

Mercury™ IT Governance Center

Migrators Guide and Reference

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 Test-Center, 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.

Table of Contents

Chapter 1	
Introduction	1
About This Document	2
Intended Audience	2
Document Conventions	2
Additional Resources	3
Related Documentation	4
Customer Support	4
Education Services	4
Chapter 2	
Key Concepts	7
Migrator	7
Migrators and XML	8
Using Workflows	9
Environments for Migrators	10
Migrators and Ownership	11
Supported Entities	12
Chapter 3	
Migrator Overview	13
Migrator Architecture	13
Migrator Processes - Best Practices	14
Running a Migration	14
Replacing an Existing Workflow	14
Deprecating a Workflow.....	16
Migrating a Request Type with Rules.....	16
Migrating a Request Type with an Associated Workflow	16
Migrating Nested Project Templates	17
Migrations and Entity Restrictions	17
Migrator Setup	18
Configuring Environments.....	18

Defining the Workflow.....	22
Using Migrators	25
Assumptions	25
Building the Migrator Package	25
Using the Execution Log	26
Chapter 4	
Using Migrators.....	27
Standard Migrator Fields and Behaviors.....	27
Migrator Action	28
Preview Import?	29
Target Entity.....	29
Content Bundle Fields	29
Import Behavior Fields	30
Source Password	30
Destination Password.....	31
Internationalization	31
Migrator Object Types	32
Validation Migrator.....	32
User Data Context Migrator	34
Special Command Migrator	35
Workflow Migrator	36
Report Type Migrator	40
Object Type Migrator	42
Request Type Migrator.....	44
Request Header Type Migrator	46
Project Template Migrator	48
Portlet Migrator	50

Chapter 1 Introduction

Mercury IT Governance Migrators are used to move Mercury ITG configuration data such as Validations, Workflows, and Request Types between Mercury ITG instances (installations).



Note

Only use Mercury ITG Migrators to move configuration data between Mercury ITG instances of the same version.

Migrators provide change control to configuration data through their use of Change Management Workflows and execution logs. Change control often means having separate non-production systems for development and testing of configuration data, then moving configuration changes to the production system through a structured process following a review.

Migrators are provided as Change Management Object Types. This makes it possible to use the production Change Management system to formalize the change control process into a Workflow, and use Packages to automate and audit the migration of configuration changes into production.

Use Migrators to document processes by exporting configuration information to text files that conform to the XML (eXtended Markup Language) specification, compressed into a ZIP file. The ZIP files 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 their production Mercury ITG configurations.

About This Document

This document provides details for configuring and using the Migrator Object Types to migrate or archive data from instances. Each chapter or appendix covers specific topics on the Migrator and includes the following information:

<i>Key Concepts</i>	Defines the common concepts and terms related to Migrators.
<i>Migrator Overview</i>	Provides an overview of Migrator architecture, setup, and use.
<i>Using Migrators</i>	Discusses in detail common Migrator behaviors and all Migrator Object Types.
<i>Using Migrators to Archive</i>	Provides instructions to archive configuration entities in a version-control system.

Intended Audience

The intended audience for this document include:

- System administrators responsible for maintaining and updating Mercury ITG instances
- Users responsible for configuring Object Types






Note

You must have a Change Manager Power license to access the screens and windows described in this document. You must also belong to a Security Group with the correct Access Grants in order to define and process Packages.

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:/Mercury/conf/ resin.conf
<i>Link</i>	Linked URLs, filenames, and cross references are shown as blue italicized text.	www.mercury.com
<i>Variable</i>	Variables are shown as italicized text.	<i>ITG_Home</i> /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 maintain Mercury ITG instances:

- [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.

<i>Using the Workbench</i>	This document explains how to navigate through the Mercury IT Governance Workbench interface.
<i>Installation Guide</i>	This document describes the requirements and procedures for installing and configuring the Mercury ITG product suite.
<i>Upgrade Guide</i>	This document contains information on topics related to upgrading a Mercury ITG instance, including: new features, upgrade impacts, and upgrade process.
<i>System Administration Guide</i>	This document provides information necessary to implement, configure, and maintain Mercury ITG Servers.
<i>Processing Packages (Change Management)</i>	This document explains how to process Packages using Change Management.
<i>Configuring a Deployment System (Change Management)</i>	This document provides instructions for configuring a deployment system using Mercury ITG. This includes requirements gathering, modeling your processes in a Workflow, defining commands used by Mercury ITG's execution engine, and rolling out this system to your users.

Customer Support

Customer support for the ITG Suite 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 common concepts and terms related to Mercury IT Governance Migrators:

- *Migrator*
- *Migrators and XML*
- *Using Workflows*
- *Environments for Migrators*
- *Migrators and Ownership*
- *Supported Entities*

Migrator

A Migrator is a Change Management Object Type. Each Migrator is designed to migrate a specific Mercury ITG entity, as well as all of its dependent objects, from one Mercury ITG instance to another.



Example

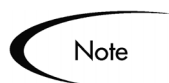
It is possible to migrate a Request Type from the Testing instance to the Production instance. When migrating the Request Type, the following information related to the Request Type is also migrated:

- Validations referenced by the Request Type fields
- Any Special Commands referenced by Request Type Commands or Validations.

Migrators and XML

The Migrators use the eXtensible Markup Language (XML) specification to archive configuration data for Mercury ITG entities. Much of the behavior of the Mercury ITG product is configurable (specific configurable entities include Workflows, Object Types, Request Types, Validations, etc.), and in any Mercury ITG instance, this configuration data is stored within the Mercury ITG server. A Migrator Object Type can extract this data into a collection of data files. Once extracted from a Mercury ITG instance by a Migrator Object Type, this collection is referred to as a “content bundle.”

A content bundle can be imported into another Mercury ITG instance, or archived separately. When archived, a content bundle takes the form of a ZIP file containing a collection of XML files that can later be stored or imported into a Mercury ITG instance.



Note

It is recommended to not unzip this file manually except for debugging purposes. Even in such cases, only do so when all possible consequences have been considered.

All Migrator Object Types contain fields that can be used for to specify an archival directory, or temp directories for the ZIP files, to be used for export/import on a one-time basis.

Migrator Content
Bundle Fields

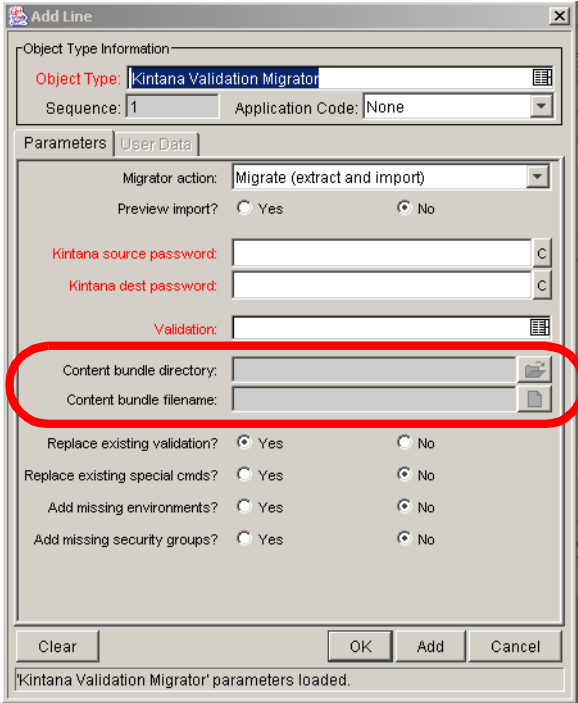


Figure 2-1 Migrator Object Type Content Bundle Fields

Using Workflows

Like all Change Management Object Types, Migrators are used as Package Lines. A user builds a Package with one or many Migrators and then submits the Package along a pre-defined Change Management Workflow. A Workflow might be built for every Migrator Object Type in a 1:1 ratio, or a single generic Workflow could be created for all Migrator Object Types to use. For more detailed information on setting up a Workflow for use with Migrators, see *"Migrator Setup"* on page 18.

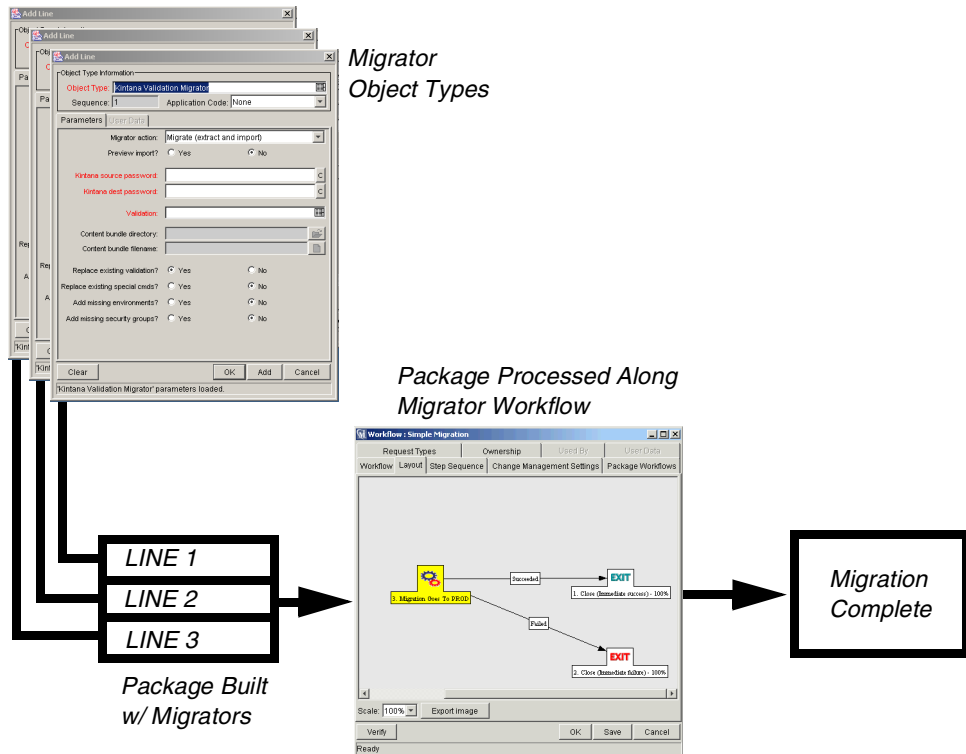


Figure 2-2 Migrator Process: High-Level Overview

Environments for Migrators

Before a migration takes place, Mercury ITG Environments should be defined representing the source and destination instances. For more detailed information on setting up Environments for Migrators, see *“Migrator Setup”* on page 18.

Figure 2-3 Sample Environment Setup for Source Instance

Migrators and Ownership

Different groups of Mercury ITG users have ownership and control over Mercury ITG entities that are specific for their respective groups. 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 Users Access Grant is needed for Users.

Application administrators can assign multiple Ownership Groups to the various entities. The Ownership Groups will have sole control over the entity, providing greater security. Ownership Groups are defined in the Security Groups window. Security Groups become Ownership Groups when used in the Ownership configuration.

Ownership applies to Mercury ITG entities during migrations in the following ways:

- If no Ownership security is configured for the entity, any user able to perform migrations can migrate it.
- If Ownership is configured for the entity and the user migrating is not in the Ownership Group, the migration will fail.
- If Ownership is configured for the entity and the user migrating is in the Ownership Group, the migration succeeds.
- If Ownership is configured for the entity and the user migrating is not in the Ownership Group but has the Ownership Override Access Grant, the migration succeeds.



These conditions apply to import, but not export.

Supported Entities

The following entities can be migrated using Mercury ITG Migrators:

- Validations
- User Data Contexts
- Special Commands
- Workflows
- Report Types
- Object Types
- Request Types
- Request Header Types
- Project Templates
- Portlets

Chapter
3**Migrator Overview**

This chapter provides an overview of Mercury IT Governance Migrator architecture, setup, and use. Examples illustrating particular aspects of each are also provided, where applicable.

The chapter discusses the following topics:

- *Migrator Architecture*
- *Migrator Processes - Best Practices*
- *Migrator Setup*
- *Using Migrators*

Migrator Architecture

A Migrator Object Type works by extracting configuration data for a particular entity (see “*Supported Entities*” on page 12 for a full list of entities that can be migrated) into a collection of data files called a content bundle that can be exported and imported into other Mercury ITG instances or a source control system for archival purposes.

When the Migrator is being used to extract configuration data for archiving, the content bundle takes the form of a collection of XML files, compressed into a ZIP file that can be stored in a version control system and imported into a Mercury ITG instance anytime in the future.

Migrator Processes - Best Practices

Migrating configuration entities from one instance to another can be a complex undertaking. When migrating certain entities such as Workflows or Request Types, there are some situations where guidelines may prove helpful. The following sections discuss some of the particulars of migrating certain Mercury ITG entities:

- [*Running a Migration*](#)
- [*Replacing an Existing Workflow*](#)
- [*Deprecating a Workflow*](#)
- [*Migrating a Request Type with Rules*](#)
- [*Migrating a Request Type with an Associated Workflow*](#)
- [*Migrating Nested Project Templates*](#)
- [*Migrations and Entity Restrictions*](#)

Running a Migration

Using a Migrator, Mercury ITG configuration information is moved with standard Change Management methods: processing a Package through a Workflow. The Package in a migration consists of Package Lines made from Migrator Object Types.

The instance that actually runs the migration should be the destination instance. In a typical case of migrating from a test instance to a production instance, the driving instance would be the production system as opposed to testing.

Replacing an Existing Workflow

When using the Workflow migrator to make changes to an existing process that is already in use (by Requests or Package Lines), there are some restrictions. These restrictions help to ensure that these existing Requests or Package Lines will not be damaged by the migration.

Specifically, the Workflow migration will not succeed unless the migrator logic can find a Workflow Step in the incoming process that corresponds to each step in the previous process. The following conditions are used to match Workflow Steps between instances:

- The Step Source (the particular decision, execution, or condition) of a Workflow Step is always used for matching Workflow Steps to each other. If the Step Source is not identical, then two Workflow Steps cannot be considered to match.
- When both the incoming and previous Workflows assign a unique name to each Workflow Step, these Workflow Step names are used in combination with the Step Source to assess matching.
- When a Workflow Step name is repeated within either process, the step sequence is used instead, in combination with the Step Source, to assess matching.

Note

The Workflow Migrator is not able to handle a single change in which both the names of existing Workflow Steps and the step sequence of existing Workflow Steps has changed.

To change both the names and step sequences of a Workflow:

1. Change step names, but do not change any step sequences. Migrate the changed Workflow.
2. Change step sequences, but do not change any step names. Migrate the changed Workflow a second time.

Due to this matching restriction, each open Request will still be at the same process step following the migration as it was prior to the migration. The migration may have changed the name of this step, but it will not have transitioned any Request Workflows.

It is important to note, however, that the migrator does not prevent the removal of outgoing transitions from Workflow Steps. It is therefore important to avoid “stranding” open Requests at a Workflow Step that will be deprecated. When deprecating a process step, you will want to remove incoming transitions, but still want to leave at least one outgoing transition from the step. This will allow open Requests to move forward. This is also true for Packages and Release Distributions.

The migration’s execution log will contain a table listing old and new Workflow Steps. Mercury ITG recommends using the Preview import mode first when replacing an existing Workflow, and inspecting this table of matched Workflow Steps before running such a Workflow migration in non-preview mode.

Deprecating a Workflow

When the changes to a Workflow are extensive, it is possible to deprecate the existing Workflow and bring the changes into the production instance as a new Workflow. One advantage of implementing the changes as a new Workflow is simplicity, since the new Workflow is not required to contain all of the steps of the old Workflow for backward compatibility.

To bring a new Workflow into the production instance:

1. Rename the existing Workflow and disable it in production.

Disabling the Workflow removes it from lists of Workflow options when creating new Requests. Requests that are already in process will continue to follow the old Workflow until they close, unless each is manually shifted to the new process and transitioned to an appropriate point in the process. Existing defaulting rules and other configurations will also continue to refer to the old Workflow regardless of the change of name.

2. Migrate the new version of the Workflow into the production instance, using the original name.

Since the production instance no longer contains a Workflow by this name, the migration will treat this as a new Workflow.

3. Following the migration, it is possible to update defaulting rules in Request Types to reference this new Workflow.

This can be done manually, or by migrating in versions of the Request Types that refer to the new Workflow by the original name.

Migrating a Request Type with Rules

Simple Default Rules, defined in the Request Type **Rules** tab, may reference Users, Workflows, or other objects. The Request Type Migrator will transfer these references, but will not create a missing User or Workflow. If the referenced User or Workflow does not exist in the destination instance, the reference will be discarded in transit, and a message to that effect will appear in the migration's execution log. Advanced Default Rules should be manually reconfirmed after migration.

Migrating a Request Type with an Associated Workflow

Circular references between Request Types and Workflows could make it necessary to migrate either a Request Type or Workflow twice.

1. A new Request Type referring to a new Workflow is migrated. Since the new Workflow does not exist in the destination instance, all references to that Workflow are not included in the new instance destination.
2. The new Workflow is migrated.
3. The new Request Type is migrated again. This time, since the Workflow it refers to exists, the references are included in the destination instance.

Migrating Nested Project Templates

A Project Template may also contain references to other Project Templates that have been used to create the current Template. The Project Template Migrator will transfer these references, but will not create a missing nested Project Template.

To ensure that these references are preserved, any Project Templates that have been nested inside other Project Templates should be migrated first. Otherwise, if the referenced nested Project Template does not exist in the destination instance, the reference will be discarded in transit. A message to that effect will appear in the migration's execution log.

Migrations and Entity Restrictions

A Report Type may refer to Security Groups through Entity Restrictions. The Report Type Migrator will transfer references to Security Groups, but will not create a Security Group. The behavior is described below:

- If the referenced Security Group does not exist in the destination instance, the reference will be discarded in transit. A message to that effect will appear in the migration's execution log.
- If the source instance contains Security Groups that do not exist in the destination instance at the time of migration, the Entity Restrictions for the migrated Report Type will not be accurate.

Report Types that contain Entity Restrictions should therefore be verified manually in the destination instance following migration. This ensures that Security Groups are configured as desired.

Migrator Setup

Mercury ITG Migrators are processed as Change Management Object Types. These Object Types include the information needed (fields) and the actions required (commands) to migrate information between instances, or information from an instance to a data file for archiving. Using a Migrator, Mercury ITG configuration information is moved with standard Change Management methods: processing a Package through a Workflow.

In order to use a Migrator Object Type, some configuration is necessary beforehand.

To use the Mercury ITG Migrators to migrate Mercury ITG configuration information between instances:

1. Configure the appropriate Environments to represent Mercury ITG instances involved in the migration.
2. Define a Change Management Workflow to model the desired migration process.

If desired, use entity restrictions to associate this Workflow with the Migrator Object Types.

3. Build a Package using this Workflow, using 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.

The following sections provide more detailed information on each step in the migration process:

- [Configuring Environments](#)
- [Defining the Workflow](#)

Configuring Environments

In order to configure an Environment representing a Mercury ITG installation, perform the following actions:

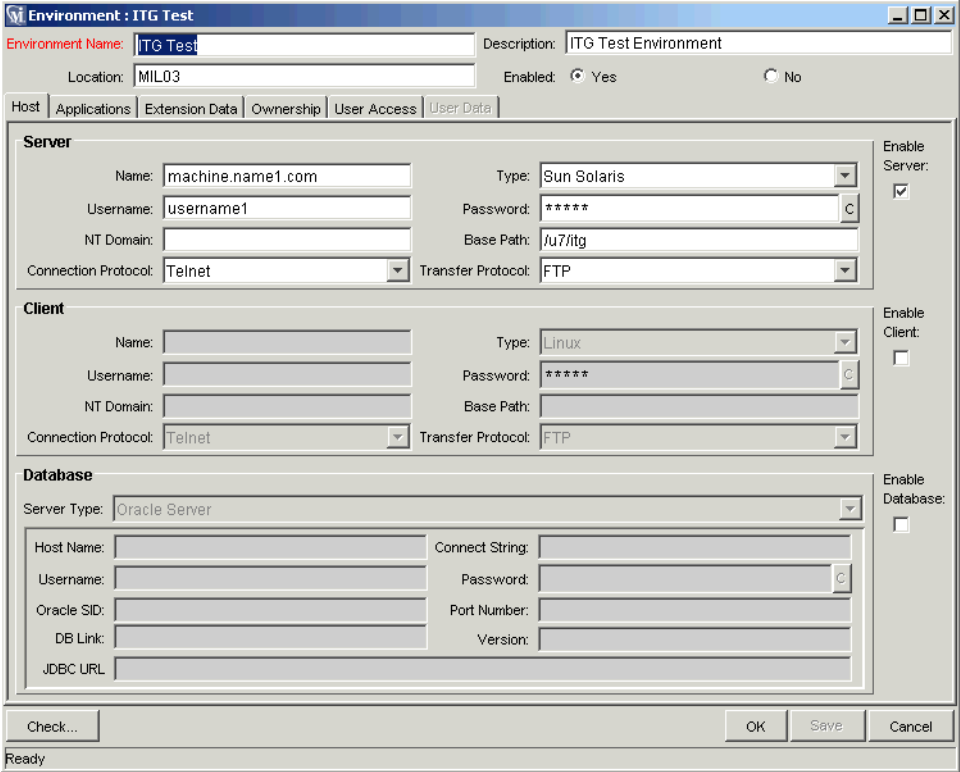
- [Configuring the Environment for the Source Instance](#)
- [Configuring the KINTANA_SERVER Environment for the Destination Instance](#)

- *Checking the server.conf File*

Configuring the Environment for the Source Instance

An Environment for the source instance should be specified.

1. Select the **Environments** screen group and click the **Environments** screen.
The Environment Workbench opens.
2. Click **New Environment** to create a new Environment.



3. Specify the following field values:

Field	Value
Environment Name	Name of the source Environment.
Server Name	Host name of the server.
Server Username	Username for the User on this server.
Server Password	Password for the User on this server.

Field	Value
NT Domain	Required only for Windows servers.
Server Base Path	The installation directory for Mercury ITG.

The Mercury ITG user must have write access to the Installation Path (SERVER_BASE_PATH) and subdirectories in order to migrate data.

- To save changes and continue to work with this Environment, click **Save**.

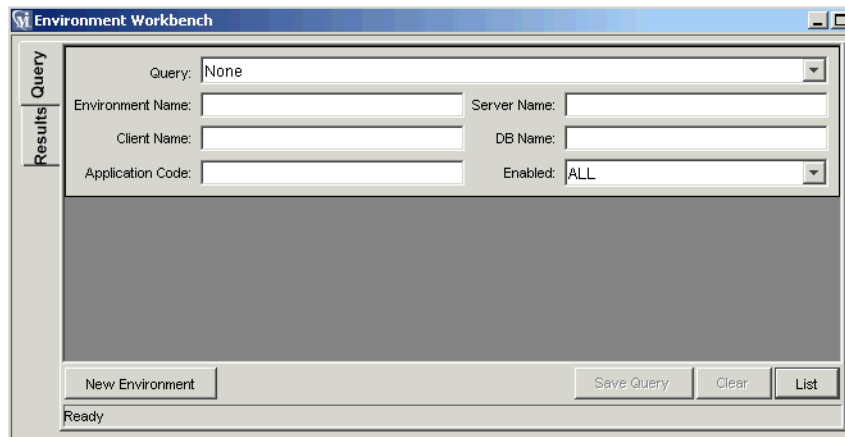
To save the Environment and close the window, click **OK**.

Configuring the KINTANA_SERVER Environment for the Destination Instance

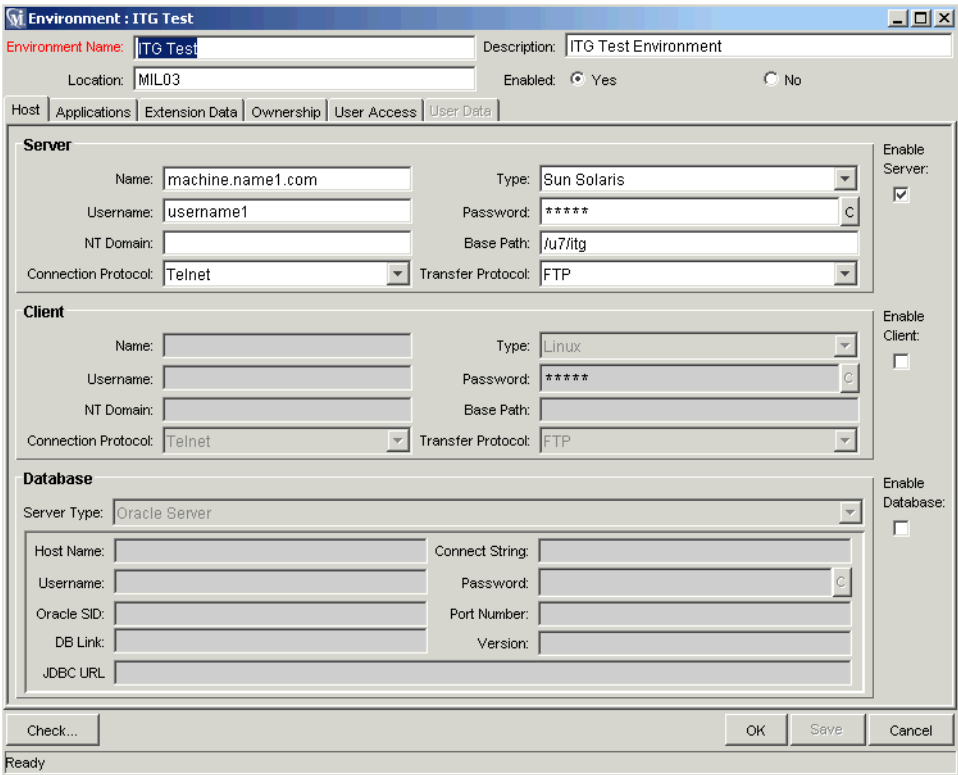
Mercury ITG installation automatically generates an Environment named KINTANA_SERVER, which represents the server hosting the Mercury ITG application server. This Environment, like all others, can be maintained using the Environment Workbench.

- Select the **Environments** screen group and click the **Environments** screen.

The Environment Workbench opens.



- Query for the KINTANA_SERVER Environment, or click **List** to display all of the system Environments.
- Open the KINTANA_SERVER Environment.



4. Specify the following field values:

Field	Value
Server Name	Host name of the server.
Server Username	Username for the KINTANA_SERVER.
Server Password	Password for the KINTANA_SERVER.
NT Domain	Required only for Windows servers.
Server Base Path	The installation directory for Mercury ITG.

Additionally, the Mercury ITG user must have write access to the Installation Path (SERVER_BASE_PATH) and subdirectories in order to migrate data.

5. To save changes and continue to work with this Environment, click **Save**.

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

Checking the server.conf File

The Mercury ITG Migrators depend upon a parameter in the Mercury ITG configuration file named `SERVER_ENV_NAME`. Set this parameter to the name of an Environment in the Mercury ITG system that describes the host server running that Mercury ITG instance. Since the destination instance should be the driving instance, `SERVER_ENV_NAME` should be set to `KINTANA_SERVER`.

On either platform, by default, the server Environment configuration entry should appear as below. Ensure this parameter is properly configured.

```
SERVER_ENV_NAME=KINTANA_SERVER
```

As a Mercury ITG installation automatically generates the Environment `KINTANA_SERVER`, it is also captured as the default `SERVER_ENV_NAME` parameter in the `server.conf` file.



Note

There should be no reason to change the default value of `KINTANA_SERVER`. In case it must be modified, both the `server.conf` parameter and the Mercury ITG Environment name must be synchronized.

Defining the Workflow

To use the Mercury ITG Migrator Object Types, it is necessary to define a Change Management Workflow. This Workflow can reflect the desired review and testing processes, and always includes an Execution step that migrates (performs the `execute_object_commands` Workflow command) between a source and a destination Environment.

Figure 3-1 shows a sample Mercury ITG Migrator Workflow.

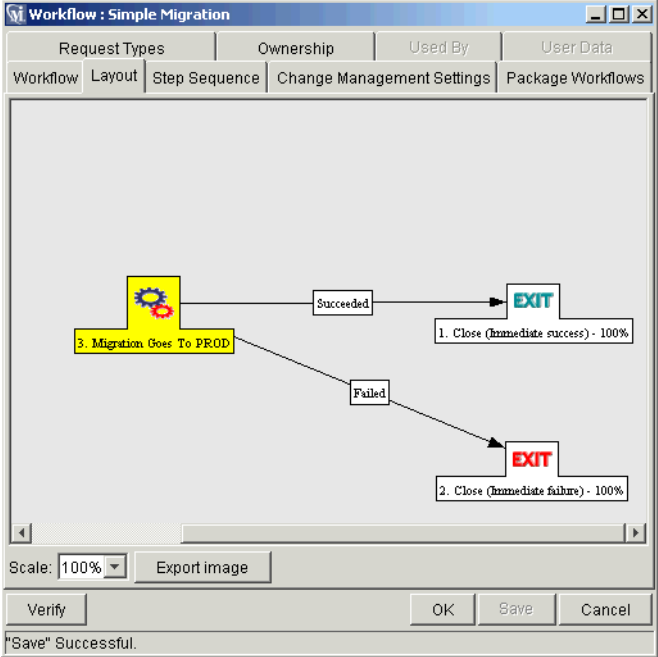


Figure 3-1 Sample Change Management Migrator Workflow

The first Workflow Step in this example is the necessary Execution step that actually migrates Mercury ITG configuration data between a source and a destination Environment. For the sample Workflow Step’s configuration, see [Figure 3-2](#).

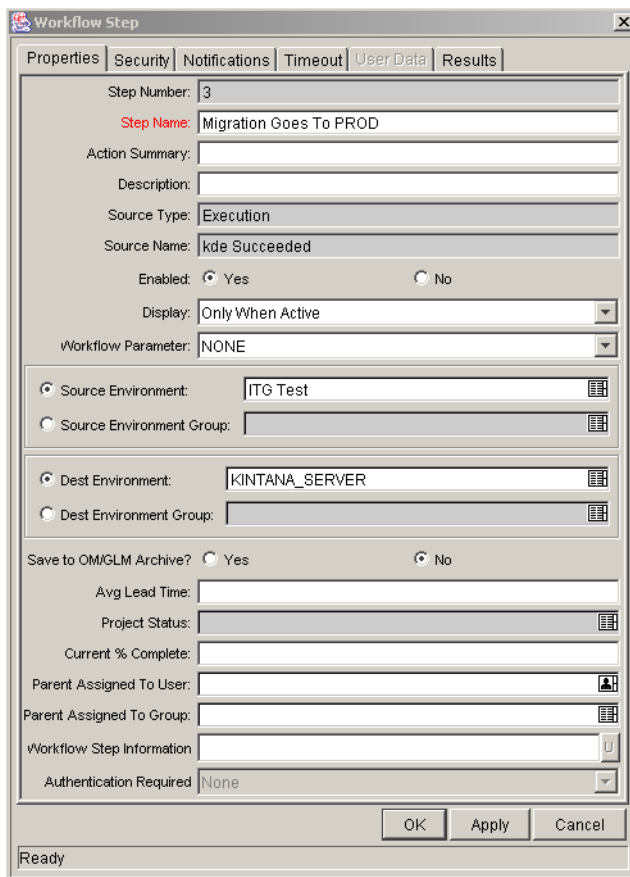


Figure 3-2 Sample Change Management Migrator Workflow Step



When building a Workflow for local import or extraction of content bundles, set both source and destination Environments to be KINTANA_SERVER.

As with all Workflows in Mercury ITG, this Workflow can be configured to conform to specific needs and processes surrounding critical Production and Development instances. For more information on defining and configuring Workflows, see *Configuring a Deployment System (Change Management)*.

Using Migrators

The actual process of migrating a configuration entity from one Mercury ITG instance to another is handled using standard Change Management methods: a Package is processed along a Workflow. The Package in a migration consists of Package Lines made from Migrator Object Types, and the Workflow is typically configured solely for migration purposes.

The following sections discuss the use of Migrators in more detail:

- [Assumptions](#)
- [Building the Migrator Package](#)
- [Using the Execution Log](#)

Assumptions

This document assumes the user has enough experience with Change Management to create and process a Package. For more detailed step-by-step instructions on creating a Package, see *Processing Packages (Change Management)*.

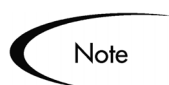
Building the Migrator Package

The actual Mercury ITG configuration information to be migrated is gathered in a Change Management Package. To specify Mercury ITG information and associated behaviors to be migrated:

1. Click the **Change Management** screen group and then the **Packages** screen.

The Package Workbench opens.

2. Click **New Package**.
3. From the Workflow auto-complete list, select the desired Migration Workflow.
4. Add Package Lines using the Object Migrator Object Types.
Specify any desired configurations and behaviors.



If the Migrator Object Type was modified, the source and destination passwords will need to be entered for import and export.

5. To save the Package without closing or submitting it for release, click **Save**.
To save the Package and close the window without submitting it for release, click **OK**.
6. To submit the Package for release along the Migration Workflow, click **Submit**.

Using the Execution Log

When a Package Line is migrated using a Migration Workflow, it automatically generates a report or execution log. This report is designed to answer the following questions:

- Did the migration succeed?
- If the migration succeeded, what was changed?
- If the migration failed, what problems were encountered?
- What parameters were used during the migration?
- If “Preview import” was chosen, what would the effects have been?

The report is designed to present users with the information that is most relevant to the task at hand, rather than an overabundance of detail.

Chapter 4 Using Migrators

Mercury ITG Migrators are Change Management Object Types. Like all Change Management Object Types, Migrators are used as Package Lines. Each Migrator is designed to migrate a specific Mercury ITG entity as well as all of its dependent objects from one Mercury ITG instance to another.



Example

It is possible to migrate a Mercury ITG Object Type from the Testing instance to the Production instance of Mercury ITG. When migrating the Object Type, the following information related to the Object Type is also migrated:

- Validations referenced by the Object Type fields
- Environments referenced by the Validations
- Special Commands referenced by Commands or Validations.

There are certain fields and behaviors that are common to all Mercury ITG Migrators, and others that are specific to particular Migrators. The following sections discuss in detail common Migrator behaviors and all Migrator Object Types:

- *Standard Migrator Fields and Behaviors*
- *Migrator Object Types*

Standard Migrator Fields and Behaviors

The following sets of fields and behaviors are common to all Migrators.

- *Migrator Action*

- *Preview Import?*
- *Target Entity*
- *Content Bundle Fields*
- *Import Behavior Fields*
- *Source Password*
- *Destination Password*
- *Internationalization*

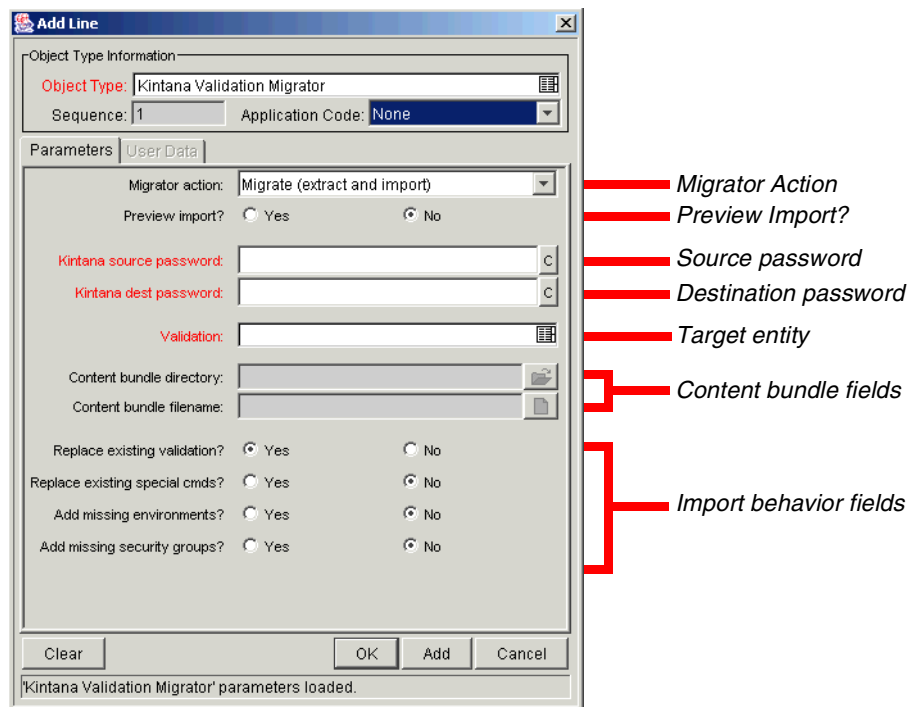


Figure 4-1 Migrator Parameters

Migrator Action

The Migrator Action field governs the migration of its particular Package Line by controlling whether other fields are enabled or required. The Migrator Action field is a drop down list with three possible values:

- **Migrate (extract and import)**
- **Extract only**
- **Import only**

Table 4-1 summarizes the behavior of the fields that are dependent with each Migrator Action value:

Table 4-1. Migrator Action Field Dependencies

Field and Field Set Names	Migrator Action value:		
	Migrate (extract and import)	Extract Only	Import Only
Preview Import?	Enabled	Disabled	Enabled
Target entity field	Required	Required	Disabled
Content bundle fields	Disabled	Enabled	Required
Import behavior fields	Enabled	Disabled	Enabled
Source password	Required	Required	Disabled
Destination password	Required	Disabled	Required

Preview Import?

This field enables the user to either perform the actual migration or to simulate it and report on the potential results. This is useful for determining the impact of a migration. If **Yes** is selected, then the object will not be migrated, but an execution log will be generated.

Target Entity

This auto-complete field takes the name of the target Mercury ITG entity to be migrated or extracted.



Note

Since the Validation values are determined by contacting the source Mercury ITG server, this auto-complete may take several seconds to display its values.

Content Bundle Fields

These fields behave differently depending on the Migrator action specified.

- Migrate (extract and import) — A content bundle is temporarily created, but the user is not prompted to provide any information related to this temporary file.
- Extract only — The user can specify the content bundle location and file name, or leave these fields blank and accept the default behavior. By default, the bundle will be created in the file system of the source Mercury ITG application server under the “transfers” directory. The filename is based on the type of entity migrated, its Package number, and Package Line number.
- Import only — The user must specify the content bundle location and file name. The file may be selected by browsing the file system of the destination Mercury ITG application server.

Import Behavior Fields

These fields modify the specific import behavior for the Mercury ITG entity to be migrated.

- Replace existing (entity)? — If the entity to be migrated already exists in the target Mercury ITG instance, the user can decide whether to replace it. The default value is **Yes**. If the entity does not exist in the destination instance, it will be created.
- Replace existing validations? — If the target entity references Validations that already exist in the target Mercury ITG instance, the user can decide whether to overwrite them. This default value is **No**. Regardless of the field’s value, any Validations that are missing from the destination instance will be automatically created.
- Add missing security groups? — If the entity to be migrated references Security Groups that are not included in the target instance, the user can select to add those Security Groups. Note that only the list of associated Access Grants, but not associated users, is transferred. The default value is **No**.

There may be other import behavior fields, depending on the specific Migrator Object Type. Any additional import behavior fields are detailed in “*Migrator Object Types*” on page 32.

Source Password

When the Migrator contacts the source application server, the current user’s name and password is used to log on to the other instance. Any application administrator performing a migration should already be linked to the proper

Security Group containing the Access Grant allowing access to the source Mercury ITG instance, Sys Admin: Migrate Mercury ITG Objects. However, if the password for the current user in the source instance is different from the password in the current instance, this field can be used to enter that override value.



Note

If the Migrator Object Type is modified, the source and destination passwords will need to be entered for import and export.

Destination Password

When the Mercury ITG Migrator contacts the destination Mercury ITG application server, the current user's name and password is used to log on to the other instance. Any application administrator performing a migration should already be linked to the proper Security Group containing the Access Grant allowing access to the destination Mercury ITG instance, Sys Admin: Migrate Mercury ITG Objects. However, if the password for the current user in the destination instance is different from the password in the current instance, this field can be used to enter that override value.



Note

If the Migrator Object Type is modified, the source and destination passwords will need to be entered for import and export.

Internationalization

This field may not be displayed on Migrator Object Types, but is enabled and defaulted to **Same language and character set**. To change this value, the field must first be displayed by editing its Migrator Object Type.

This field takes three possible values:

- **Same language and character set** — This is the default option, for migrating content between Mercury ITG instances running under the same language and character set configuration.
- **Different language or character set** — This option allows users to override character set or language incompatibilities within the same localization.

- **Different localization** — This option provides for the migration of content between instances belonging to different localizations: English to German, German to English, etc.

Migrator Object Types

The following sections describe the fields and behaviors particular to each Migrator Object Type:

- *Validation Migrator*
- *User Data Context Migrator*
- *Special Command Migrator*
- *Workflow Migrator*
- *Report Type Migrator*
- *Object Type Migrator*
- *Request Type Migrator*
- *Request Header Type Migrator*
- *Project Template Migrator*
- *Portlet Migrator*

Validation Migrator

The Validation Migrator Object Type contains all standard Migrator Object Type fields (see “*Standard Migrator Fields and Behaviors*” on page 27 for more detailed information).

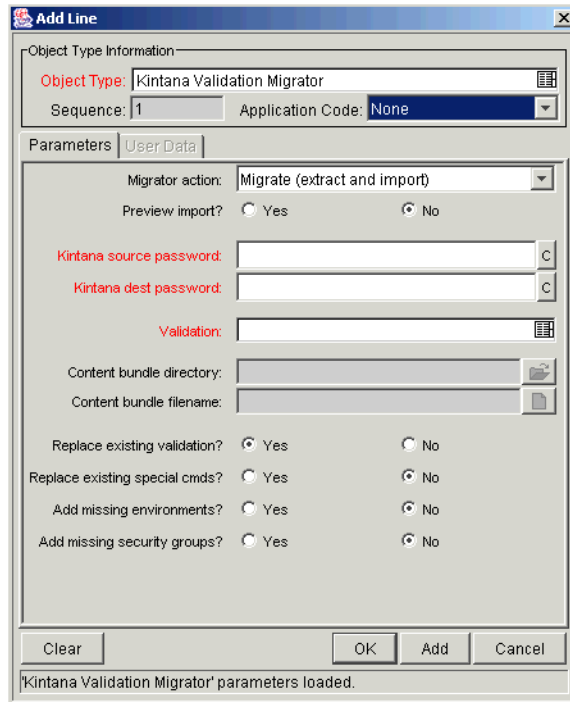


Figure 4-2 Validation Migrator

Validation Migrator-Specific Parameters

The Validation Migrator Object Type contains two additional import behavior fields:

- **Replace existing special cmds?** — If the Validation to be migrated references Mercury ITG Special Commands that already exist in the target Mercury ITG instance, the user can decide whether to replace them. This includes both Special Commands directly referenced by the Validation, and also Special Commands referenced by these Special Commands. The default value is **No**. Regardless of the value, any Special Commands missing from the destination instance will always be created automatically.
- **Add missing environments?** — If the Validation to be migrated references Environments or Environment Groups that do not exist in the target Mercury ITG instance, the user can decide whether to create them (assuming that the option has been marked **Yes**). However, only the Environment header information and User Data are transferred. Application codes and Extension-specific Environment tabs are not transferred. The default value is **No**.

Similarly, Environment Group application code info is not transferred. If an Environment Group already exists in the destination instance, it will not be updated with Environments that were added in the source instance. After migrating, Environment data should be confirmed and completed manually, if any Environments have been created by the Migrator.

Validation Migrator Considerations

Validation values can also carry context-sensitive User Data. When migrating Validation values that have such fields, the User Data configuration should be set up manually in the destination instance before migration for the transfer to succeed.

User Data Context Migrator

The User Data Context Migrator Object Type contains all standard Migrator Object Type fields (see *“Standard Migrator Fields and Behaviors”* on page 27 for more detailed information).

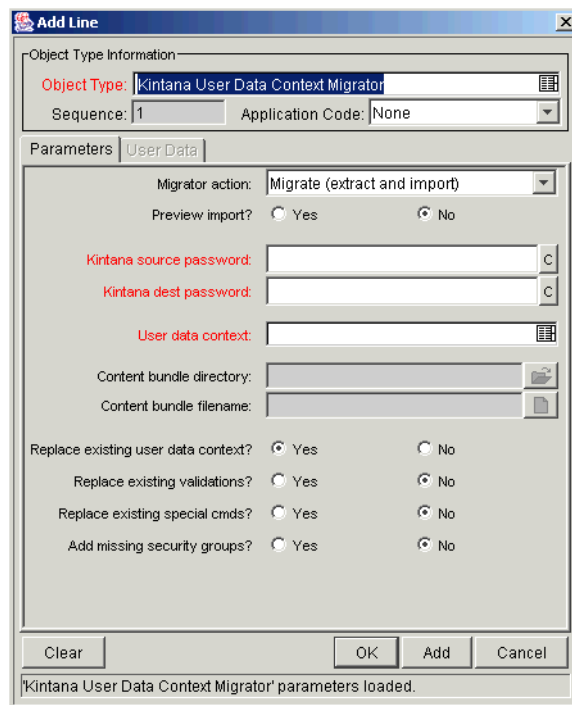


Figure 4-3 User Data Context Migrator

User Data Context Migrator-Specific Parameter

The User Data Context Migrator Object Type contains one additional import behavior field:

- Replace existing special cmds? — If the User Data Context to be migrated references Mercury ITG Special Commands that already exist in the target Mercury ITG instance, the user can decide whether to replace them. This applies to Special Commands referenced by Validations that are themselves referenced by the User Data Context. The default value is **No**. Regardless of the value, any Special Commands missing from the destination instance will always be created automatically.

Special Command Migrator

The Special Command Migrator Object Type contains all standard Migrator Object Type fields (see “*Standard Migrator Fields and Behaviors*” on page 27 for more detailed information), except for the Replace existing validations field. This migrator does not have any additional import behavior fields.

The screenshot shows a dialog box titled "Add Line" with a close button in the top right corner. The dialog is divided into several sections:

- Object Type Information:** Contains a text field for "Object Type" with the value "Kintana Special Command Migrator", a "Sequence" field with the value "1", and an "Application Code" dropdown menu set to "None".
- Parameters:** A tabbed section with a "User Data" tab selected. It contains:
 - "Migrator action:" dropdown menu set to "Migrate (extract and import)".
 - "Preview import?" radio buttons: "Yes" is unselected, "No" is selected.
 - "Kintana source password:" text field with a "C" button to its right.
 - "Kintana dest password:" text field with a "C" button to its right.
 - "Special command:" text field with a list icon to its right.
 - "Content bundle directory:" text field with a folder icon to its right.
 - "Content bundle filename:" text field with a file icon to its right.
 - "Replace existing special cmd?" radio buttons: "Yes" is selected, "No" is unselected.
 - "Add missing security groups?" radio buttons: "Yes" is unselected, "No" is selected.
- Buttons:** "Clear", "OK", "Add", and "Cancel" buttons are located at the bottom.
- Status Bar:** A message at the bottom reads "'Kintana Special Command Migrator' parameters loaded."

Figure 4-4 Special Command Migrator

Workflow Migrator

The Workflow Migrator Object Type contains all standard Migrator Object Type fields (see *“Standard Migrator Fields and Behaviors”* on page 27 for more detailed information).

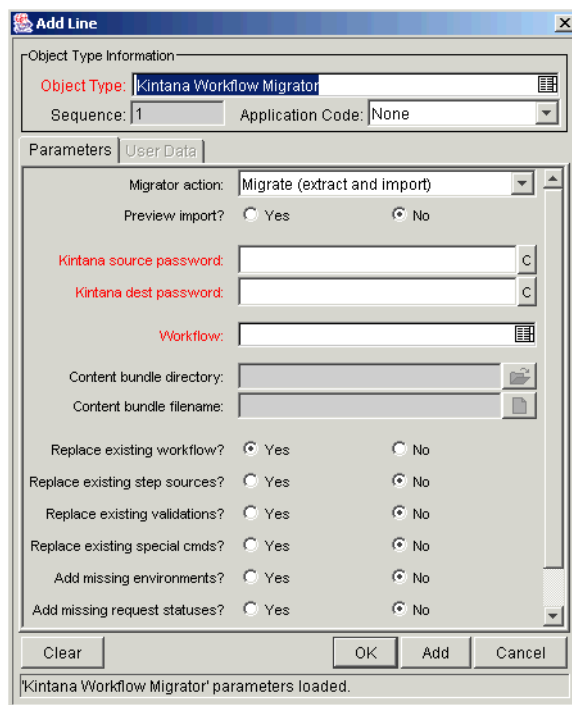


Figure 4-5 Workflow Migrator

Workflow Migrator-Specific Parameters

The Workflow Migrator Object Type contains the following additional import behavior fields:

- **Replace existing special cmds?** — If the Workflow to be migrated references Mercury ITG Special Commands that already exist in the target Mercury ITG instance, the user can decide whether to replace them. This includes both Special Commands directly referenced by the Workflow, and also Special Commands referenced by these Special Commands. Special Commands in Validations referenced by the Workflow are also migrated. The default value is **No**. Regardless of the value, any Special Commands missing from the destination instance will always be created automatically.
- **Replace existing step sources?** — If the Workflow to be migrated references Workflow Decision and Execution Step Sources that already exist in the

target Mercury ITG instance, the user can decide whether to replace them. If the existing Step Sources are already in use by Workflows in the destination instance, however, certain fields cannot be changed even if Replace Existing Step Sources? is set to **Yes**. These fields include Workflow Scope, Validation, and Decision Type.

- Add missing environments? — If the Workflow to be migrated references Environments or Environment Groups that do not exist in the target Mercury ITG instance, the user can decide whether to create them. However, only the Environment header information and User Data are transferred. Application codes and Extension-specific Environment tabs are not transferred. The default value is **No**.

Similarly, Environment Group application code info is not transferred. If an Environment Group already exists in the destination instance, it will not be updated with Environments that were added in the source instance. After migrating, Environment data should be confirmed and completed manually, if any Environments have been created by the Migrator.

- Add missing request statuses? -- If the Workflow to be migrated references Request statuses that do not exist in the target Mercury ITG instance, the user can decide whether to create them. The default value is **No**.

Workflow Migrator Considerations

The Workflow Migrator also transfers the following information:

- Subworkflows referenced by Workflow Steps
- Special Commands referenced by Command steps
- Workflow Step Sources referenced by Workflow Steps
- Validations referenced by parameters or Workflow Step Sources
- Environments and Environment Groups referenced by Workflow Steps
- Environments referenced by Environment Groups referenced by Workflow Steps
- Environments referenced by Validations
- Special Commands referenced by Validations
- Special Commands referenced by Workflow Step Sources
- Special Commands referenced by other Special Commands already referenced elsewhere

- Security Groups referenced by Workflow Steps
- Request Statuses referenced by Workflow Steps
- Notifications referenced by Workflow Steps
- Notification Intervals referenced by Notifications
- Security Groups referenced by Notifications
- Ownership Group information for the Workflow and Workflow Steps

If a Notification in a Workflow uses a Notification Interval that does not exist in the destination instance, this Notification Interval will be created. The Workflow Migrator will never replace an existing Notification Interval.

The Workflow Migrator will transfer entity restriction references to Object Types, but will not create an Object Type. If the referenced Object Type does not exist in the destination instance, the reference will be discarded in transit. A message to that effect will appear in the migration's execution log.

The Workflow Migrator will transfer references to Request Types, but will not create a Request Type. If the referenced Request Type does not exist in the destination instance, the reference will be discarded in transit. A message to that effect will appear in the migration's execution log.



Note

Circular references between Workflows and Request Types could make it necessary to migrate either a Workflow or Request Type twice.

1. A new Request Type referring to a new Workflow is migrated. Since the new Workflow does not exist in the destination instance, all references to that Workflow are dropped in transit.
2. The new Workflow is migrated.
3. The new Request Type is migrated again. This time, since the Workflow it refers to exists, the references are preserved.

Replacing an Existing Workflow

When using the Workflow migrator to make changes to an existing process that is already in use (by Requests or Package Lines), there are some restrictions. These restrictions help to ensure that these existing Requests or Package Lines will not be damaged by the migration.

Specifically, the Workflow migration will not succeed unless the migrator logic can find a Workflow Step in the incoming process that corresponds to each step in the previous process. The following conditions are used to match Workflow Steps between instances:

- The Step Source (the particular decision, execution, or condition) of a Workflow Step is always used for matching Workflow Steps to each other. If the Step Source is not identical, then two Workflow Steps cannot be considered to match.
- When both the incoming and previous Workflows assign a unique name to each Workflow Step, these Workflow Step names are used in combination with the Step Source to assess matching.
- When a Workflow Step name is repeated within either process, the step sequence is used instead, in combination with the Step Source, to assess matching.

Note

The Workflow Migrator is not able to handle a single change in which both the names of existing Workflow Steps and the step sequence of existing Workflow Steps has changed.

To change both the names and step sequences of a Workflow:

1. Change step names, but do not change any step sequences. Migrate the changed Workflow.
2. Change step sequences, but do not change any step names. Migrate the changed Workflow a second time.

Due to this matching restriction, each open Request will still be the same process step following the migration as it was prior to the migration. The migration may have changed the name of this step, but it will not have transitioned any Request Workflows.

It is important to note, however, that the migrator does not prevent the removal of outgoing transitions from Workflow Steps. It is therefore important to avoid “stranding” open Requests at a Workflow Step that will be deprecated. When deprecating a process step, you will want to remove incoming transitions, but still want to leave at least one outgoing transition from the step. This will allow open Requests to move forward.

The migration’s execution log will contain a table listing old and new Workflow Steps. Mercury ITG recommends using the **Preview import** mode

first when replacing an existing Workflow, and inspecting this table of matched Workflow Steps before running such a Workflow migration in non-preview mode.

Deprecating a Workflow

When the changes to a Workflow are extensive, it is possible to deprecate the existing Workflow and bring the changes into the production instance as a new Workflow. One advantage of implementing the changes as a new Workflow is simplicity, since the new Workflow is not required to contain all of the steps of the old Workflow for backward compatibility.

To bring a new Workflow into the production instance:

1. Rename the existing Workflow and disable it in production.

Disabling the Workflow removes it from lists of Workflow options when creating new Requests. Requests that are already in process will continue to follow the old Workflow until they close, unless each is manually shifted to the new process and transitioned to an appropriate point in the process. Existing defaulting rules and other configurations will also continue to refer to the old Workflow regardless of the change of name.

2. Migrate the new version of the Workflow into the production instance, under the original name.

Since the production instance no longer contains a Workflow by this name, the migration will treat this as a new Workflow.

3. Following the migration, it is possible to update defaulting rules in Request Types to reference this new Workflow.

This can be done manually, or by migrating in versions of the Request Types that refer to the new Workflow by the original name.

Report Type Migrator

The Report Type Migrator Object Type contains all standard Migrator Object Type fields (see *“Standard Migrator Fields and Behaviors”* on page 27 for more detailed information).

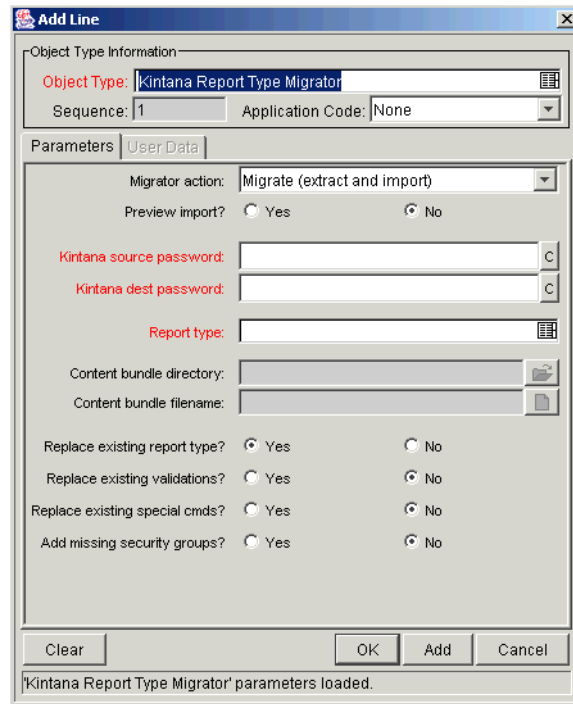


Figure 4-6 Report Type Migrator

Report Type Migrator-Specific Parameter

The Report Type Migrator Object Type contains one additional import behavior field:

- **Replace existing special cmds?** — If the Report Type to be migrated references Mercury ITG Special Commands that already exist in the target Mercury ITG instance, the user can decide whether to replace them. This includes both Special Commands directly referenced by the Report Type, and also Special Commands referenced by these Special Commands, and so forth. The default value is **No**. Regardless of the value, any Special Commands missing from the destination instance will always be created automatically.

Report Type Migrator Considerations

The Report Type Migrator also transfers the following information:

- Special Commands referenced by command steps
- Validations referenced by fields
- Environments referenced by Validations

- Special Commands referenced by Validations
- Special Commands referenced by other Special Commands
- Ownership Group information for the Report Type



Note

The Report Type Migrator transfers references to Environments from Validations, but will not create an Environment. If the referenced Environment does not exist in the destination instance, the migration will fail. In this case, the missing Environment should be created manually in the destination instance.

Migrations and Entity Restrictions

A Report Type may refer to Security Groups through Entity Restrictions. The Report Type Migrator will transfer references to Security Groups, but will not create a Security Group. The behavior is described below:

- If the referenced Security Group does not exist in the destination instance, the reference will be discarded in transit. A message to that effect will appear in the migration's execution log.
- If the source instance contains Security Groups that do not exist in the destination instance at the time of migration, the Entity Restrictions for the migrated Report Type will not be accurate.

Report Types that contain Entity Restrictions should therefore be verified manually in the destination instance following migration. This ensures that Security Groups are configured as desired.

Object Type Migrator

The Object Type Migrator Object Type contains all standard Migrator Object Type fields (see "*Standard Migrator Fields and Behaviors*" on page 27 for more detailed information).

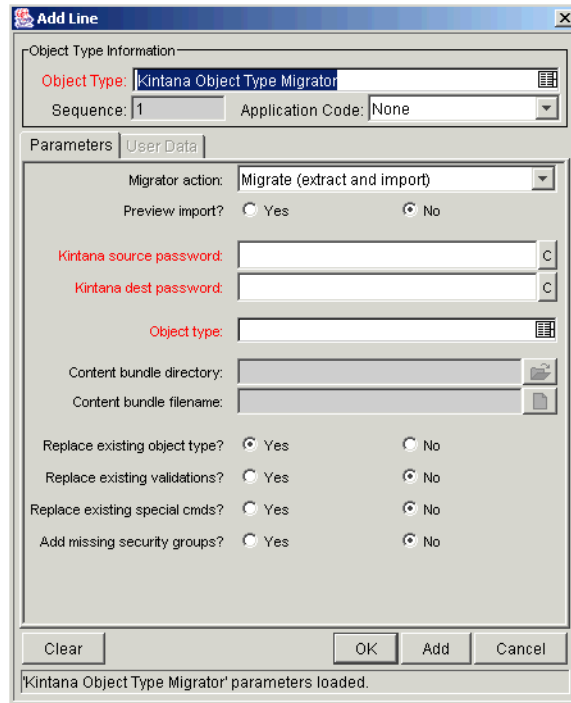


Figure 4-7 Object Type Migrator

Object Type Migrator-Specific Parameter

The Object Type Migrator Object Type contains one additional import behavior field:

- **Replace existing special cmds?** — If the Object Type to be migrated references Mercury ITG Special Commands that already exist in the target Mercury ITG instance, the user can decide whether to replace them. This includes both Special Commands directly referenced by the Object Type, Special Commands referenced by these Special Commands, and Special Commands referenced by Validations in the Object Type. The default value is **No**. Regardless of the value, any Special Commands missing from the destination instance will always be created automatically.

Object Type Migrator Considerations

The Object Type Migrator also transfers the following information:

- Special Commands referenced by Command steps
- Validations referenced by fields
- Environments referenced by Validations

- Special Commands referenced by Validations
- Special Commands referenced by other Special Commands
- Ownership Group information for the Object Type



Note

The Object Type Migrator will transfer references to Environments from Validations, but will not create an Environment. If the referenced Environment does not exist in the destination instance, the migration will fail. In this case, the missing Environment should be created manually in the destination instance.

Request Type Migrator

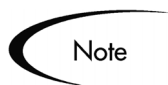
The Request Type Migrator Object Type contains all standard Migrator Object Type fields (see *“Standard Migrator Fields and Behaviors”* on page 27 for more detailed information).

Figure 4-8 Request Type Migrator

Request Type Migrator-Specific Parameters

The Request Type Migrator Object Type contains three additional import behavior fields:

- Replace existing req hdr type? — If the Request Type to be migrated references a Request Header Type that already exists in the target Mercury ITG instance, the user can decide whether to replace it. The default value is **No**.
- Replace existing special cmds? — If the Request Type to be migrated references Mercury ITG Special Commands that already exist in the target Mercury ITG instance, the user can decide whether to replace them. This includes both Special Commands directly referenced by the Request Type, and also Special Commands referenced by these Special Commands, and so forth. The default value is **No**. Regardless of the value, any Special Commands missing from the destination instance will always be created automatically.
- Add missing request statuses? — If the Request Type to be migrated references Request statuses that do not exist in the target Mercury ITG instance, the user can decide whether to create them. The default value is **No**. A message will appear in the execution log for each referenced Request Status that is not created.



Note

If this field is set to **No** and one of the missing Request Statuses is the initial status of the Request Type, the migration will fail. In this case, the Request Status for the initial status may be created manually.

Request Type Migrator Considerations

The Request Type Migrator also transfers the following information:

- Request Header Types referenced by the Request Type
- Special Commands referenced by Command steps
- Validations referenced by fields of the Request Type or Request Header Type
- Environments referenced by Validations
- Special Commands referenced by Validations
- Special Commands referenced by other Special Commands already referenced elsewhere

- Request Statuses referenced by the Request Type
- Security Groups referenced by the Request Type (in the **Access** tab)
- Workflows referenced by the Request Type
- Notifications referenced by the Request Type
- Ownership Group information for the Request Type

The Request Type Migrator will transfer references to Environments from Validations, but will not create an Environment. If the referenced Environment does not exist in the destination instance, the migration will fail. In this case, the missing Environment should be created manually in the destination instance.

Simple Default Rules, defined in the Request Type **Rules** tab, may reference Users, Workflows, or other objects. The Request Type Migrator will transfer these references, but will not create a missing User or Workflow. If the referenced User or Workflow does not exist in the destination instance, the reference will be discarded in transit, and a message to that effect will appear in the migration's execution log. Advanced Default Rules should be manually reconfirmed after migration.



Note

Circular references between Request Types and Workflows could make it necessary to migrate either a Request Type or Workflow twice.

1. A new Request Type referring to a new Workflow is migrated. Since the new Workflow does not exist in the destination instance, all references to that Workflow are not included in the new instance destination.
2. The new Workflow is migrated.
3. The new Request Type is migrated again. This time, since the Workflow it refers to exists, the references are included in the destination instance.

Request Header Type Migrator

The Request Header Type Migrator Object Type contains all standard Migrator Object Type fields (see "[Standard Migrator Fields and Behaviors](#)" on page 27 for more detailed information).

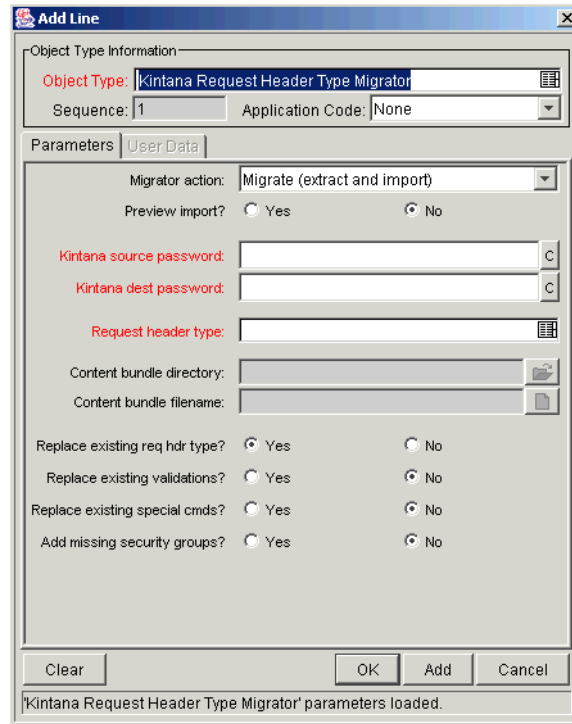


Figure 4-9 Request Header Type Migrator

Request Header Type Migrator-Specific Parameter

The Request Header Type Migrator Object Type contains one additional import behavior field:

- **Replace existing special cmds?** — If the Request Header Type to be migrated references Mercury ITG Special Commands that already exist in the target Mercury ITG instance, the user can decide whether to replace them. This includes both Special Commands directly referenced by the Request Header Type and Special Commands referenced by these Special Commands. The default value is **No**. Regardless of the value, any Special Commands missing from the destination instance will always be created automatically.

Request Header Type Migrator Considerations

The Request Header Type Migrator also transfers the following information:

- Validations referenced by fields
- Environments referenced by Validations

- Special Commands referenced by Validations
- Special Commands referenced by other Special Commands
- Ownership Group information for the Request Header Type



Note

The Request Header Type Migrator transfers references to Environments from Validations, but will not create an Environment. If the referenced Environment does not exist in the destination instance, the migration will fail. In this case, the missing Environment should be created manually in the destination instance.

Project Template Migrator

The Project Template Migrator Object Type contains all standard Migrator Object Type fields (see *“Standard Migrator Fields and Behaviors”* on page 27 for more detailed information).

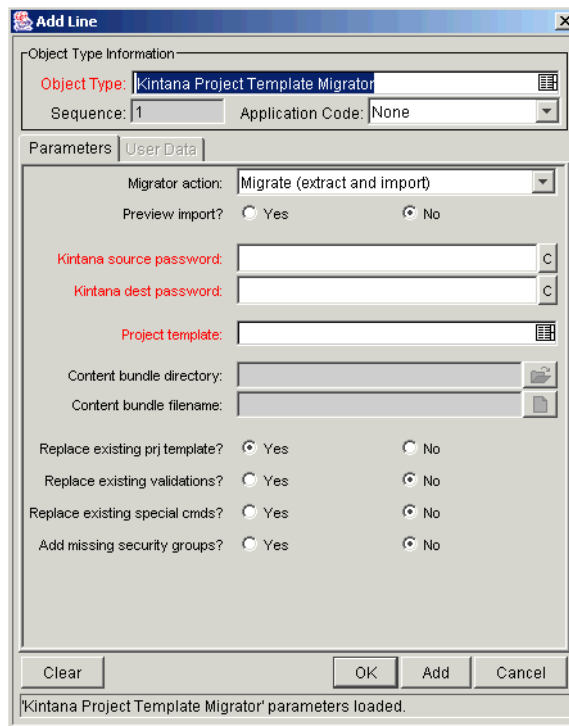


Figure 4-10 Project Template Migrator

Project Template Migrator-Specific Parameter

The Project Template Migrator Object Type contains one additional import behavior field:

- Replace existing special cmds? — If the Validation to be migrated references Mercury ITG Special Commands that already exist in the target Mercury ITG instance, the user can decide whether to replace them. This includes both parent and children Special Commands. The default value is **No**. Regardless of the value, any Special Commands missing from the destination instance will always be created automatically.

Project Template Migrator Considerations

The Project Template Migrator also transfers the following information:

- Special Commands referenced by Command steps
- Validations referenced by fields
- Environments referenced by Validations
- Special Commands referenced by Validations
- Special Commands referenced by other Special Commands already referenced elsewhere
- Security Groups referenced by Resource lists
- Notifications referenced by Project Tasks
- Notification Intervals referenced by Notifications
- Security Groups referenced by Notifications
- Ownership Group information for the Project Template
- Project Team tab information

Project Templates may reference Users and Security Groups. The Project Template Migrator will transfer these references, but will not create a missing User or Security Group. If the referenced User or Security Group does not exist in the destination instance, the reference will be discarded in transit. A message to that effect will appear in the migration's execution log.

A Project Template may also contain references to other Project Templates that have been used to create the current Template. The Project Template Migrator will transfer these references, but will not create a missing nested Project Template.



Note

To ensure that these references are preserved, any Project Templates that have been nested inside other Project Templates should be migrated first. Otherwise, if the referenced nested Project Template does not exist in the destination instance, the reference will be discarded in transit. A message to that effect will appear in the migration's execution log.

Portlet Migrator

The Portlet Migrator Object Type contains all standard Migrator Object Type fields (see *“Standard Migrator Fields and Behaviors”* on page 27 for more detailed information). When migrating a Portlet to replace an existing enabled Portlet in an active instance, the changes migrated will be propagated to every user that has added the same Portlet to their Dashboard.

The screenshot shows the 'Add Line' dialog box for the Portlet Migrator. The 'Object Type Information' section includes 'Object Type: Kintana Portlet Migrator', 'Sequence: 1', and 'Application Code: None'. The 'Parameters' tab is active, showing 'Migrator action: Migrate (extract and import)', 'Preview import?' with 'No' selected, 'Kintana source password', 'Kintana dest password', 'Portlet', 'Content bundle directory', and 'Content bundle filename'. There are also radio buttons for 'Replace existing portlet?' (Yes), 'Replace existing validations?' (No), and 'Add missing security groups?' (No). Buttons for 'Clear', 'OK', 'Add', and 'Cancel' are at the bottom. A status bar at the bottom indicates 'Kintana Portlet Migrator' parameters loaded.

Figure 4-11 Portlet Migrator

Appendix

A

Using Migrators to Archive

The Content Bundle fields in Mercury ITG Migrator Object Types can be used to archive configuration entities in a version-control system.

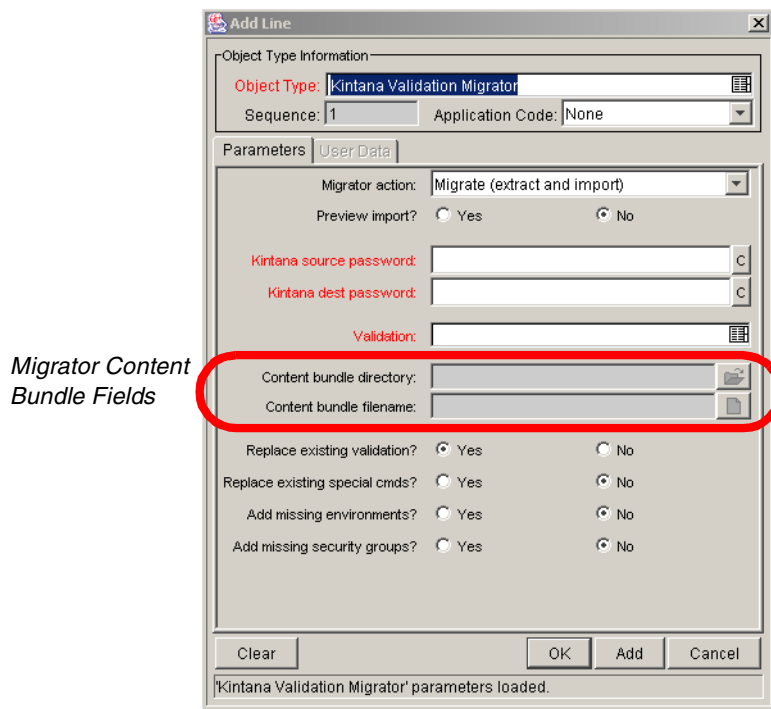


Figure A-1 Migrator Object Type Content Bundle Fields

To archive configuration entities in the version-control system:

1. Specify a path and filename for the Migrator content bundle.
2. Run a migration with the Migrator action field set to **Extract Only**.
3. Check the ZIP file into the version-control system.

Make sure to keep track of the Mercury ITG version number on each content bundle. A single content bundle is compatible only with other instances of the same version.



Example

A content bundle foo.zip is extracted from a Mercury ITG 4.6 instance. This bundle can only be imported into other Mercury ITG 4.6 instances.

To use this bundle with Mercury ITG version 5.x, import the bundle into a Mercury ITG 4.6 instance, and then upgrade the instance to Mercury ITG 5.x after the import.

Index

A

- Action Fields 28
- Add missing environments 33, 37
- Add missing request statuses 37, 45
- Add missing security groups 30
- Archiving with Migrators 51

B

- Best Practices 14
 - deprecating a workflow 16
 - migrating request type with rules 16
 - nested project templates 17
 - replacing a workflow 14
 - request type and workflow 16
 - running migration 14

C

- Configure Environments 18
- Content Bundle Fields 29

D

- Deprecating a Workflow 16

- Destination Password 31
- Different language or character set 31
- Different localization 32

E

- Environment
 - for source instance 19
- Environments
 - configuring 18
 - for the destination instance 20
- Execution Log 26
- Extract only 30

I

- Import Behavior Fields 30
- Import only 30
- Internationalization 31

M

- Migrate (extract and import) 30
- Migrator Architecture 13
- Migrator Object Types 32
 - action fields 28
 - content bundle fields 29
 - destination password 31
 - fields and behaviors 27

- import behavior fields 30
- internationalization field 31
- Object Type migrator 42
- Portlet migrator 50
- preview import field 29
- Project Template migrator 48
- Report Type migrator 40
- Request Header Type migrator 46
- Request Type migrator 44
 - source password 30
 - special command migrator 35
 - target entity field 29
 - user data context migrator 34
- Validation migrator 32
- Workflow migrator 36

Migrators

- architecture 13
- assumptions for users 25
- best practices 14
- building package 25
- checking server.conf file 22
- configuring destination environment 20
- configuring source environment 19
- defining migration Workflow 22
- environments overview 10
- execution log 26

- key concepts 7
- local import and/or extraction of content bundles 24
- Object Type fields and behaviors 27
- object types 32
- overview 1, 7
- ownership overview 11
- setup 18
- supported entities 12
- using 25
- using to archive 51
- workflows overview 9
- XML files 8
- ZIP archive 8

O

- Object Type Migrator 42
 - content transferred 43

P

- Package 25
- Portlet Migrator 50
- Preview Import 29
- Project Template Migrator 48
 - content transferred 49
 - nested Project Templates 17, 49

R

- Replace existing (entity) 30
- Replace existing req hdr type 45
- Replace existing special cmds 33, 35, 36, 41, 43, 45,

- 47, 49

- Replace existing validations 30
- Replacing a Workflow 14, 38
- Report Type Migrator 40
 - migrations and entity restrictions 42
- Request Header Type Migrator 46
 - content transferred 47
- Request Type Migrator 44
 - content transferred 45
 - Workflow considerations 46
- Request Type Rules 16
- Rules 16, 46

S

- Same language and character set 31
- server.conf 22
- Source Password 30
- Special Command Migrator 35
- Supported Entities 12

T

- Target Entity 29

U

- User Data Context Migrator 34

V

- Validation Migrator 32

W

- Workflow
 - defining for migration 22
 - deprecating 16, 40
 - replacing existing 14, 38
- Workflow Migrator 36
 - Request Type considerations 38
- Workflow Step 24