KINTANA™

# Kintana Migrators

**Version 5.0.0**

Publication Number: KintanaMigration-0603A

Kintana Version 5.0.0



**Kintana, Inc.**
1314 Chesapeake Terrace, Sunnyvale, California 94089
Telephone: (408) 543-4400
Fax: (408) 752-8460
http://www.kintana.com

# Contents

# Chapter 1 Introduction

Kintana Migrators are used to move Kintana configuration data such as Validations, Workflows, and Request Types between instances (installations) of Kintana.


Kintana Migrators are used to move configuration data between Kintana instances of the same version.

Kintana Migrators provide change control to Kintana configuration data through their use of Deliver 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.

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

Kintana Migrators also let you document your 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 Kintana Migrator Object Types, allowing organizations to maintain a detailed record of the specific changes made to their production Kintana configurations.

This document discusses the following topics:

- *Using Migrators to Archive*

# Who Should Read this Guide

This document provides details for configuring and using the Kintana Migrator Object Types to migrate or archive data from Kintana instances.

This instance administration guide is primarily intended for users responsible for maintaining and updating instances of Kintana.


You must have a Deliver 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.

# How to Use this Guide

This document provides background information and details for using the Kintana Migrators to migrate Kintana configuration data between instances. Since the Migrators are a particular kind of Object Type that use Kintana Deliver Workflows and Object Type Commands, you may want to use this document in conjunction with *"Configuring a Deployment System in Kintana"* to get a comprehensive instruction set.

Navigate to one of the following chapter topics or use the Index to find information related to key words.

## What this Guide is NOT

This instance administration guide is not meant to provide detailed information on every screen and field in Kintana. Nor will this document provide comprehensive instructions on configuring Workflows and Object Types. For detailed screen and field information refer to the Kintana Application Reference Guides, accessible from the Kintana Library. See *"Additional Resources"* on page 3 for a list of the most relevant documents.

# Additional Resources

Kintana provides the following additional resources to help you successfully implement, configure, maintain and fully utilize your Kintana installation:

- *Kintana Documentation*
- *Kintana Services*
- *Kintana Education*
- *Kintana Support*

### *Kintana Documentation*

Kintana product documentation is linked from the Kintana Library page. This page is accessed by:

- Selecting **HELP > KINTANA LIBRARY** from the Kintana Workbench menu.
- Selecting **HELP > CONTENTS AND INDEX** from the menu bar on the HTML interface. You can then click the **KINTANA LIBRARY** link to load the full list of product documents.

Kintana organizes their documents into a number of user-based categories. The following section defines the document categories and lists the documents currently available in each category.

- *Kintana Business Application Guides*
- *User Guides*
- *Kintana Application Reference Guides*
- *Kintana Instance Administration Guides*
- *External System Integration Guides:*

- *Kintana Solution Guides*
- *Kintana Accelerator Guides*

## Kintana Business Application Guides

Provides instructions for modeling your business processes in Kintana. These documents contain process overviews, implementation instructions, and detailed examples.

- Configuring a Request Resolution System (Create)
- Configuring a Deployment and Distribution System (Deliver)
- Configuring a Release Management System
- Configuring the Kintana Dashboard
- Managing Your Resources with Kintana
- Kintana Reports

## User Guides

Provides end-user instructions for using the Kintana products. These documents contain comprehensive processing instructions.

- Processing Packages (Deliver) User Guide
- Processing Requests (Create) User Guide
- Processing Projects (Drive) User Guide
- Navigating the Kintana Workbench:
  Provides an overview of using the Kintana Workbench
- Navigating Kintana:
  Provides an overview of using the Kintana (HTML) interface

## Kintana Application Reference Guides

Provides detailed reference information on other screen groups in the Kintana Workbench. Also provides overviews of Kintana's command usage and security model.

- Reference: Using Commands in Kintana
- Reference: Kintana Security Model

- Workbench Reference: Deliver
- Workbench Reference: Configuration
- Workbench Reference: Create
- Workbench Reference: Dashboard
- Workbench Reference: Sys Admin
- Workbench Reference: Drive
- Workbench Reference: Environments

### Kintana Instance Administration Guides

Provides instructions for administrating the Kintana instances at your site. These documents include information on user licensing and archiving your Kintana configuration data.

- Kintana Migration
- Kintana Licensing and Security Model

### External System Integration Guides:

Provides information on how to use Kintana's open interface (API) to access data in other systems. Also discusses Kintana's Reporting meta-layer which can be used by third party reporting tools to access and report on Kintana data.

- Kintana Open Interface

### Kintana Solution Guides

Provides information on how to configure and use functionality associated with the Kintana Solutions. Each Kintana Solution provides a User Guide for instructions on end-use and a Configuration Guide for instructions on installing and configuring the Solution.

### Kintana Accelerator Guides

Provides information on how to configure and use the functionality associated with each Kintana Accelerator. Kintana Accelerator documents are only provided to customers who have purchased a site-license for that Accelerator.


Kintana provides documentation updates in the Download Center section of the Kintana Web site (http://www.kintana.com/support/download/download_center.htm).

A username and password is required to access the Download Center. These were given to your Kintana administrator at the time of product purchase. Contact your administrator for information on Kintana documentation or software updates.

## *Kintana Services*

Kintana is a strategic partner to its clients, assisting them in all aspects of implementing a Kintana technology chain - from pilot project to full implementation, education, project turnover, and ongoing support. Our Total Services Model tailors solution and service delivery to specific customer needs, while drawing on our own knowledgebank and best practices repository. Learn more about Kintana Services from our Web site:

*http://www.kintana.com/services/services.shtml*

## *Kintana Education*

Kintana has created a complete product training curriculum to help you achieve optimal results from your Kintana applications. Learn more about our Education offering from our Web site:

*http://www.kintana.com/services/education/index.shtml*

## *Kintana Support*

Kintana provides web-based interactive support for all products in the Kintana product suite via Contori.

*http://www.contori.com*

Login to Contori to enter and track your support issue through our quick and easy resolution system. To log in to Contori you will need a valid email address at your company and a password that will be set by you when you register at Contori.

# Chapter 2 Key Concepts

This chapter defines the common concepts and terms related to Kintana Migrators. Knowledge of these terms will help you gain a more thorough understanding of Kintana Migrators.

The following concepts and terms are discussed:



## Migrator

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


You can migrate a Kintana Request Type from your Testing instance to your Production instance of Kintana. When you migrate 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.

# Migrators and XML

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

A content bundle can be imported into another Kintana 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 Kintana instance.

Note

We recommend that you do not unzip this file manually except for debugging purposes, and only then if you are reasonably certain you know what the consequences will be.

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*

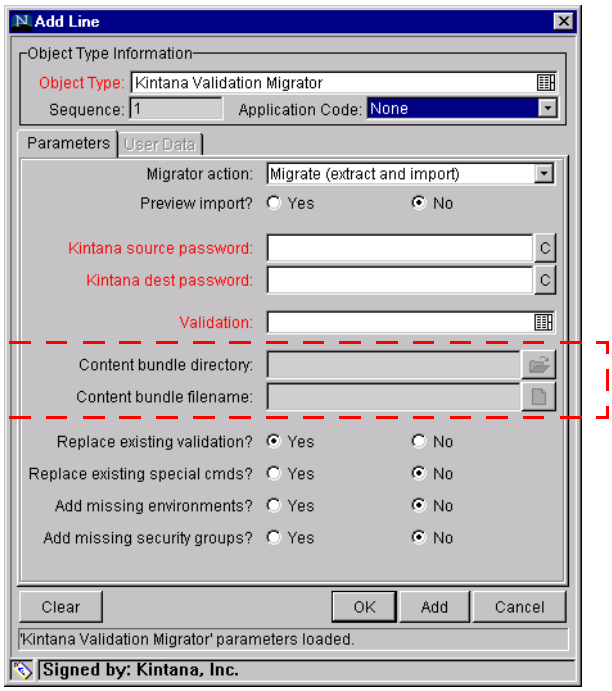


*Figure 2-1 Migrator Object Type Content Bundle Fields*

# Using Workflows

Like all Deliver 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 Deliver 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 19.

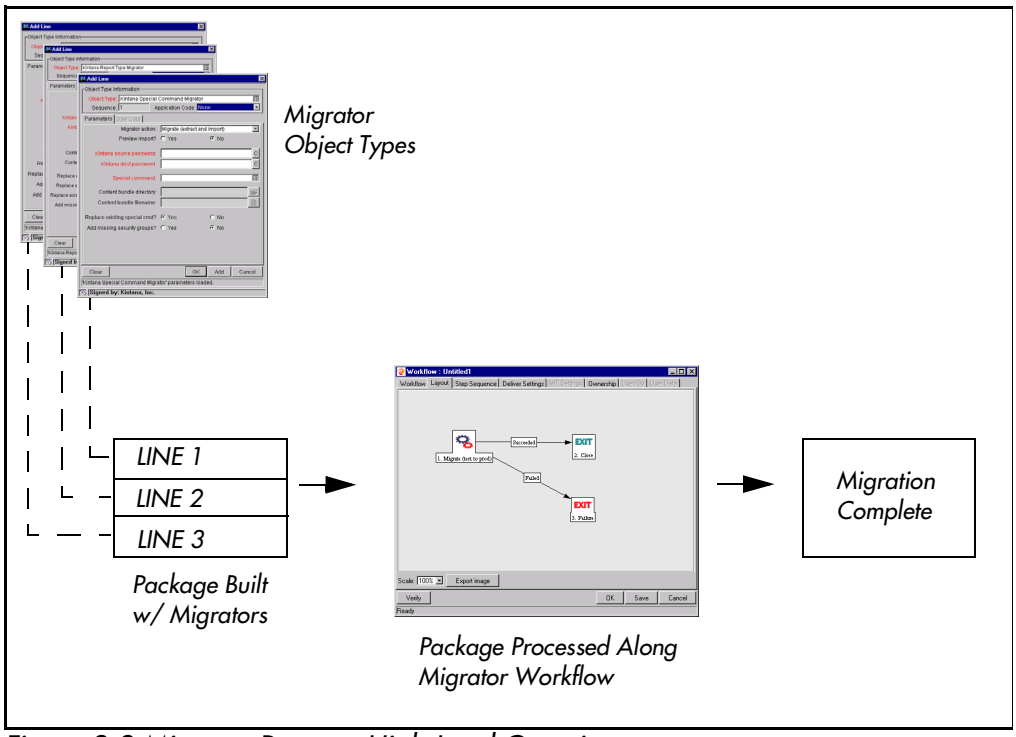



*Figure 2-2 Migrator Process: High-Level Overview*

# Environments for Migrators

Before a migration takes place, Kintana 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 19.

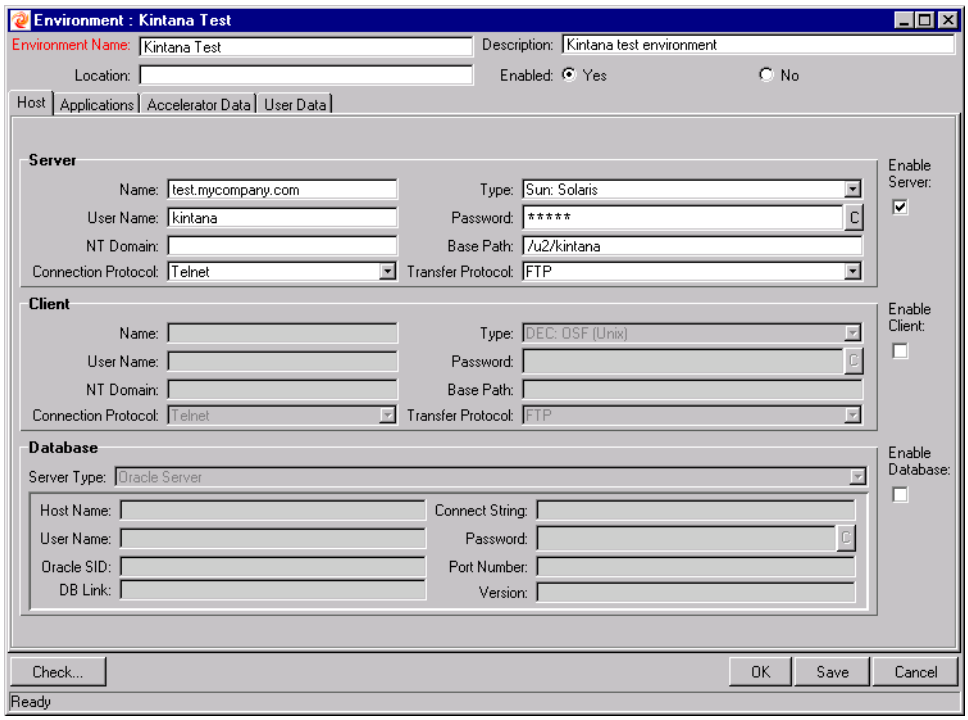



*Figure 2-3 Sample Environment Setup for Source Instance*

# Migrators and Ownership

Different groups of Kintana users have ownership and control over Kintana entities that are specific for their own 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.

Kintana 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 Kintana 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 hold true for import, but not for export.

# Supported Entities

The following entities can be migrated using Kintana 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 Kintana Migrator architecture, setup, and use. Examples illustrating particular aspects of each are also provided, where applicable.

The following topics are discussed:



## Migrator Architecture

A Kintana Migrator Object Type works by extracting configuration data for a particular entity (see *"Supported Entities"* on page 13 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 Kintana 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 Kintana instance anytime in the future.

# Migrator Processes - Best Practices

Migrating configuration entities from one Kintana 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 Kintana entities:



## Running a Migration

Using a Migrator, Kintana configuration information is moved with standard Kintana Deliver 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 you 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.


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.

Because of this matching restriction, you can be sure that, after the migration, each open Request will still be at the same process step 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 you from removing outgoing transitions from Workflow Steps. It is therefore important to avoid "stranding" open Requests at a Workflow Step that you are trying to deprecate. When deprecating a process step, you will want to remove incoming transitions, but you will 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. Kintana 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, you may prefer 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 in this manner:

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 the this as a new Workflow.

3. Following the migration, you may also want 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.

# Migrator Setup

Kintana Migrators are processed as Kintana Deliver 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, Kintana configuration information is moved with standard Kintana Deliver methods: processing a Package through a Workflow.

In order to use a Migrator Object Type, some configuration is necessary beforehand. To use the Kintana Migrators to migrate Kintana configuration information between instances:

1. Configure the appropriate Environments to represent Kintana instances involved in the migration.

2. Define a Kintana Deliver Workflow to model the migration process you want. If desired, you can 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 give more detailed information on each step in the migration process:

- *Configuring Environments*
- *Defining the Workflow*

## Configuring Environments

The Kintana Migrators are used to transfer configuration information between different Kintana installations.

In order to configure an Environment representing a Kintana installation, it is necessary to perform the following actions:

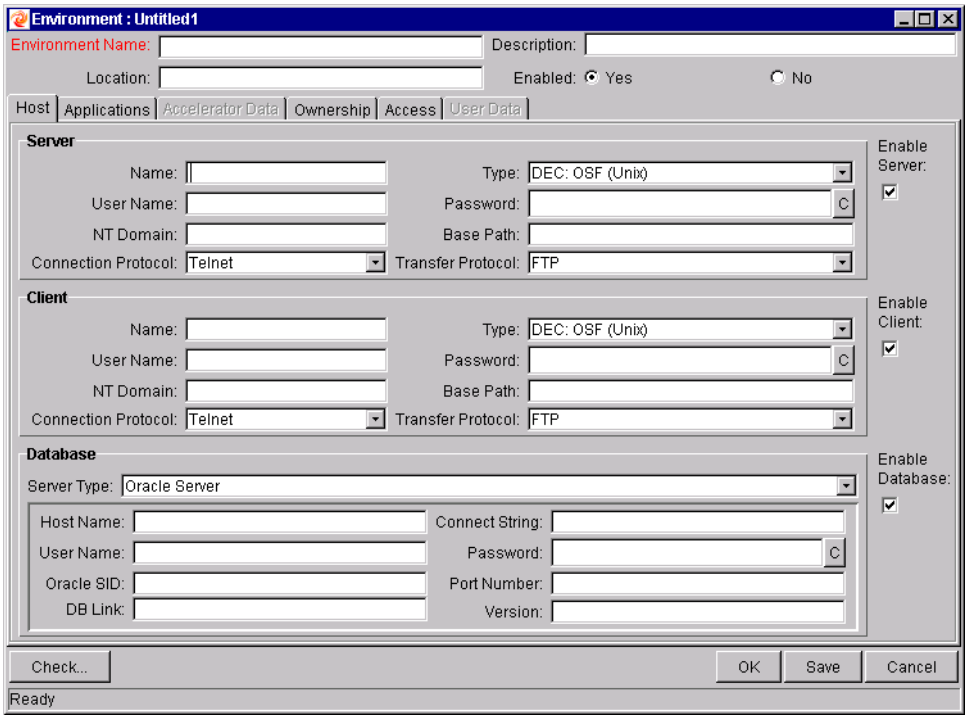
- *Configuring the Environment for the Source Instance*
- *Configuring the KINTANA_SERVER Environment for the Destination Instance*
- *Checking the server.conf File*

### *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 Kintana User on this server. |
| SERVER PASSWORD | Password for the Kintana User on this server. |
| NT DOMAIN | Required only for Windows servers. |
| SERVER BASE PATH | The installation directory for Kintana. |

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

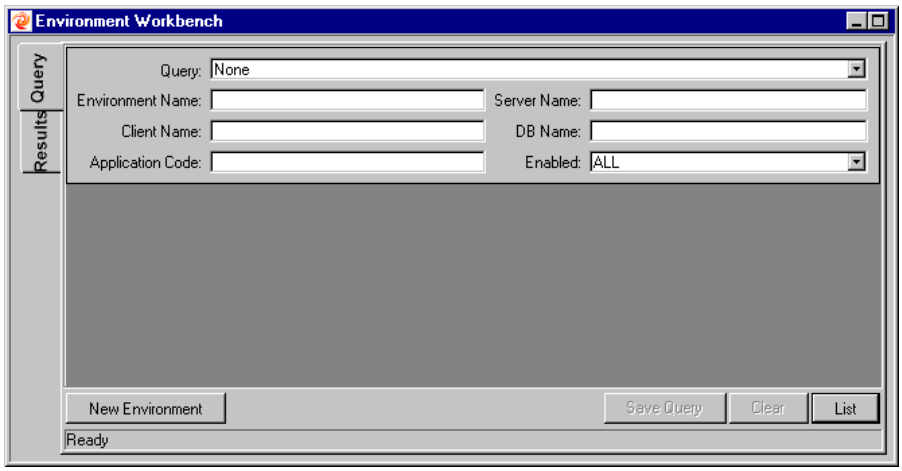
4. Click **SAVE** to save changes and continue to work with this Environment. Click **OK** to save the Environment and close the window.

## *Configuring the KINTANA_SERVER Environment for the Destination Instance*

Kintana installation automatically generates an Environment called KINTANA_SERVER, which represents the server hosting the Kintana

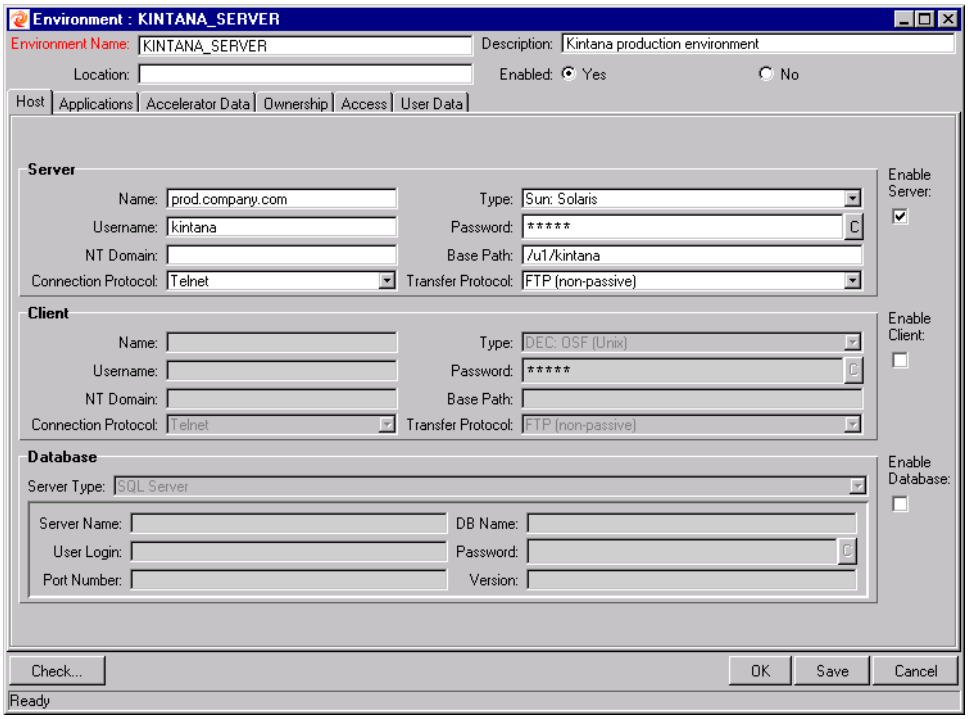
application server. This Environment, like all others, can be maintained using the Environment Workbench.

1. Select the **ENVIRONMENTS** screen group and click the **ENVIRONMENTS** screen. The ENVIRONMENT WORKBENCH opens.




2. Query for the KINTANA_SERVER Environment or click **LIST** to display all of the system Environments.

3. 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 Kintana. |

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

5. Click **SAVE** to save changes and continue to work with this Environment. Click **OK** to save the Environment and close the window.

## *Checking the server.conf File*

The Kintana Migrators depend upon a parameter in the Kintana configuration file. This parameter is SERVER_ENV_NAME. The value for this parameter should be the name of an environment in the Kintana system that describes the

host server running that Kintana 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. Please ensure this parameter is correctly configured.

```
SERVER_ENV_NAME=KINTANA_SERVER
```

As a Kintana installation automatically generates the environment KINTANA_SERVER, it is also captured as the default SERVER_ENV_NAME parameter in the server.conf file.


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 Kintana environment name must be synchronized.

## Defining the Workflow

To use the Kintana Migrator Object Types, it is necessary to define a Kintana Deliver 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.

A sample Kintana Migrator Workflow is shown in *Figure 3-1*.

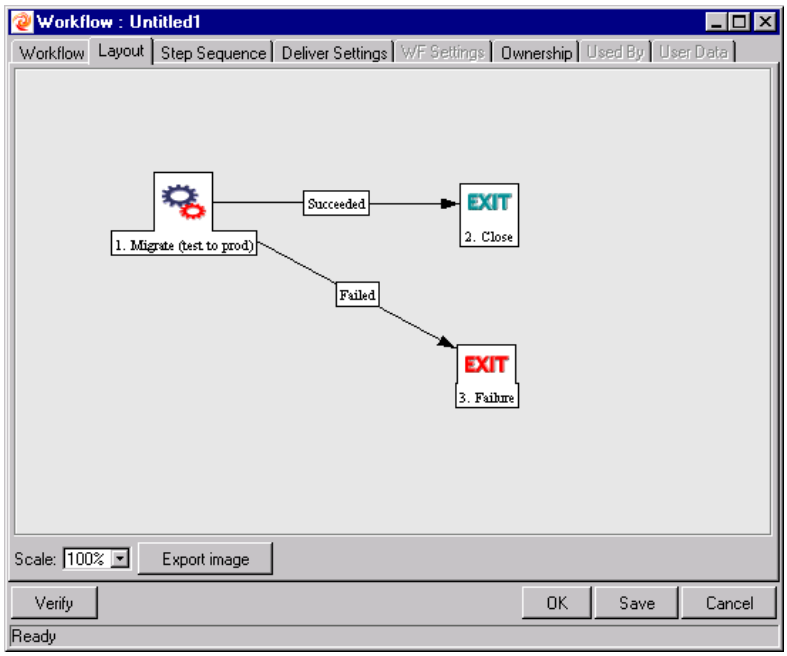



*Figure 3-1 Sample Kintana Deliver Migrator Workflow*

The first Workflow Step in this example is the necessary Execution step that actually migrates Kintana configuration data between a source and a destination Environment. See *Figure 3-2* below for the sample Workflow Step's configuration.

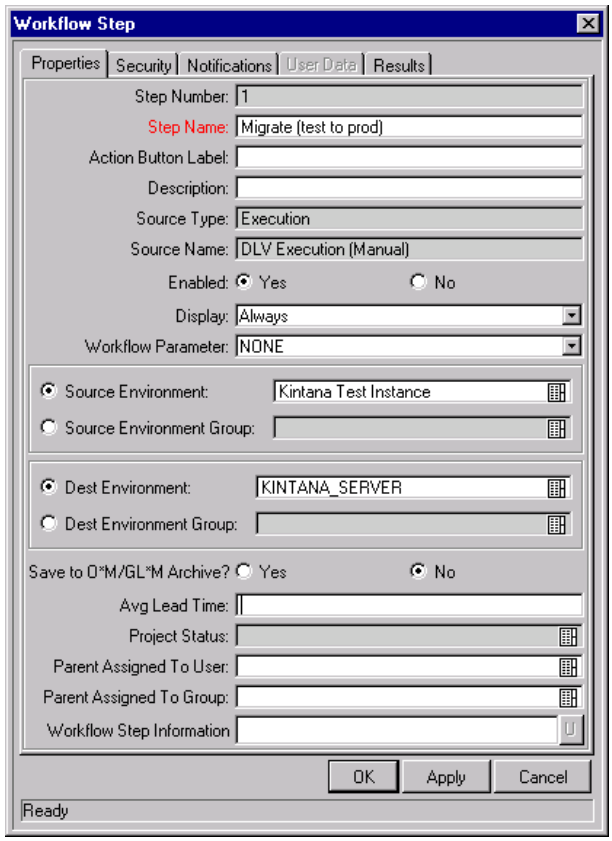



*Figure 3-2 Sample Kintana Deliver Migrator Workflow Step*

Note

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 Kintana, 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 Kintana Workflows, see *"Configuring a Deployment System in Kintana"*.

# Using Migrators

The actual process of migrating a configuration entity from one Kintana instance to another is handled using standard Deliver 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*

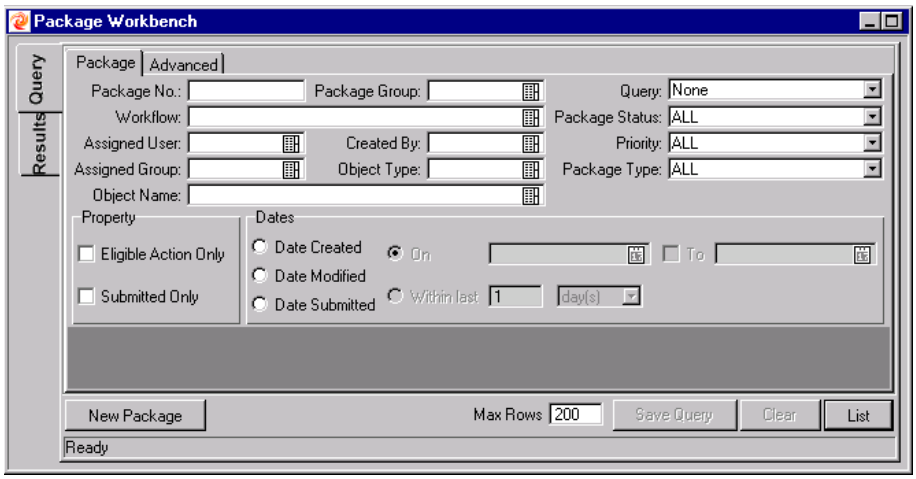
## Assumptions

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

## Building the Migrator Package

The actual Kintana configuration information to be migrated is gathered in a Kintana Deliver Package. To specify Kintana information and associated behaviors to be migrated:

1. Click the **DELIVER** screen group and click the **PACKAGES** screen. The PACKAGE WORKBENCH opens.




2. Click **NEW PACKAGE**.

3. Select your Migration Workflow from the WORKFLOW auto-complete list.

4. Add Package Lines using the Object Migrator Object Types. Specify any desired configurations and behaviors.

5. Click **SAVE** to save the Package without closing or submitting it for release. Click **OK** to save the Package and close the window without submitting it for release.

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

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


You can migrate a Kintana Object Type from your Testing instance to your Production instance of Kintana. When you migrate 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, and Special Commands referenced by Commands or Validations.

There are certain fields and behaviors that are common to all Kintana Migrators, and others that are specific to particular Migrators. The following sections discuss in detail common Kintana 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 Kintana Migrators.

- *Migrator Action*
- *Preview Import?*
- *Target Entity*

- *Content Bundle Fields*
- *Import Behavior Fields*
- *Source Password*
- *Destination Password*
- *Internationalization*

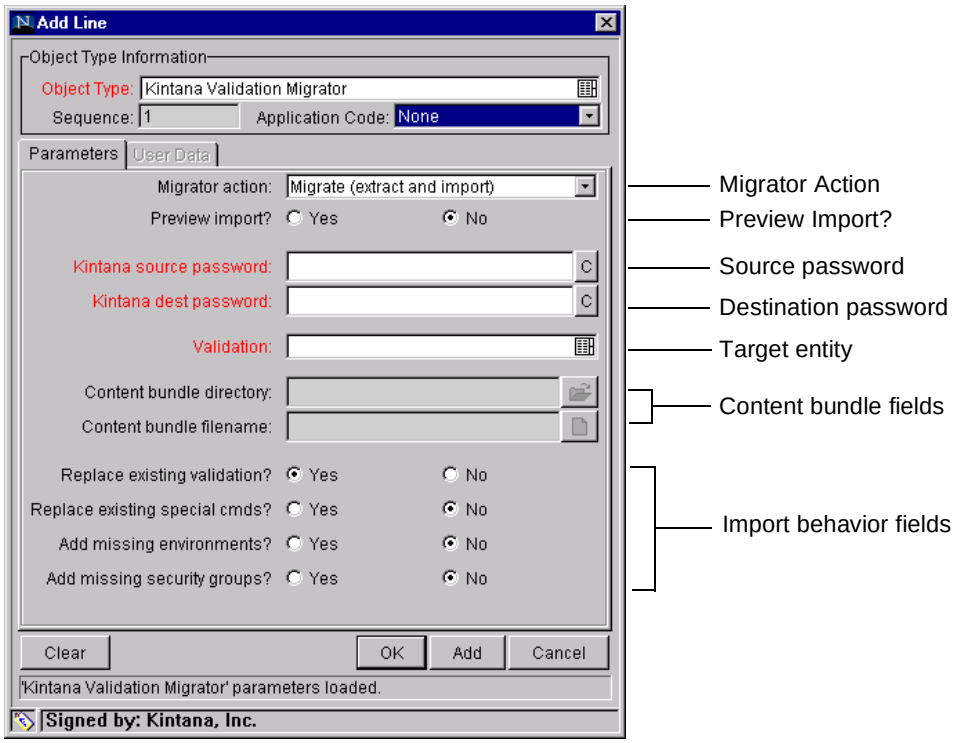



*Figure 4-1 Kintana Migrator Parameters*

## Migrator Action

The Migrator Action field governs the migration of its particular Package Line by controlling whether other fields are enabled or required. It 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 Kintana entity to be migrated or extracted.


Since the validation values are determined by contacting the source Kintana 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 Kintana

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 Kintana application server.

## Import Behavior Fields

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

- REPLACE EXISTING (ENTITY)? — If the entity to be migrated already exists in the target Kintana 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 Kintana instance, the user can decide whether to overwrite them. This field’s 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 field’s default is **NO**.

There may be other import behavior fields, depending on the specific Migrator Object Type. Any additional import behavior fields are detailed in the *Migrator Object Types* sections below.

## Source Password

When the Kintana Migrator contacts the source Kintana application server, the current user’s name and password is used to log on to the other instance. Any Kintana system administrator performing a migration should already be linked to the proper security group containing the access grant allowing access to the source Kintana instance, SYS ADMIN: MIGRATE KINTANA 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.

## Destination Password

When the Kintana Migrator contacts the destination Kintana application server, the current user's name and password is used to log on to the other instance. Any Kintana system administrator performing a migration should already be linked to the proper security group containing the access grant allowing access to the destination Kintana instance, SYS ADMIN: MIGRATE KINTANA 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.

## Internationalization

This field may not be displayed on Migrator Object Types, but it 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 Kintana 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:

- *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 29 for more detailed information).

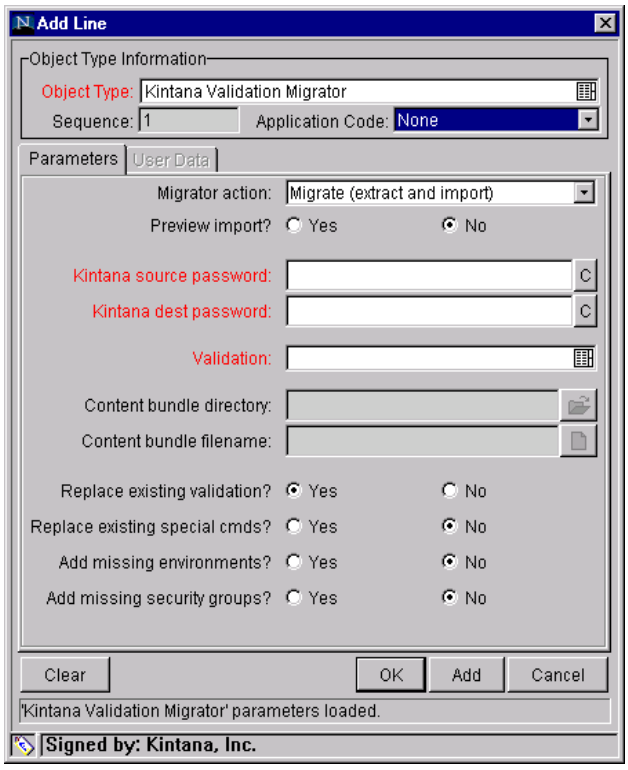



*Figure 4-2 Validation Migrator*

### *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 Kintana Special Commands that already exist in the target Kintana 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, and so forth. This field's 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 Kintana 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 Accelerator-specific Environment tabs are not transferred.

  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. This field's default value is **NO**.

### *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 29 for more detailed information).

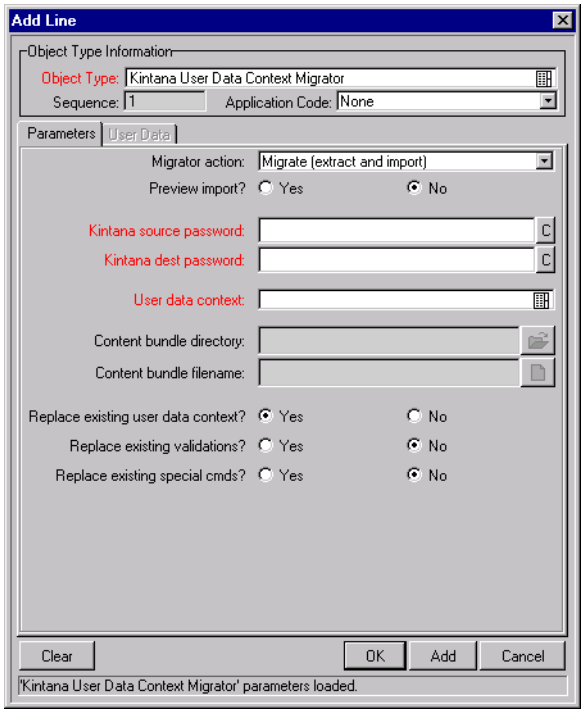



*Figure 4-3 User Data Context Migrator*

### *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 Kintana Special Commands that already exist in the target Kintana 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. This field's 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 29 for more detailed information), except for the 'Replace existing validations' field. This migrator does not have any additional import behavior fields.

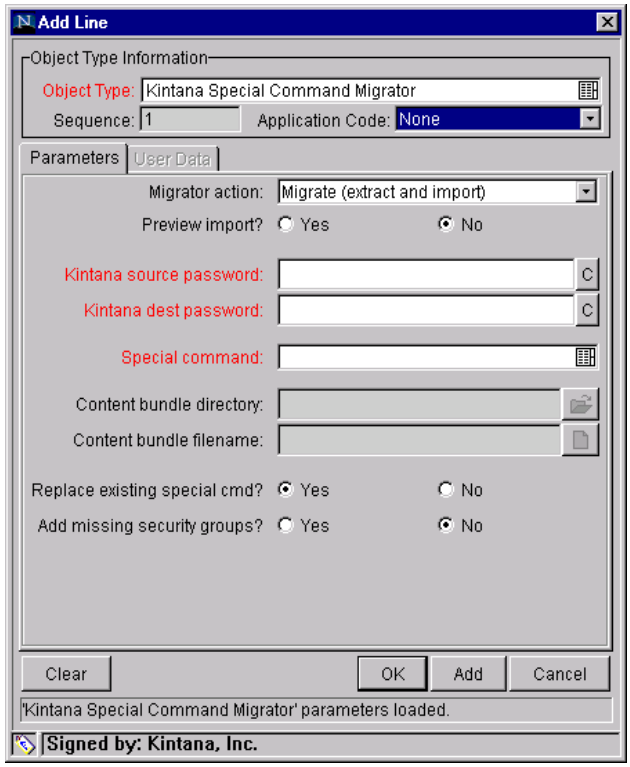



*Figure 4-4 Special Command Migrator*

## Workflow Migrator

The Workflow Migrator Object Type contains all standard Migrator Object Type fields (see *"Standard Migrator Fields and Behaviors"* on page 29 for more detailed information).

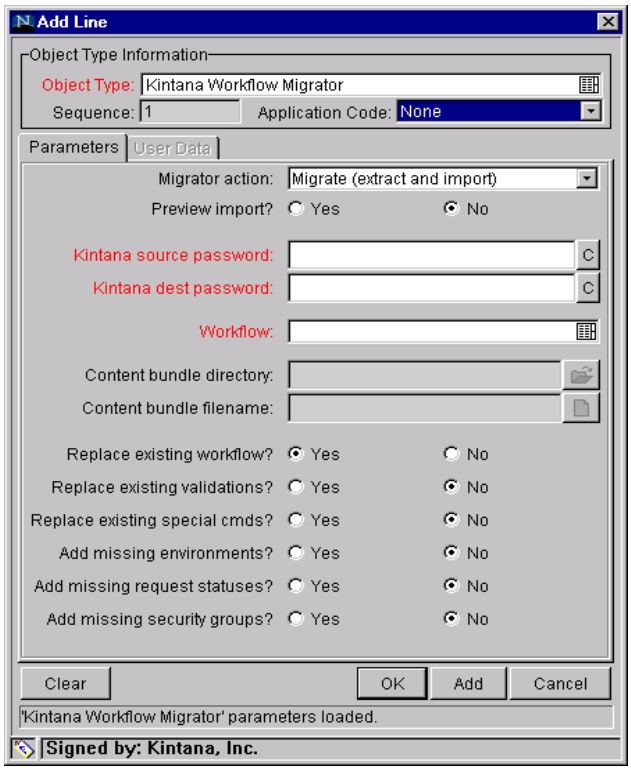



*Figure 4-5 Workflow Migrator*

## *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 Kintana Special Commands that already exist in the target Kintana 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. This field's 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 Kintana 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 Kintana instance, the user can decide whether to create them. However, only the Environment header information and User Data are transferred. Application codes and Accelerator-specific Environment tabs are not transferred.

  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. This field's default value is **NO**.

- ADD MISSING REQUEST STATUSES? -- If the Workflow to be migrated references Request statuses that do not exist in the target Kintana 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.


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 you are 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.


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.

Because of this matching restriction, you can be sure that, after the migration, each open Request will still be the same process step 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 you from removing outgoing transitions from Workflow Steps. It is therefore important to avoid “stranding” open Requests at a Workflow Step that you are trying to deprecate. When deprecating a process step, you will want to remove incoming transitions, but you will 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. Kintana 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, you may prefer 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 in this manner:

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 the this as a new Workflow.

3. Following the migration, you may also want 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 29 for more detailed information).

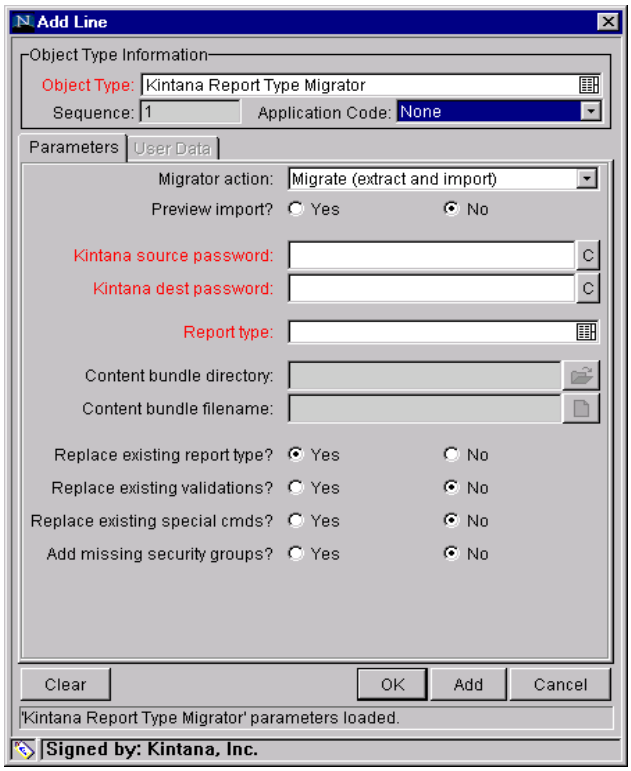



*Figure 4-6 Report Type Migrator*

## *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 Kintana Special Commands that already exist in the target Kintana 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. This field's 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


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 29 for more detailed information).

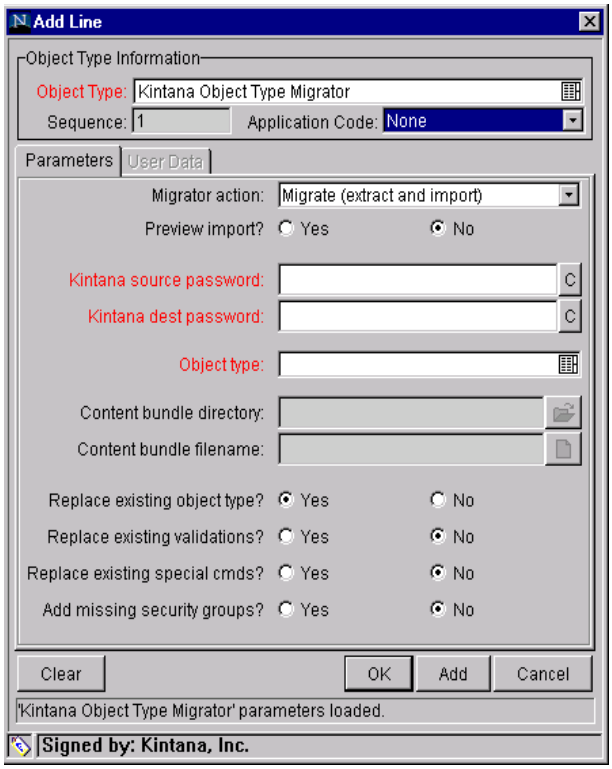



*Figure 4-7 Object Type Migrator*

## 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 Kintana Special Commands that already exist in the target Kintana 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. This field's 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 29 for more detailed information).

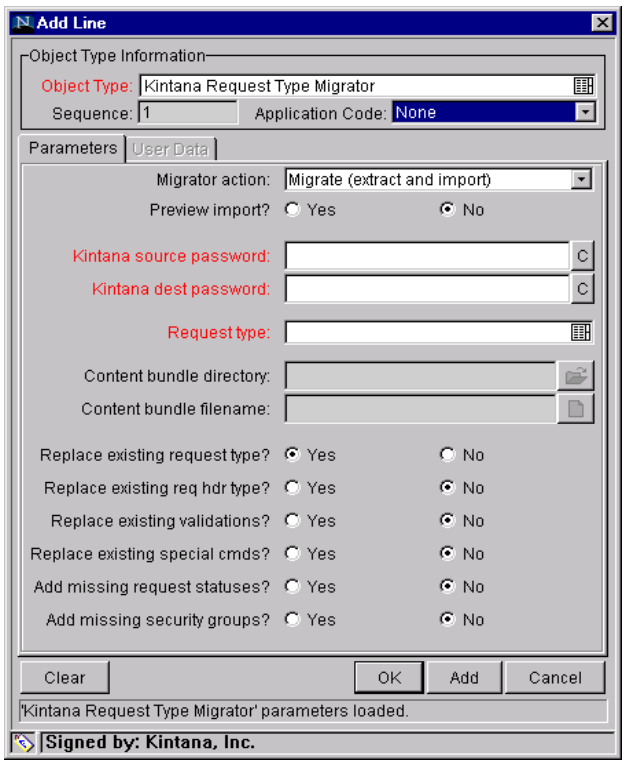



*Figure 4-8 Request Type Migrator*

## *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 Kintana 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 Kintana Special Commands that already exist in the target Kintana 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. This field's 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 Kintana 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.


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.


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 29 for more detailed information).

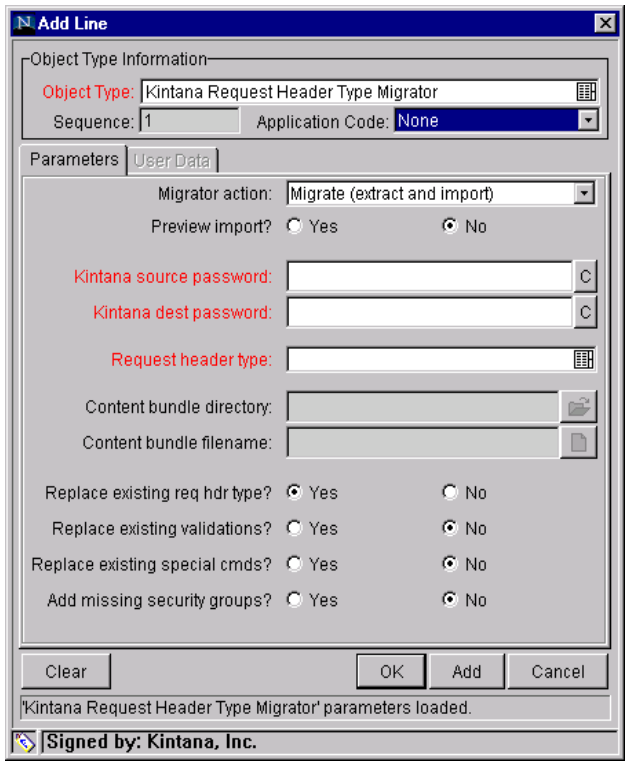



*Figure 4-9 Request Header Type Migrator*

## *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 Kintana Special Commands that already exist in the target Kintana 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. This field's 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 29 for more detailed information).

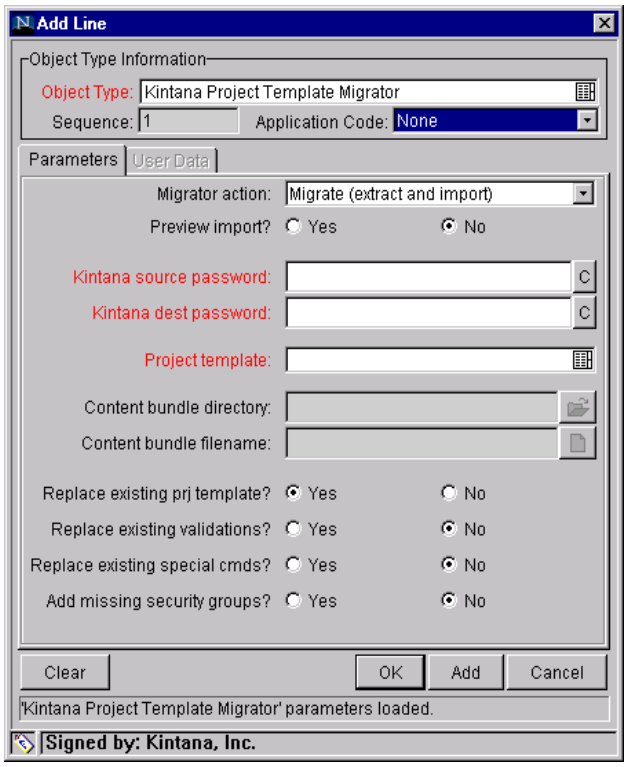



*Figure 4-10 Project Template Migrator*

## *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 Kintana Special Commands that already exist in the target Kintana instance, the user can decide whether to replace them. This includes both parent and children Special Commands. This field's 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 29 for more detailed information). When you migrate 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.

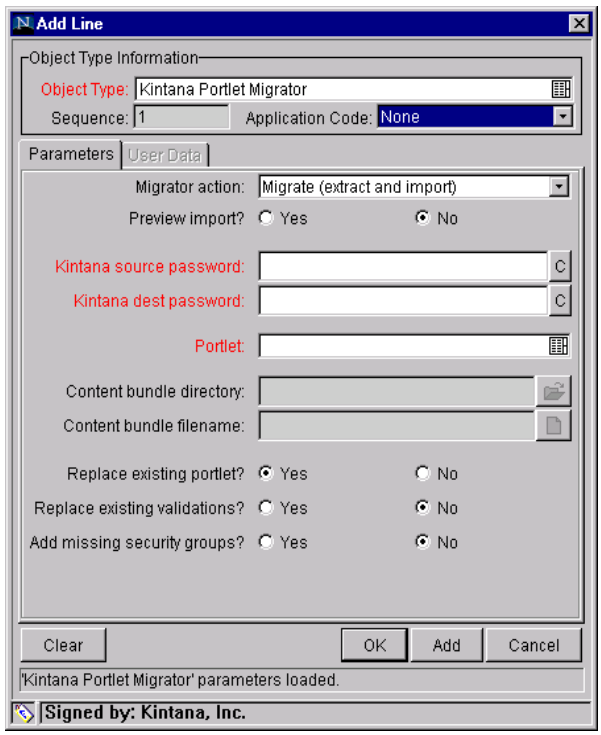



*Figure 4-11 Portlet Migrator*

# Appendix A
# Using Migrators to Archive

The CONTENT BUNDLE fields in Kintana Migrator Object Types can be used to archive configuration entities in a version-control system.

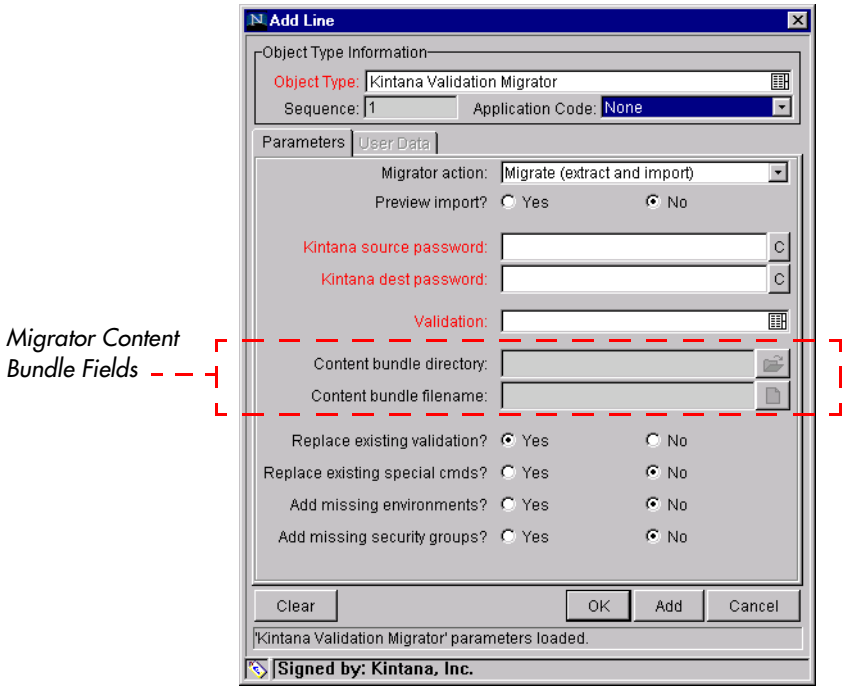



*Figure A-1 Migrator Object Type Content Bundle Fields*

To archive your configuration entities in your version-control system:

1. Specify a path and filename for your Migrator content bundle.
2. Run a migration with the MIGRATOR ACTION field set to **EXTRACT ONLY**.
3. Check the ZIP file into your version-control system.

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


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

If, later on, you wish to use this bundle with Kintana version 5.0, the bundle should be imported into a Kintana 4.6 instance, and the instance upgraded to Kintana 5.0 after the import.

# Index

## A



## B



## C



## D



## E



## I



## M

## O



## P



## R



## S



## T



## U



## V



## W