click **OK**.

The User Data Type's Context Field has now been changed.

The Context Field for the Validations Value User Data Type is always **Validation Name** and cannot be changed.

Changing the Context Value

To change an existing User Data Type's Context Value**:**

1. Click the **Configuration** screen group and click the **User Data** screen.

   The User Data Workbench opens.

2. Click **List** to display all of the existing User Data Types.

3. Select the desired User Data Type that has the Context Value to be changed.

4. Click **Open**.

   The User Data Context window opens.

5. From the drop down list or auto-complete list, select a new Context Value.

6. Click **OK** to save the changes and close the window.

The User Data's Context Value has been changed.

Editing Context Sensitive Fields

User Data fields to be used with the specified Context Value are edited just as in other areas of the product. For details on editing the field content and layout, see one of the following sections:

- *"Creating User Data Fields"* on page 365
- *"Editing User Data Fields"* on page 371
- *"Removing Fields"* on page 375

Deleting Context Sensitive User Data

To delete a Context Sensitive User Data Type:

1. Click the **Configuration** screen group and click the **User Data** screen.

The User Data Workbench opens.

2. Click **List** to display all of the existing User Data Types.
3. Select the User Data Type to be deleted.
4. Click **Delete**.

   A Question dialog opens with the message "Delete 1 User Data Context[s]?"

5. Click **Yes** to confirm deletion.

Copying Context Sensitive User Data

To copy a Context Sensitive User Data Type:

1. Click the **Configuration** screen group and click the **User Data** screen.

   The User Data Workbench opens.

2. Click **List** to display all of the existing User Data Types.
3. Select the User Data Type with Scope=**Context** that is to be copied.
4. Click **Copy**.

   The Copy User Data window opens.

5. From the New Context Value list, select the new value.
6. Click **OK**.

   A Question dialog opens.

7. Click **Yes** to edit the context sensitive fields or **No** to accept.

Example - Using Context Sensitive User Data for a Field in a Request Header Type

Different values can appear in the Request's Applications field depending on which Request Header Type is used. For the Applications field in an ERP Request, the following distinct set of fields are available:

- Accounts Receivable
- Accounts Payable
- General Ledger
- Inventory

For an eCommerce Request, the following fields are available:

- Registration
- User Preferences
- Order Entry
- Order Tracking

Changing the Validation of a field in a Request Header Type can affect how information is returned from queries and reports. Use a context sensitive User Data approach when setting up such a system.

The following procedure provides an example for setting up the Applications Validation for the ERP Request introduced above:

Setting Up the Context Sensitive User Data

First, the Context Sensitive User Data must be configured.

1. Click the **Configuration** screen group and click the **User Data** screen.

   The User Data Workbench opens.

2. Click **New User Data Context**.

   The User Data Context window opens.

3. From the User Data Type auto-complete list, select **Validation Value User Data**.

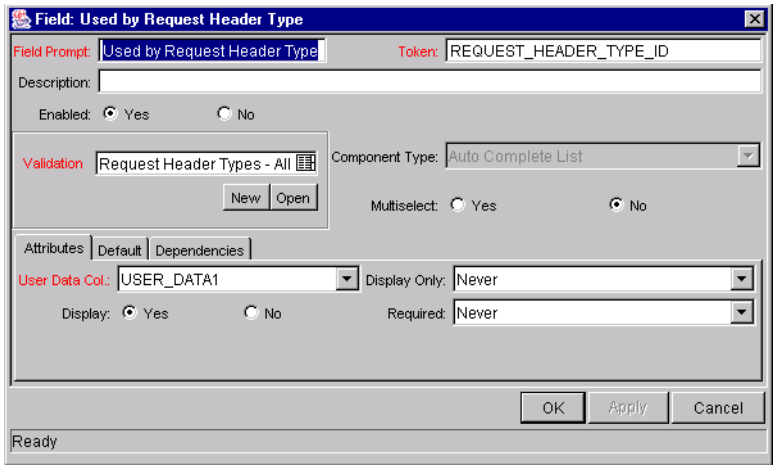
4. From the Context Value auto-complete list, select **KNTA - Application - Enabled**.

5. Click **New**.

   The Field: New window opens.

6. Create a new field with the following specifications:

- Field Prompt = **Used by Request Header Type**
- Token = **REQUEST_HEADER_TYPE_ID**
- Validation = **CRT - Request Header Types - All**



7. Click **OK**.

Example: Configuring the Validations

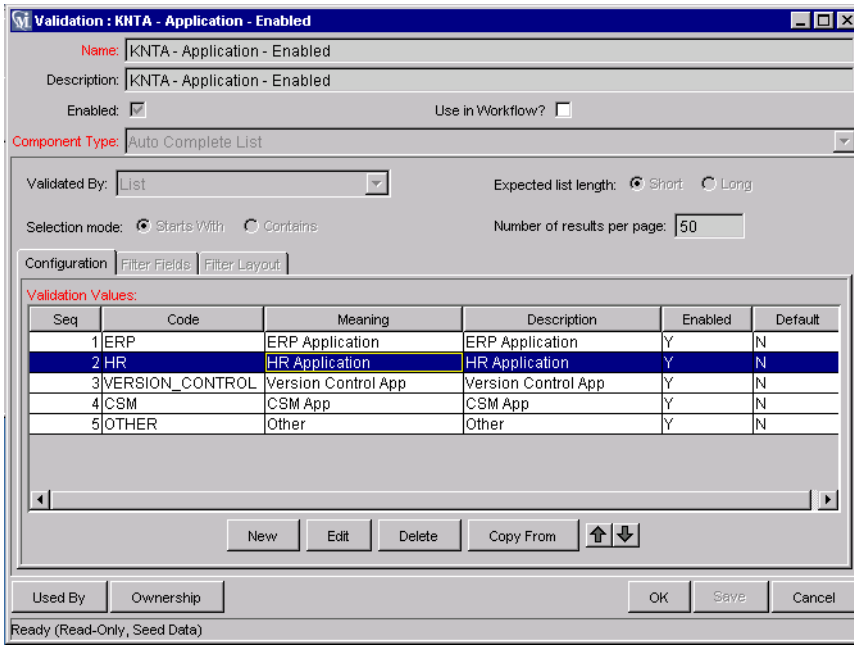
The Validation can now be configured:

1. Click the **Configuration** screen group and click the **Validations** screen.

   The Validation Workbench opens.

2. Search for and open the global KNTA - Application - Enabled Validation.

   The Validation window opens.



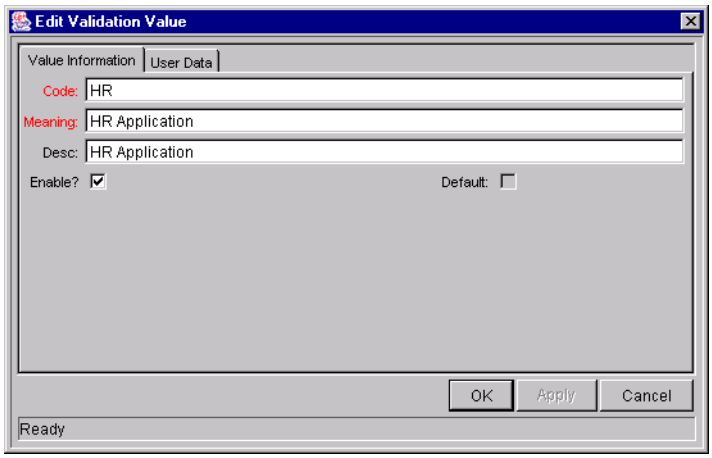
3. Associate each Validation Value with the appropriate Request Header Type shown in *Table 0-7*.

   To associate a Validation Value with a Request Header Type:

   a. Select a Validation Value from the Validation Values list.

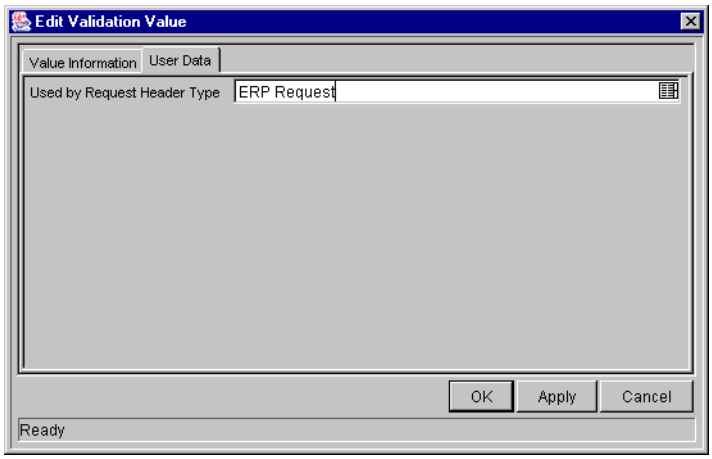
   b. Click **Edit**.

      The Edit Validation Value window opens.



c. Click the **User Data** tab.

d. Select the Request Header Type (**ERP Request** in this example) from the Used by Request Header Type auto-complete list.



e. Click **OK**.

f. Repeat these steps for each row in *Table 0-7*.

Table 0-7. Validation values associated with Request Header Types

Validation Value	Used by Request Header Type
Accounts Receivable	ERP Request
Accounts Payable	ERP Request
General Ledger	ERP Request
Inventory	ERP Request
Registration	eCommerce Request
User Preferences	eCommerce Request
Order Entry	eCommerce Request
Order Tracking	eCommerce Request

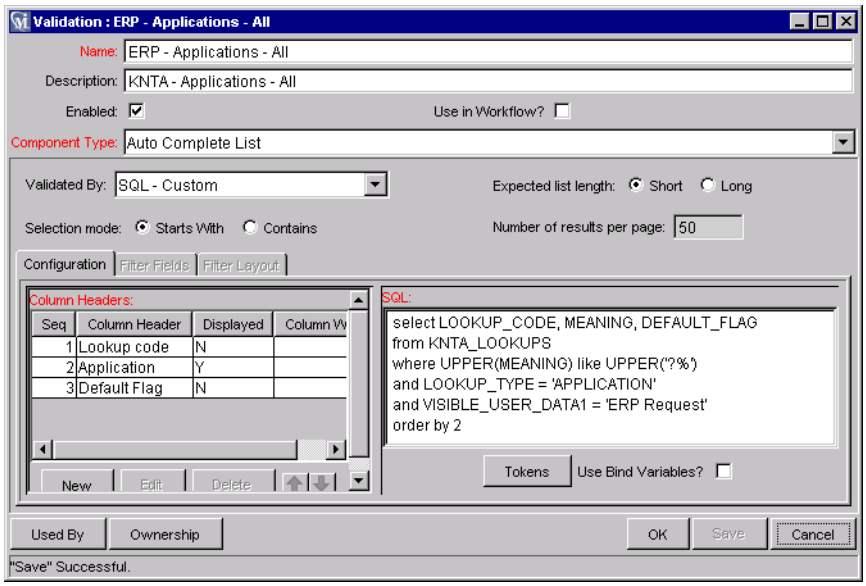
Example: Modifying the SQL

It is possible to now create variants of the standard Application Validation which vary depending on which Request Header Type is being used.

1. From the Validations Workbench, copy the KNTA - Application - All Validation.

2. Rename the Validation to ERP Applications - All.

3. Edit the Validation's SQL as shown below.

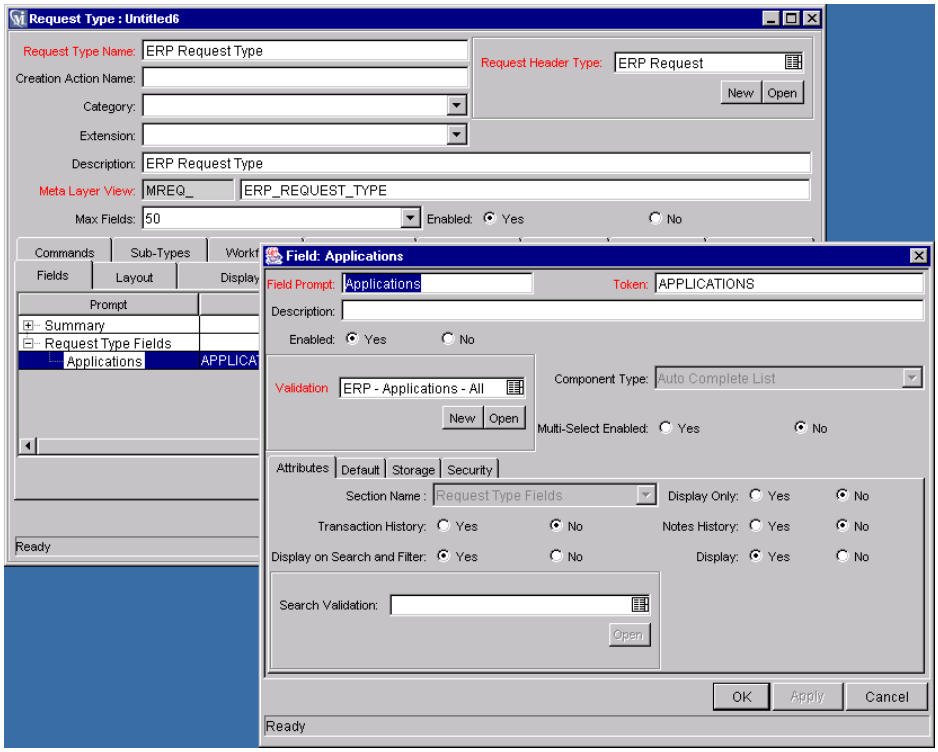


The specific variant for the ERP Request Header Type would be (assuming that the context-specific user data field was captured in the USER_DATA1 column):

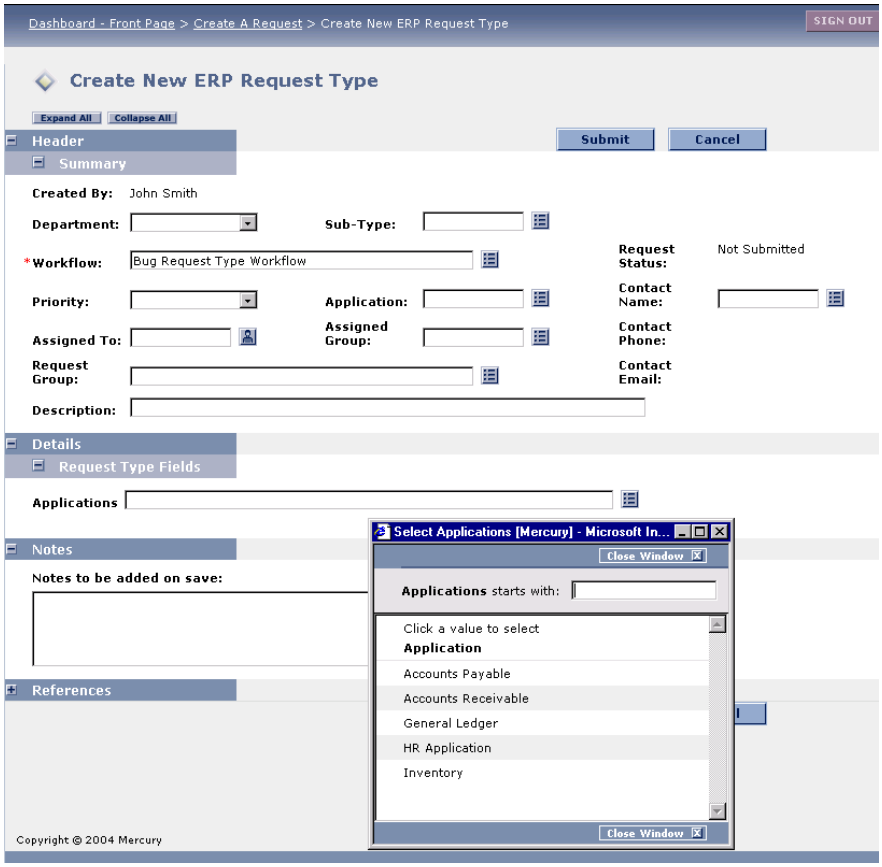
```
select LOOKUP_CODE, MEANING, DEFAULT_FLAG
from KNTA_LOOKUPS
where UPPER(MEANING) like UPPER('?%')
and LOOKUP_TYPE = 'APPLICATION'
and VISIBLE_USER_DATA1 = 'ERP Request'
order by 2
```

Example: Resulting Behavior

It is now possible to create a context-sensitive Applications field for any Request Type. In this example, simply create a new field with the Validation ERP - Applications - All.



When a user creates a Request with this Request Type (which references the ERP Request Header Type), the following Validations are associated with the **Applications** field.



Project/Task User Data Roll-Up

Values from Task User Data fields can be configured to “roll up” (combine/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/Task User Data Roll-Up can be used to capture various important aspects of a Project.

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.

Creating Project/Task User Data Roll-Up

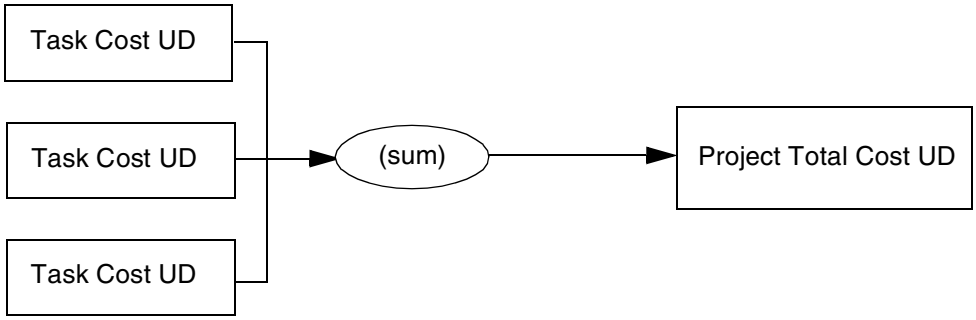
User Data must be configured for Projects and Tasks before specifying Roll-Up Methods.

1. Create and configure Project and Task User Data fields.
2. Link Project and Task User Data fields with Roll-Up Method.

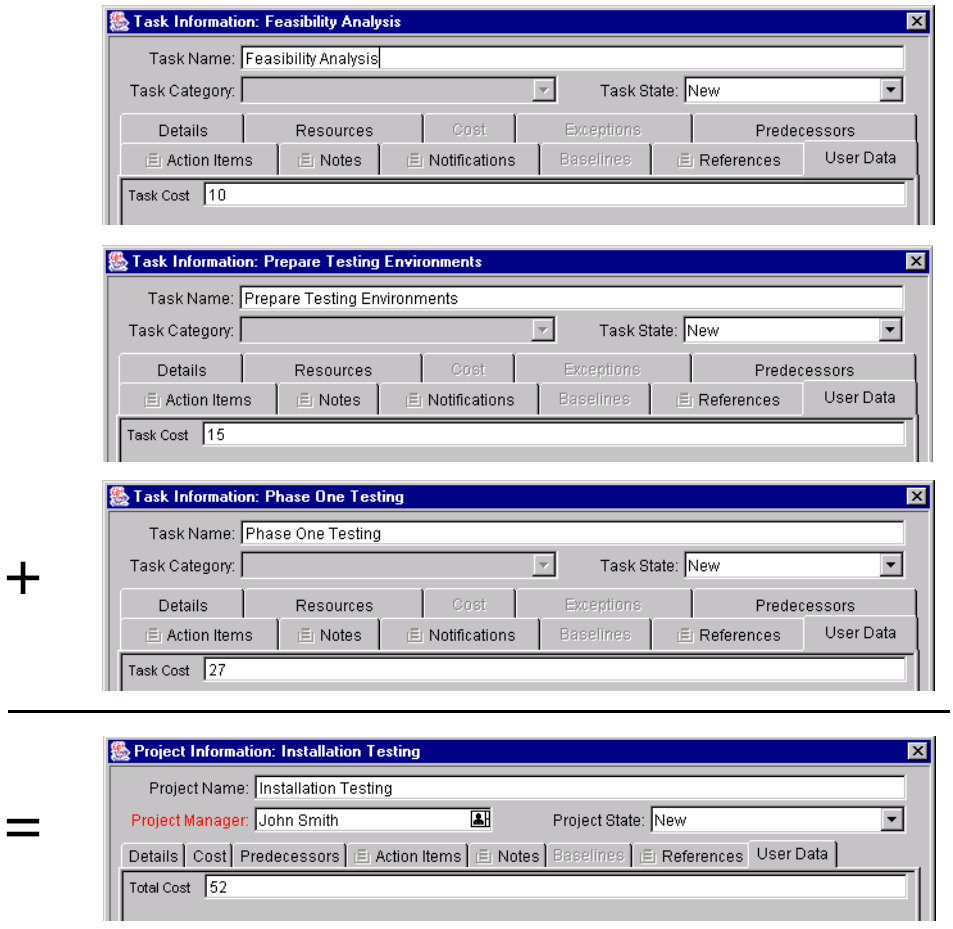
For more detailed information on configuring User Data, see *"Creating and Editing User Data"* on page 365.

Example: Using Project/Task User Data Roll-Up

Company X would like to capture total cost for its Projects. 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 below:



Each Task has its own Cost User Data field. 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.

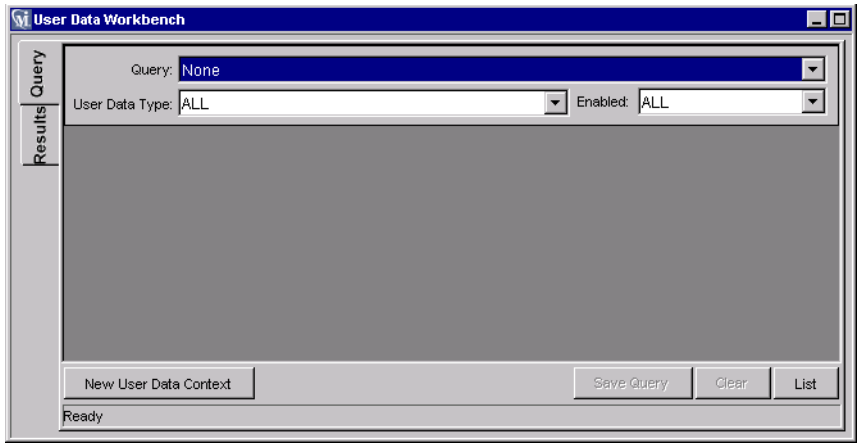


Once Project and Task User Data fields have been configured and saved, the Roll-Up relationship can be specified.

To specify the Roll-Up Method:

1. Click the **Configuration** screen group and click the **User Data** screen.

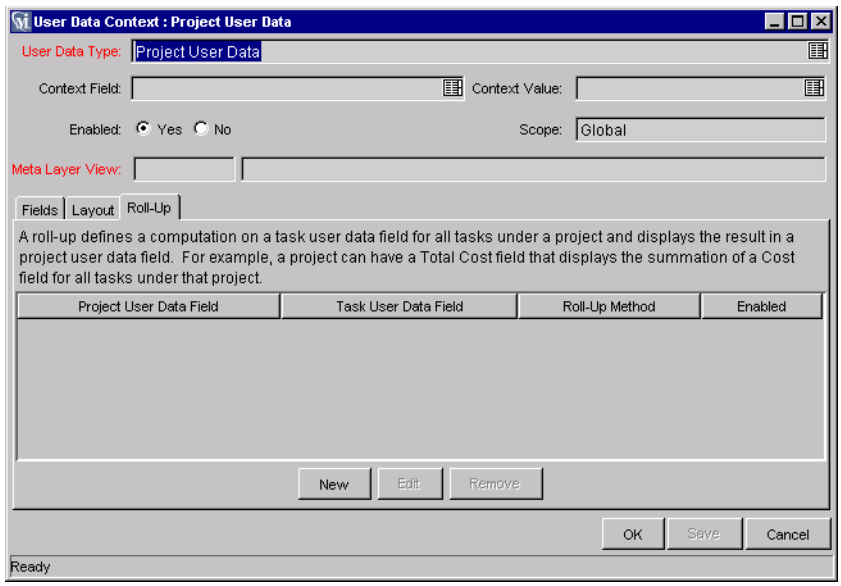
   The User Data Workbench opens.



2. From the User Data Type drop down list, select **Project User Data.**
3. Click **List**.

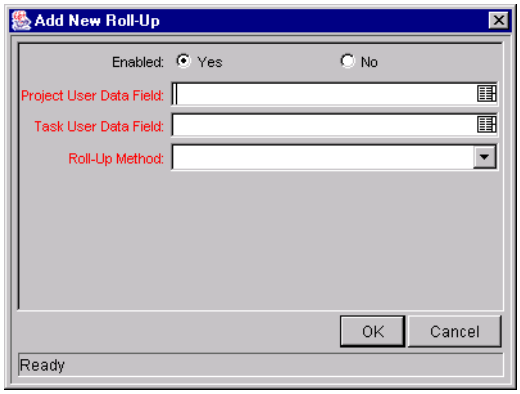
   The **Results** tab opens with the Project User Data Type loaded.

4. Select the Project User Data and click **Open**.
5. Click the **Roll-Up** tab.



6. Click **New**.

   The Add New Roll-Up window opens.



7. Select the Project User Data Field that will contain rolled-up Task User Data values.

8. Select the Task User Data Field whose values will roll up into the chosen Project User Data Field.

9. Select the Roll-Up Method from the drop down list.

   The drop down list will only display valid options for the data types of the Project and Task User Data fields.

10. To enable the Roll-Up, select **Yes**.

11. Click **OK**.

    The Roll-Up relationship is added to the **Roll-Up** tab.

12. Click **Save**.

Note

Only two User Data fields of the same type can be selected for Roll-Up (for example, a Numeric field cannot roll up into a Date field, nor vice versa).

While a Task User Data field can have multiple Roll-Up relationships associated with it, a Project User Data field cannot have more than one Roll-Up relationship defined.

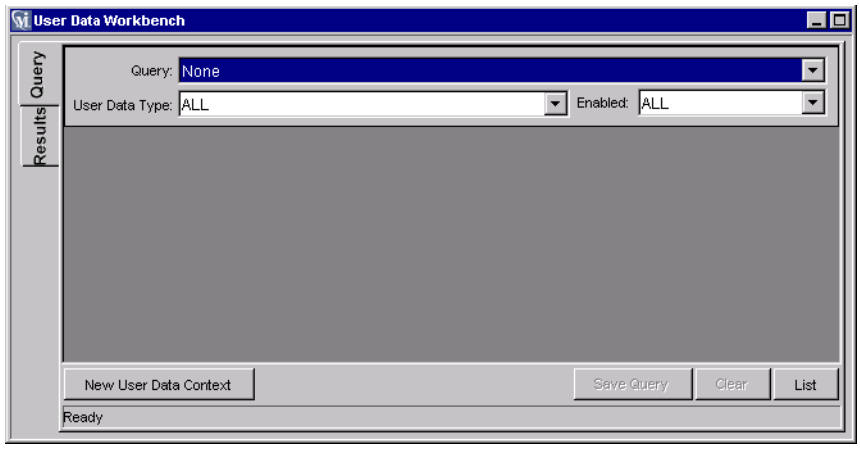
Editing Project/Task User Data Roll-Up

Project/Task User Data Roll-Up can be edited from the Workbench once it has been created.

To edit a Project/Task User Data Roll-Up relationship:

1. Click the **Configuration** screen group and click the **User Data** screen.

   The User Data Workbench opens.



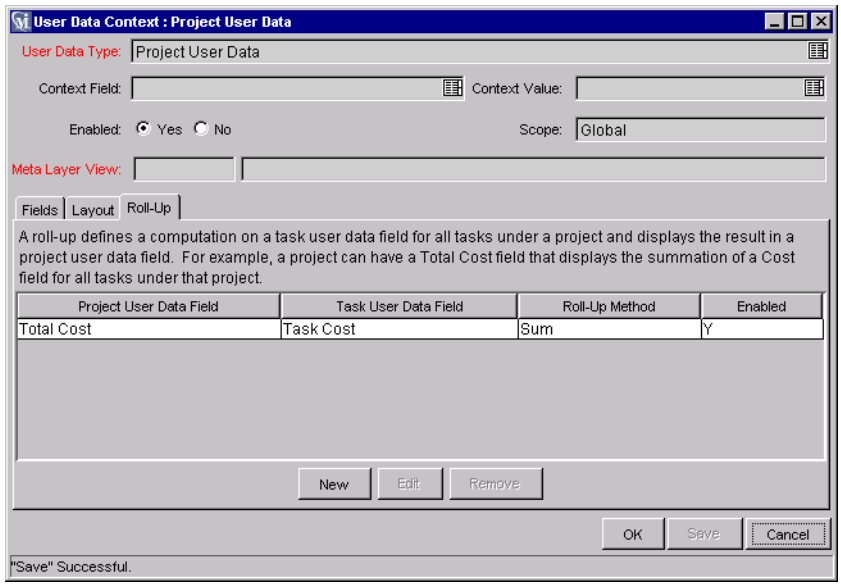
2. From the User Data Type drop down list, select **Project User Data**.

3. Click **List**.

The **Results** tab opens with the Project User Data Type loaded.

4. Select the Project User Data and click **Open**.

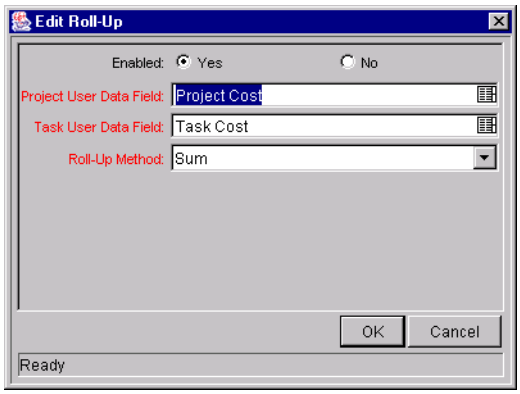
5. Click the **Roll-Up** tab.



6. Select the Roll-Up relationship you wish to edit.

7. Click **Edit**.

The Edit Roll-Up window opens.



8. Make any desired changes to the Project/Task User Data field or Roll-Up Method.

9. Click **OK**.

   The Roll-Up definition is updated in the **Roll-Up** tab.

10. Click **Save**.

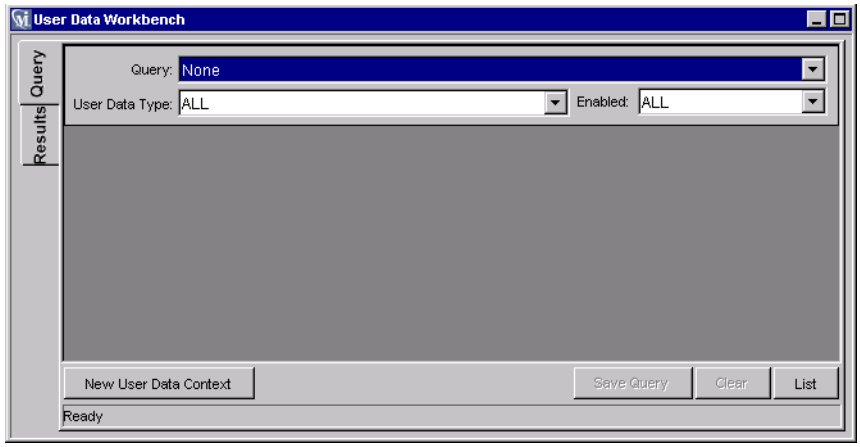
Deleting Project/Task User Data Roll-Up

Project/Task User Data Roll-Up can be deleted. This deletion only removes the Roll-Up relationship; it does not delete the referenced User Data fields.

To delete a Project/Task User Data Roll-Up relationship:

1. Click the **Configuration** screen group and click the **User Data** screen.

   The User Data Workbench opens.



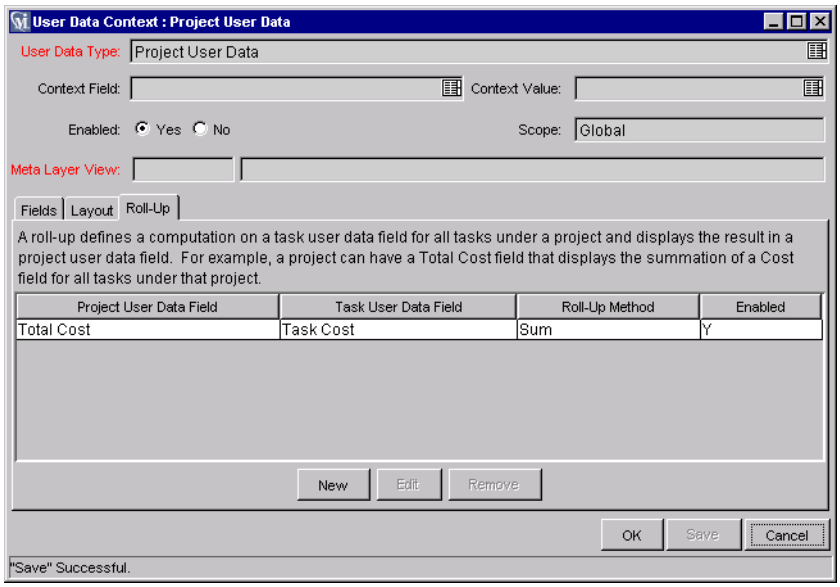
2. From the User Data Type drop down list, select **Project User Data.**

3. Click **List**.

   The **Results** tab opens with the Project User Data Type loaded.

4. Select the Project User Data and click **Open**.

5. Click the **Roll-Up** tab.



6. Select the Roll-Up relationship you wish to remove.
7. Click **Remove**.
8. Click **Save**.

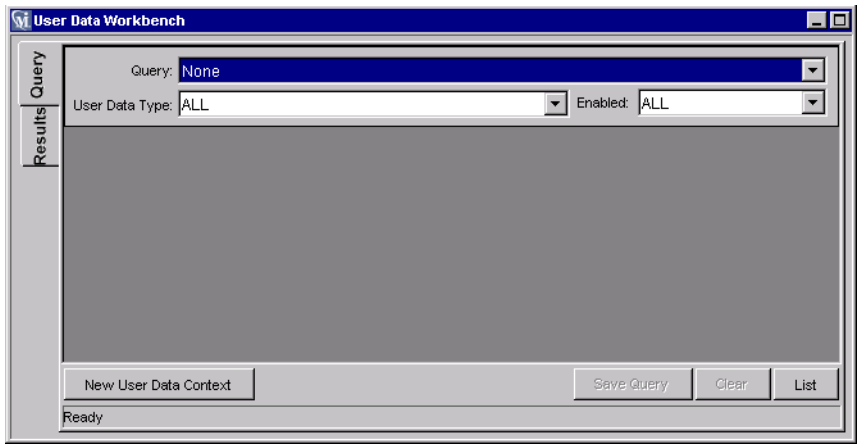
Example: Creating and Using Project/Task User Data Roll-Up

ACME wants to capture the total cost of any Project. This value will be calculated as the sum of all Task costs. They also want the calculated cost to be updated every time a Task cost is changed. They will accomplish this using Project/Task User Data Roll-Up.

To create Project/Task User Data Roll-Up that will calculate total Project cost:

1. Click the **Configuration** screen group and click the **User Data** screen.

   The User Data Workbench opens.



2. Create the Project User Data field.

   a. Select **Project User Data** from the User Data Type drop down list.

   b. Click **List**.

      The **Results** tab opens with the Project User Data Type loaded.

   c. Open the Project User Data Type.

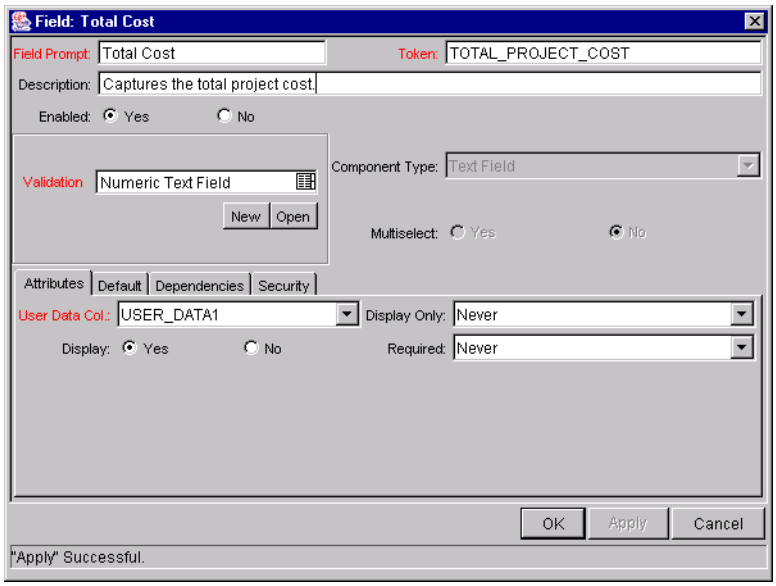
   d. Click **New** in the **Fields** tab.

      The Field: New window opens.

   e. Fill in the following information:

Field	Value
Field Prompt	Total Cost
Token	(any useful token)
Description	(any useful description)
Validation	Numeric Text Field
Enabled	Yes
User Data Col	(any available User Data column)
Display Only	Never
Display	Yes
Required	Never

Field	Value
Workbench Only	No



f. In the Field: New window, click **OK**.

g. In the Project User Data window, click **OK** to save the new field.

3. Create the Task User Data field.

   a. In the User Data Workbench **Query** tab, select **Task User Data** from the User Data Type drop down list.

   b. Click **List**.

      The **Results** tab opens with the Task User Data Type loaded.

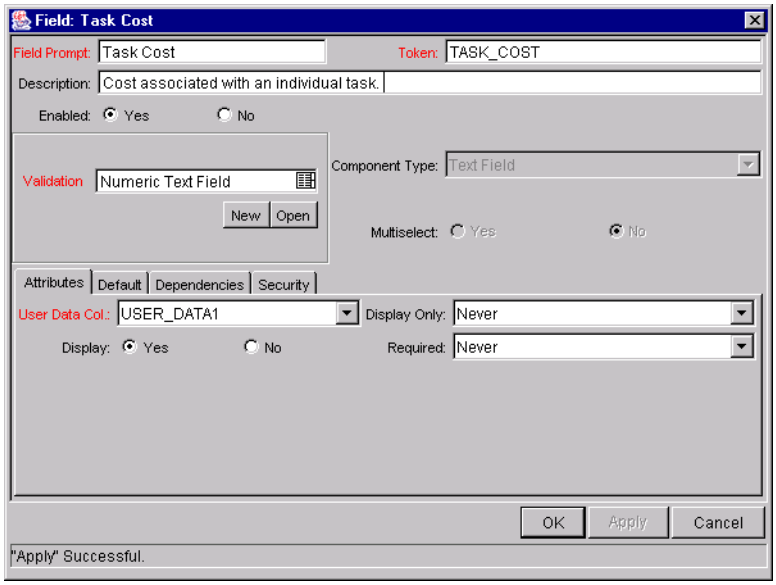
   c. Open the Task User Data Type.

   d. Click **New** in the **Fields** tab.

      The Field: New window opens.

   e. Fill in the following information:

Field	Value
Field Prompt	Task Cost
Token	(any useful token)
Description	(any useful description)
Validation	Numeric Text Field
Enabled	Yes
User Data Col	(any available User Data column)
Display Only	Never
Display	Yes
Required	Never
Workbench Only	No



f. In the Field: New window, click **OK**.

g. In the Task User Data window, click **OK** to save the new field.

4. Create the Roll-Up relationship between the Task and Project User Data fields.

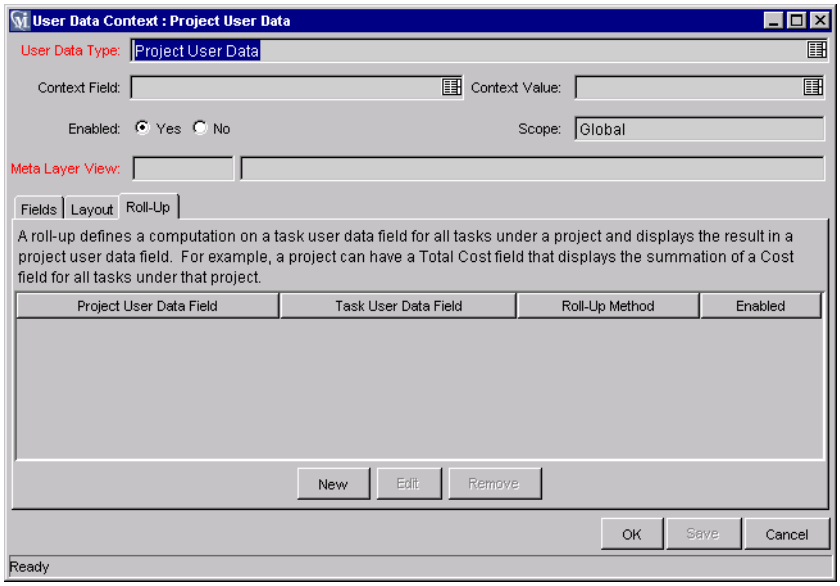
a. In the User Data Workbench **Query** tab, select **Project User Data** from the User Data Type drop down list.

b. Click **List**.

The **Results** tab opens with the Project User Data Type loaded.

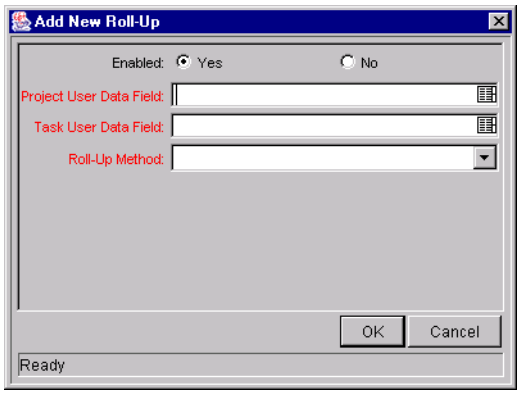
c. Open the Project User Data Type.

d. Click the **Roll-Up** tab.



e. Click **New**.

The Add New Roll-Up window opens.



f. In the Project User Data Field, select **Total Cost**.

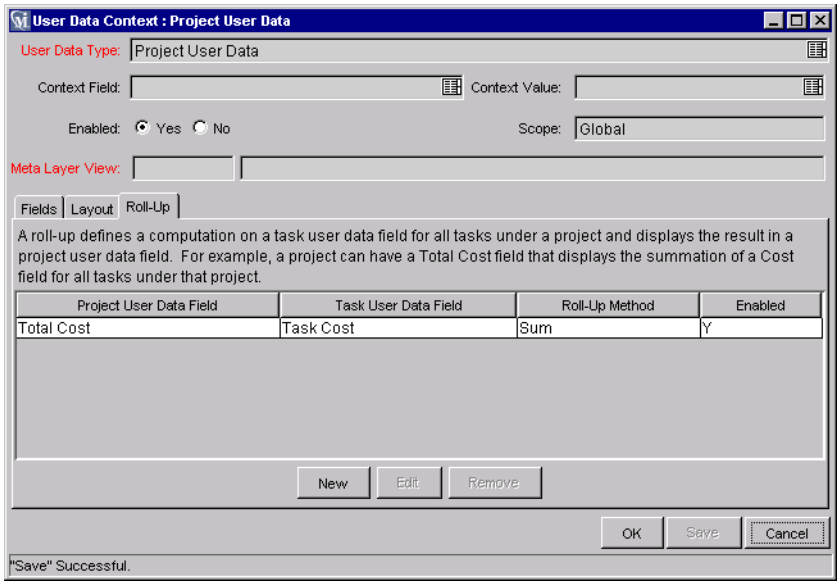
g. For the Task User Data Field, select **Cost**.

h. From the Roll-Up Method drop down list, select **Sum**.

i. Click **OK**.

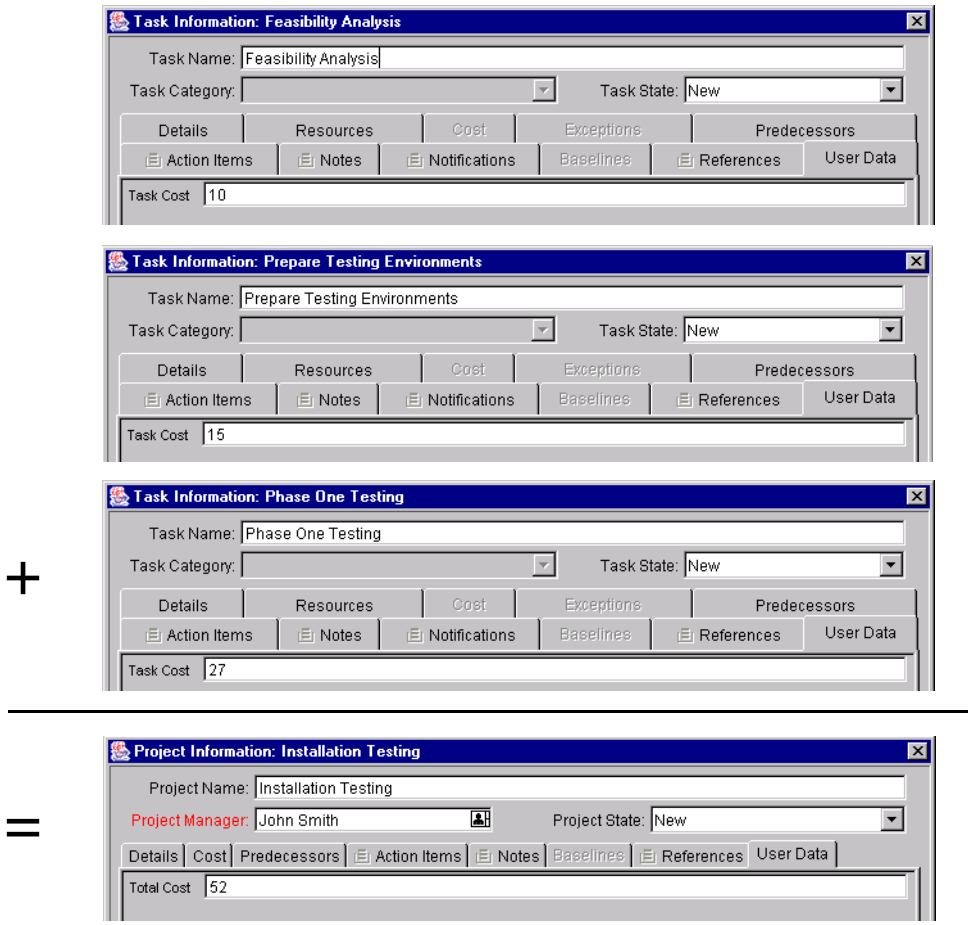
The Roll-Up relationship is added to the **Roll-Up** tab.

5. Click **Save**.



The rolled-up fields can now be accessed from the **User Data** tab of the respective Project or Task.

Each Task has its own Cost User Data field. 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.



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, proceeded by the entity abbreviation and the UD qualifier.

It is possible to define a custom field for the Users entity to store the Department each user is in. This custom field would be defined using the user's User Data and would generate a field with a token of 'DEPARTMENT.' Then, in an Object Type command, a Workflow Step, or in a Report, refer to this new field as the Token [USR.UD.DEPARTMENT]. The Mercury IT Governance Center would then look into the custom field for the value of this Token.

Migrating User Data

Product configuration data such as Workflows, Validations, and Request Types can be migrated between product instances (installations). User Data values and configurations can also be migrated between instances. The following sections contain more detailed information:

- *Migrating User Data Values*
- *Migrating User Data Contexts*

Migrating User Data Values

For any configuration entity with User Data fields (such as Request Type, Object Type, or Workflow) the data in the User Data fields is migrated along with the entity.

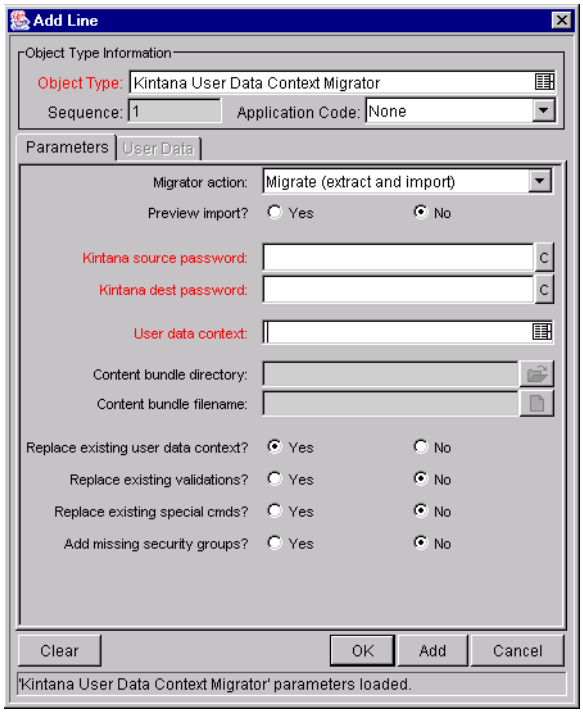
- If two instances have identical User Data configurations, then the User Data will be migrated correctly.
- If two instances do not have identical User Data configurations, then the User Data will be mapped into the data model according to the storage configuration in the source instance. For this reason, 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.

Note

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 (example: 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.

Migrating User Data Contexts

User Data field contexts can also be migrated between product instances using the User Data Context Migrator Object Type. This Migrator Object Type can migrate Global as well as Context Sensitive User Data Contexts.



For more detailed information on the User Data Context Migrator, see *Migrators Guide and Reference*.

Appendix E
Configuration Worksheets

This appendix provides worksheets that can be printed out and used to capture data required for configuring a deployment. Worksheets are provided for the following entities:

- Workflows
- Workflow Steps
- Object Types and Commands
- Object Type Fields
- Security Groups

Table E-1. Workflow Skeleton

Step No.	Step Name	Description	Type (Execution, Decision, Condition, or Subworkflow	Transition Valu
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				

Table E-2. Workflow Step [Execution] -- Step Number ____.

	Value
Step Name	
Goal / Result of Step	
Validation*	
Execution Type**	
Processing Type (**Immediate** or **Manual** execution?)	
Timeout (Days)	
Source Environment (Group)	
Dest Environment (Group)	
Security (who can act on step): • Security Group • User Name • Standard Token • User Defined Token	
Include Notification (Yes/No)	
Notification Event	
Notification Recipient: • Username • Email Address • Security Group • Standard Token • User Defined Token	
Notification Message	

Validation Information*	Value
Existing Validation?	
New Validation?	
Validation Type: *(text field, auto-complete, drop down list, etc.)*	
Validation Definition (list of values or SQL)	

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	

Table E-3. Workflow Step [Decision] -- Step Number ____.

	Value
Step Name	
Goal / Result of Step	
Validation*	
Decisions Required (Vote on Step's outcome?)	• One • At Least One • All
Timeout (Days)	
Security (who can act on step): • Security Group • User Name • Standard Token • User Defined Token	
Include Notification (Yes/No)	
Notification Event	
Notification Recipient: • Username • Email Address • Security Group • Standard Token • User Defined Token	
Notification Message	

Validation Information*	Value
Existing Validation?	
New Validation?	
Validation Type: *(text field, auto-complete, drop down list, etc.)*	
Validation Definition (list of values or SQL)	

Table E-4. Workflow Step [Sub-Workflow] -- 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): • Security Group • User Name • Standard Token • User Defined Token	
Include Notification (Yes/No)	
Notification Event	
Notification Recipient: • Username • Email Address • Security Group • Standard Token • User Defined Token	
Notification Message	

Validation Information*	Value
Existing Validation?	
New Validation?	
Validation Type: *(text field, auto-complete, drop down list, etc.)*	
Validation Definition (list of values or SQL)	

See *"Using Subworkflows"* on page 255 for notes on Validations for transitions into and out of Subworkflows.

Table E-5. Object Type Information.

	Value
Object Type Name	
Description	

#	Field Names	Description
1		

2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		

Table 0-8. Object Type Commands

Goal of Commands	

Table 0-8. Object Type Commands

Command Steps	
Conditions (When to execute)	

Table E-6. Object Type Field Information.

Field Name	
Validation*	
Field Behavior:	

Table E-6. Object Type Field Information.

Attributes (select one):	• Display • Editable • Display Only • Required
Default Value	
Dependencies:	
Clear field when	
Display only when	
Required when	

Table 0-9. Field Validation Information

Existing Validation?	
New Validation?	
Validation Type: *(text field, auto-complete, drop down list, etc.)*	
Validation Definition (list of values or SQL)	

Table E-7. Object Type Field Information.

Field Name	
Validation*	

Table E-7. Object Type Field Information.

Field Behavior:	
Attributes (select one):	• Display • Editable • Display Only • Required
Default Value	
Dependencies:	
Clear field when	
Display only when	
Required when	

Table 0-10. Field Validation Information

Existing Validation?	
New Validation?	
Validation Type: *(text field, auto-complete, drop down list, etc.)*	
Validation Definition (list of values or SQL)	

Participant and Security

Table E-8. Security Groups.

Security Group Name	Members	Act on Workflow Steps	

Index

A

B

C

D

E

F

H

P

R

S

T

U

V

W