opentext™

OpenText™ Project and Portfolio Management (PPM)

Software version: All versions

Deployment Management Extension for Oracle Technology Guide

Go to Help Center online

https://admhelp.microfocus.com/ppm/

Document release date: October 2023

Send Us Feedback

Let us know how we can improve your experience with the Deployment Management Extension for Oracle Technology Guide.
Send your email to: admdocteam@opentext.com

Legal Notices

Contents

Getting Started

Introduction to the Extension for Oracle Technology

Deployment Management Extension for Oracle Technology (usually referred to hereafter as "the Extension") helps organizations to automate deployment management in application environments built using Oracle® tools such as SQL*Plus, PL/SQL, Oracle Forms, and Oracle Reports.

The Extension enhances the functionality of the Deployment Management application in OpenText ™ Project and Portfolio Management Center (PPM) by providing predefined object types and commands specifically related to development using Oracle tools.

After PPM (the "base" software) has been installed at or upgraded to 9.50, you can install the Extension or upgrade it to version 9.50 on the same system.

"Installing or Upgrading the Extension" on page 8 describes system requirements for installing the Extension or upgrading it to version 9.50, and the impacts of upgrading the Extension to version 9.50.

What's New and What's Changed in Version 9.50

Extension version 9.50 makes no functional changes compared to version 9.40. Upgrading to version 9.50 does not affect existing Extension functionality.

After installing PPM at version 9.50, you can install Extension version 9.50 for the first time, or you can upgrade the Extension from version 9.40.

Note: PPM supports having multiple languages on the same instance, including translations of specific PPM entities and interfaces. No translations are provided for any Extension entities or interfaces with the Extension or any language pack.

However, like any customer-defined entity, Extension entities such as request types can be translated by using the `kExportAttributes.sh` and `kImportAttributes.sh` scripts as part of a translation process. For more information, see the *Multilingual User Interface Guide.*

"Installing or Upgrading the Extension" on page 8 describes general impacts of upgrading the Extension from version 9.40 to version 9.50.

About This Document

This document is intended for the following audiences:

- Oracle Applications developers building applications using Oracle tools such as SQL*Plus, PL/SQL, Oracle Forms, and Oracle Reports
- Database or application administrators responsible for installing or maintaining the toolset, or controlling changes in one or more application environments built on the Oracle toolset
- Database or application administrators responsible for maintaining access and security for, or supporting use of, Deployment Management Extension for Oracle Technology

This guide provides information about installing or upgrading of Deployment Management Extension for Oracle Technology, and this guide provides conceptual, procedural, and reference information about the product.

This guide is organized as follows:

- "Getting Started" on the previous page provides an introduction to the Extension, the features and changes introduced in version 9.50, information about the intended audiences for this document, related information, and prerequisite knowledge.
- "Installing or Upgrading the Extension" on page 8 provides overview and detailed information about installing or upgrading the Extension, including information

about the impacts of upgrading the Extension.

- "Extension Object Types" on page 16 provides information about the Extension's object types.

Related Information

The following documents also include information useful in managing Deployment Management Extension for Oracle Technology:

- *Installation and Administration Guide*
- *System Requirements and Compatibility Matrix*
- *Security Model Guide and Reference*
- *What's New and What's Changed*
- *Deployment Management Configuration Guide*
- *Deployment Management User Guide*

The following additional Extension and Migrator documentation for Oracle environments might be of particular interest:

- *Object Migrator Guide*
- *GL Migrator Guide*
- *Deployment Management Extension for Oracle E-Business Suite Guide*

Prerequisite Knowledge and Experience

To install, upgrade, configure, maintain, or use the Extension, you need to understand the following:

- Deployment management
- Environments
- Software deployment
- Packages
- PPM Dashboard pages and portlets
- PPM Workbench

- Object types
- Workflows and workflow steps
- Tokens
- PPM entities installed by the Extension

In addition, you must have practical experience in the following:

- Installing, upgrading, configuring, and using PPM, if you are responsible for configuring the Extension
- Installing, configuring, and using Oracle tools

Installing or Upgrading the Extension

Overview of Installation and Upgrade

This section describes system requirements that must be met before installing or upgrading the Extension, and this section describes upgrade impacts.

System Requirements

Before you install the Extension or upgrade it to version 9.50, PPM version 9.50 must be installed. PPM and the Extension are installed on the same system and have the same system requirements.

For information about installing PPM version 9.50, see the *Installation and Administration Guide.* For information about upgrading PPM to version 9.50, see the *Upgrade Guide*.

General Upgrade Impacts and Guidelines

Each new version or service pack of the Extension provides the following types of entities, which the Extension upgrade process installs or does not install in the PPM instance, as described:

- **Reference entities.** The names of reference entities start with (REFERENCE).

  You cannot edit reference entities. However, you can copy and rename them and then edit the copies as non-reference entities, as appropriate for your environment.

  The upgrade process deletes all of the Extension's existing (previously installed) reference entities, such as its object types, and then the process installs a new set of reference entities for the new version. The new set of reference entities can be identical to the previous set or, to support changes in functionality, the new set can have new, deleted, renamed, or modified reference entities.

- **Non-reference entities.** The non-reference entities and corresponding reference entities that the version or service pack provides are identical, except that the names of the reference entities start with (REFERENCE).

  You can edit (or copy and edit) non-reference entities, as appropriate for your environment. The currently installed version of the Extension might include non-reference entities that have been edited (customized).

  To preserve all existing customizations, the upgrade process compares the names of the non-reference entities delivered with the new version to the names of existing non-reference entities in the instance, and if the process finds an existing non-reference entity with the same name, the process does *not* overwrite or modify that entity in the instance. If the new version introduces reference entities with new names along with non-reference copies of those entities, then the upgrade process adds both the reference and non-reference copies.

In general, if an upgrade changes an Extension's reference entities, you must evaluate how those changes should affect the associated, previously customized non-reference entities. After the upgrade is performed, you can revise those non-reference entities or create new ones as necessary.

Specific Upgrade Impacts for Version 9.50

Upgrading to version 9.50 does not affect existing Extension functionality.

Preparing for Installation or Upgrade

Prepare for installation or upgrade of the Extension as described in the following sections.

Note: During installation or upgrade, the PPM Server must run in restricted mode.

General Preparations for Installation or Upgrade

To prepare for installation or upgrade of the Extension:

1. Obtain the Extension software.
2. Collect the following information, which you will need to supply during the installation procedure:
   - The logon username and password for the Extension's server (the same server on which PPM is or will be installed). The username (typically "admin") must belong to a security group that has the following access grants:
     - Sys Admin: Migrate PPM Objects
     - Sys Admin: Server Administrator
   - The database password for the server's schema.
3. Log on to the PPM Server.
4. Verify that the system requirements have been met. See "System Requirements" on page 8.
5. Save the Extension installation file (`ppm-950-OracleTech.jar`) to one of the following directories:
   - <*PPM_Home*> (the recommended location).

     <*PPM_Home*> represents the path where your PPM instance is installed. For example: `xyzserver/E/PPMServer`.
   - A subdirectory of <*PPM_Home*>. If a subdirectory of <*PPM_Home*> is specified in the `ITG_DEPLOYMENT_HOME` environment variable, the installation script finds and uses the installation file in that subdirectory. If you want to save the installation file to a subdirectory of <*PPM_Home*>, make sure the value of the `ITG_DEPLOYMENT_HOME` environment variable is <*PPM_Home*> followed by that subdirectory, for example, <*PPM_Home*>/`Extension`.
   - Any directory (with, optionally, any subdirectories) you choose, for example, `dirA/sub1/sub2`.

Performing Backup and Restarting the PPM Server in Restricted Mode

The steps in this section are recommended but not required.

Note: For more information about the steps in this procedure, see the *Installation and Administration Guide.*

For a new installation or an upgrade, do the following:

1. Back up the database and file system for the PPM Server.
2. To stop the PPM Server and restart it in restricted mode:
   a. Stop the PPM Server.
   b. Run the following script:

      ```
      sh ./setServerMode.sh RESTRICTED
      ```

   c. Start the PPM Server.

Installation Procedure

Perform the procedures in the following sections to install the Extension.

Run the Installation Script and Check the Logs

To run the installation script to install the Extension, and to check the logs:

1. Be sure you have completed all the steps in "Preparing for Installation or Upgrade" on page 9.

   In particular, be sure the PPM Server is running in restricted mode. See "Performing Backup and Restarting the PPM Server in Restricted Mode" above.
2. On the PPM Server, navigate to the `bin` subdirectory of the <*PPM_Home*> directory (or other directory as described in step 5).
3. Start the installation or upgrade. In step 5, if you saved the installation file to <*PPM_Home*>, which is the recommended directory, or to a subdirectory of

<*PPM_Home*>, run the following script:

```
sh ./kDeploy.sh -i OracleTech
```

However, if you saved the installation file to a different directory, see the example in step 5 and specify that directory in the script command, as in the following example:

```
sh ./kDeploy.sh -i OracleTech -D dirA/sub1/sub2
```

4. Follow the script's on-screen prompts to complete the installation. Prompts can include the database password for the schema and the logon name and password for the server.

   Files are installed in various subdirectories under <*PPM_Home*>. Data is also placed in the database. When the installation script is complete, the following message appears:

   Deployment OracleTech has been successfully installed.

5. Use a Web browser to check the installation summary report, which is located at:

   <*PPM_Home*>`/logs/deploy/920/OracleTech/`<*Log_x*>`/installLog.html`

   where <*Log_x*> is initially a random number generated by `kDeploy.sh` during installation. The number increments by one each time the installation script is run, so the installation summary report for the most recent run is in the directory with the highest number.

   The installation summary report lists all the entities that are installed as part of the Extension installation process. Each entity that was installed correctly is marked as Complete. If there is an error for a particular entity, the report contains a direct link to another log file (HTML page) with additional information.

   If necessary, correct any errors and repeat the installation process.

   Installation of the Extension generates the logs that are described in "Logs Generated During Installation" on the next page, depending on the installation options.

Logs Generated During Installation

Depending on the installation options that were chosen, the logs listed and described in "Table 2-1. Logs generated during installation" below can be generated during installation and saved in the following directory:

<*PPM_Home*>`/logs/deploy/920/OracleTech`

The log number (<*#####*>) shown in "Table 2-1. Logs generated during installation" below is a random number (generated by `kDeploy.sh`) that makes each log file name unique.

Table 2-1. Logs generated during installation

File Name	Description
ddlDriver.<*#####*>.log	Contains information about data model changes made during installation
jarxvf.<*#####*>.log	Contains information from the procedure that unpacks the `.jar` file
packageDriver.<*#####*>.log	Contains information about the installation of database code; for example, reports
postXMLDriver.<*#####*>.log	Contains information about the application of SQL scripts required after the installation of OOTB data
preXMLDriver.<*#####*>.log	Contains information about the application of SQL scripts required before the installation of OOTB data, such as the definition for Deployment Management Extension for Oracle Technology

Verify the Installation

We strongly recommend that you verify correct installation. To verify that Extension version 9.50 for Oracle Technology is listed among the installed Extensions, navigate to the <*PPM_Home*>`/bin` directory and run the following script:

```
sh ./kDeploy.sh -l
```

where the last character in the command is the lowercase letter "l."

The name OracleTech should appear in the list of installed Extensions.

For example, if both Deployment Management Extension for Oracle Technology and Deployment Management Extension for Oracle E-Business Suite are now installed at version 9.50, the following table entries are displayed:

Deployment	Version	Deployed	Description
OracleTech	950	<*date and time*>	OracleTech Extension
OracleApps	950	<*date and time*>	Oracle Apps Extension

Restart the PPM Server in Normal Mode

Note: For more information about the steps in this procedure, see the *Installation and Administration Guide.*

After you have completed all installation or upgrade procedures, if you previously restarted the PPM Server in restricted mode, to stop and restart it in normal mode:

1. Stop the PPM Server.
2. Run the following script:

   ```
   sh ./setServerMode.sh NORMAL
   ```

3. Start the PPM Server.

Post-Installation Procedures

After you have finished installing Deployment Management Extension for Oracle Technology, you must do the following:

- Review the object types.
- Define environments for each application environment that will be a source or destination of migration, for example:
  - Host connection information
  - Database connection information
- If required, customize the logic in the various installed entities.

Note: After the Extension has been running successfully for a substantial period of time, you can optionally delete all of the installation files. However, we recommend that you retain (or copy) the log files.

Extension Object Types

Overview of Object Types

This section provides reference information about the Oracle Technology-specific object types provided in the Extension. These object types are listed and defined in "Table 3-1. Object types included in the Extension, continued" on page 18.

Migration and compilation of object types are driven by commands included within the object types. For more information about commands in the PPM environment, see the *Commands, Tokens, and Validations Guide and Reference.* For more information about using object types in packages, see the *Deployment Management User Guide.*

You can view or modify an object type as follows:

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

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

   The Object Type Workbench opens.
4. (Optional) Select **Oracle Technology** in the **Extension** field on the **Query** tab.
5. In the Object Type Workbench, click **List.**
6. Select the object type of interest and click **Open.**
7. Edit the object type. (On the **Fields** tab, the list of fields in the **Prompts** column is alphabetized.)

Subsequent figures in this section show windows you can use to revise object types in conjunction with adding package lines. You can access these windows as follows:

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

The PPM Workbench opens.

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

   The Package Workbench opens.

4. Add a new or open an existing package, as necessary.
5. Select a workflow.
6. Add a line.
7. Select the object type of interest.

Reference Object Types

Reference object types cannot be edited, but you can copy and rename them and edit the copies to meet your needs. You can also use existing non-reference object types as is or configure them further to meet your needs.

List of Object Types

"Table 3-1. Object types included in the Extension, continued" on the next page lists and defines the object types included in the Extension. Each is described in subsequent sections.

Table 3-1. Object types included in the Extension

Object Type	Description
Forms 4.5	Moves Oracle Forms 4.5 files from one instance to another
Forms 6.0	Moves Oracle Forms 6.0 files from one instance to another
Forms 10G	Moves Oracle Forms 10G files from one instance to another
Reports 2.0	Moves Oracle Reports 2.0 files from one instance to another
Reports 2.5	Moves Oracle Reports 2.5 files from one instance to another
Reports 6.0	Moves Oracle Reports 6.0 files from one instance to another
Reports 10G	Moves Oracle Reports 10G files from one instance to another

Table 3-1. Object types included in the Extension, continued

Object Type	Description
SQL Loader File	Migrates the SQL Loader Control file from one instance to another and optionally runs SQL Loader
SQL Loader80 File	Migrates the SQL Loader control files from one instance to another and optionally runs SQL Loader against an Oracle 8 instance
SQL Script	Moves SQL and PL/SQL scripts from one instance to another and optionally executes the scripts at the destination

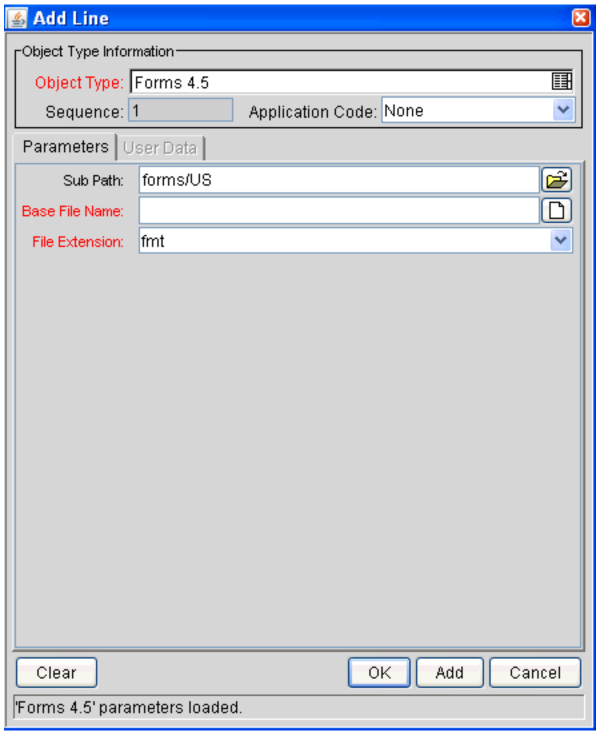
Forms 4.5 Object Type

The Forms 4.5 object type moves Oracle Forms 4.5 files from one instance to another. If the extension of the file being migrated is `fmt`, then the file is parsed at the source to generate a file with an `fmb` extension. Based on the `fmb` file, a file with an `fmx` extension is generated. The object type connects to the Oracle Account defined for the application (or on the **Host** tab, if the application value is null) in the destination environment to generate the `fmx` and `fmb` files. The `fmx` file is then moved to the destination instance.

You can modify this object type to include steps to interact with your version control system.

"Figure 3-1. Forms 4.5 object type sample data" below shows the default screen when adding a package line that uses the Forms 4.5 object type. "Table 3-2. Forms 4.5 object type field descriptions" on the next page provides field descriptions for the object type.

Figure 3-1. Forms 4.5 object type sample data



Object Type: Forms 4.5
Sequence: 1
Application Code: None
Sub Path: forms/US
Base File Name:
File Extension: fmt

'Forms 4.5' parameters loaded.

Table 3-2. Forms 4.5 object type field descriptions

Field Name (*Required)	Description
Sub Path	Subpath (relative to base path) to the Forms 4.5 files
*Base File Name	Name of the file to be migrated
*File Extension	File extension of the file to be migrated—`fmt`, `fmb`, or `fmx`
File Location (Hidden by default)	**Client** or **Server**
File Type (Hidden by default)	**ASCII** or **binary**

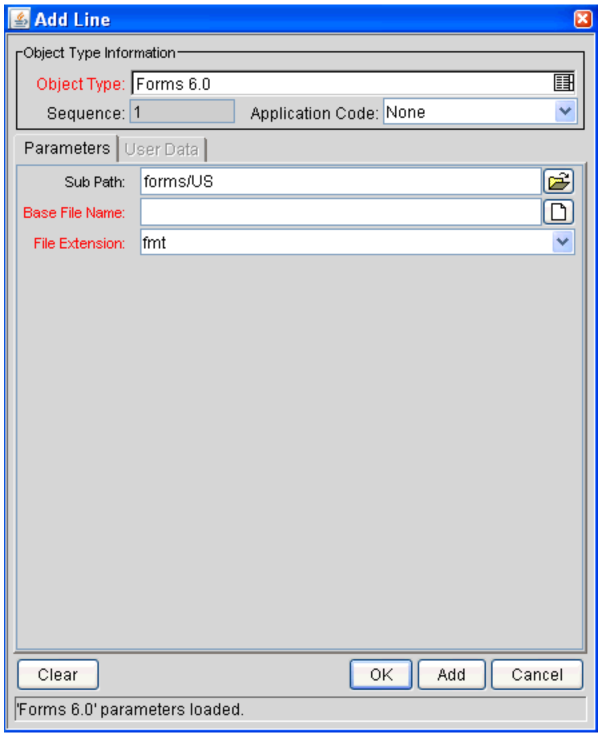
Forms 6.0 Object Type

The Forms 6.0 object type moves Oracle Forms 6.0 files from one instance to another. If the extension of the file being migrated is `fmt`, then the file is parsed at the source to generate a file with an `fmb` extension. Based on the `fmb` file, a file with an `fmx` extension is generated. The object type connects to the Oracle Account defined for the application (or on the **Host** tab, if the application value is null) in the destination environment to generate the `fmx` and `fmb` files. The `fmx` file is then moved to the destination instance.

You can modify this object type to include steps to interact with your version control system.

"Figure 3-2. Forms 6.0 object type sample data" below shows the default screen when adding a package line that uses the Forms 6.0 object type. "Table 3-3. Forms 6.0 object type field descriptions, continued" on the next page provides field descriptions for the object type.

Figure 3-2. Forms 6.0 object type sample data



Object Type: Forms 6.0
Sequence: 1
Application Code: None
Sub Path: forms/US
Base File Name:
File Extension: fmt
'Forms 6.0' parameters loaded.

Table 3-3. Forms 6.0 object type field descriptions

Field Name (*Required)	Description
Sub Path	Subpath (relative to base path) to the Forms 6.0 files
*Base File Name	Name of the file to be migrated

Table 3-3. Forms 6.0 object type field descriptions, continued

Field Name (*Required)	Description
*File Extension	File extension of the file to be migrated—`fmt`, `fmb`, or `fmx`
File Location (Hidden by default)	**Client** or **Server**
File Type (Hidden by default)	**ASCII** or **binary**

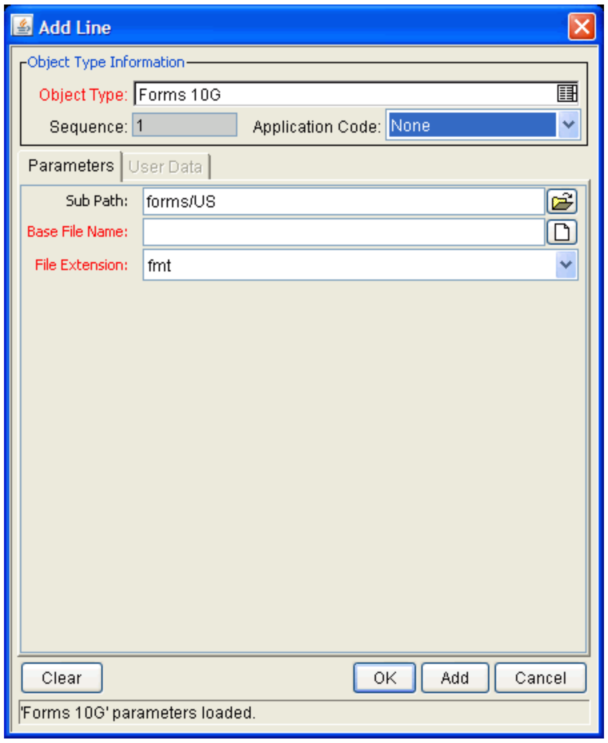
Forms 10G Object Type

The Forms 10G object type moves Oracle Forms 10G files from one instance to another. If the extension of the file being migrated is `fmt`, then the file is parsed at the source to generate a file with an `fmb` extension. Based on the `fmb` file, a file with an `fmx` extension is generated. The object type connects to the Oracle Account defined for the application (or on the **Host** tab, if the application value is null) in the destination environment to generate the `fmx` and `fmb` files. The `fmx` file is then moved to the destination instance.

You can modify this object type to include steps to interact with your version control system.

"Figure 3-3. Forms 10G object type sample data" below shows the default screen when adding a package line that uses the Forms 10G object type. "Table 3-4. Forms 10G object type field descriptions" on the next page provides field descriptions for the object type.

Figure 3-3. Forms 10G object type sample data



Object Type: Forms 10G
Sequence: 1
Application Code: None
Sub Path: forms/US
Base File Name:
File Extension: fmt
'Forms 10G' parameters loaded.

Table 3-4. Forms 10G object type field descriptions

Field Name (*Required)	Description
Sub Path	Subpath (relative to base path) to the Forms 6.0 files
*Base File Name	Name of the file to be migrated
*File Extension	File extension of the file to be migrated—`fmt`, `fmb`, or `fmx`
File Location (Hidden by default)	**Client** or **Server**
File Type (Hidden by default)	**ASCII** or **binary**

Reports 2.0 Object Type

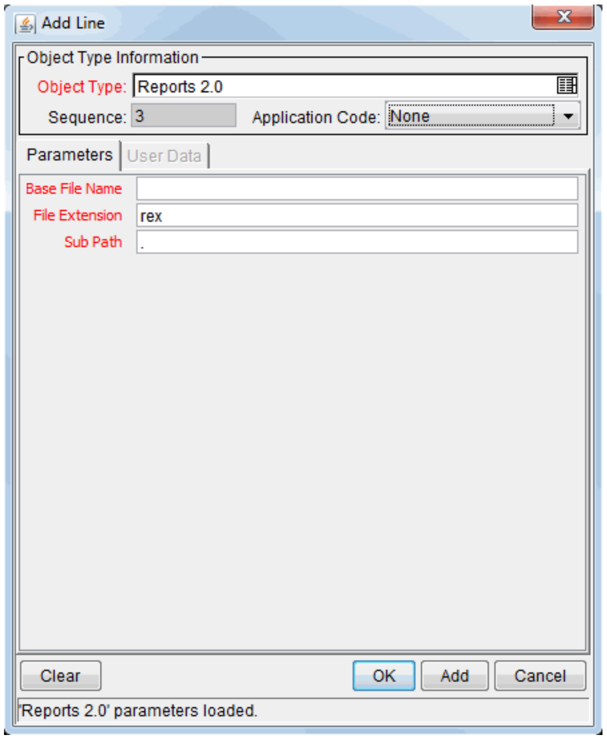
The Reports 2.0 object type moves Oracle Reports 2.0 files from one instance to another. If a `rex` file is being moved, then an `rdf` file is generated at the destination.

This migrates the report to the Oracle Account defined as the destination in the **Host** tab of the PPM Environment Workbench.

You can modify this object type to include steps to interact with your version control system.

"Figure 3-4. Reports 2.0 object type sample data" below shows the default screen when adding a package line that uses the Reports 2.0 object type. "Table 3-5. Reports 2.0 object type field descriptions" below provides field descriptions for the object type.

Figure 3-4. Reports 2.0 object type sample data



Object Type: Reports 2.0
Sequence: 3
Application Code: None
Base File Name
File Extension: rex
Sub Path: .
'Reports 2.0' parameters loaded.

Table 3-5. Reports 2.0 object type field descriptions

Field Name (*Required)	Description
*Base File Name	Name of the file to be migrated
*File Extension	File extension of the file to be migrated—`rex` or `rdf`
*Sub Path	Subpath (relative to base path) to the Reports 2.0 files

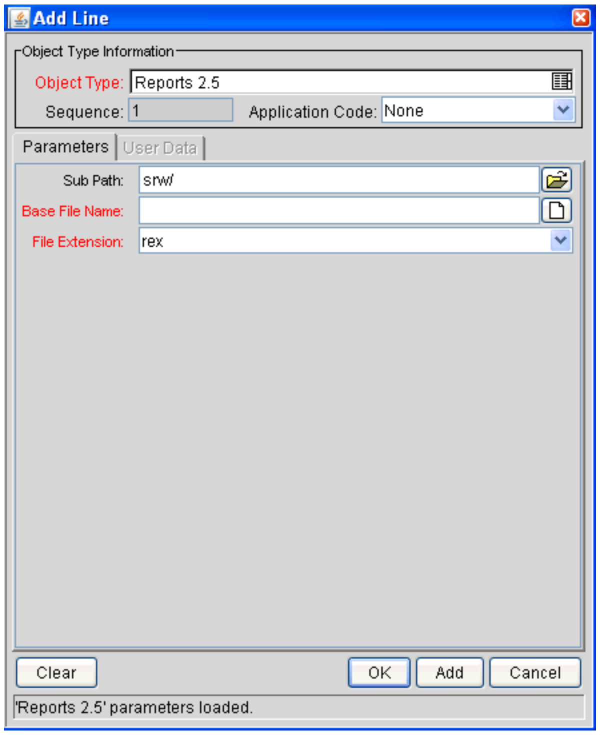
Reports 2.5 Object Type

The Reports 2.5 object type moves Oracle Reports 2.5 files from one instance to another. If a `rex` file is being moved, then an `rdf` file is generated at the destination. This migrates the report to the Oracle Account defined as the destination in the **Host** tab of the PPM Environment Workbench.

You can modify this object type to include steps to interact with your version control system.

"Figure 3-5. Reports 2.5 object type sample data" below shows the default screen when adding a package line that uses the Reports 2.5 object type. "Table 3-6. Reports 2.5 object type field descriptions" on the next page provides field descriptions for the object type.

Figure 3-5. Reports 2.5 object type sample data



Object Type:	Reports 2.5
Sequence:	1
Application Code:	None
Sub Path:	srw/
Base File Name:	
File Extension:	rex

'Reports 2.5' parameters loaded.

Table 3-6. Reports 2.5 object type field descriptions

Field Name (*Required)	Description
Sub Path	Subpath (relative to base path) to the Reports 2.5 files
*Base File Name	Name of the file to be migrated
*File Extension	File extension of the file to be migrated—`rex` or `rdf`

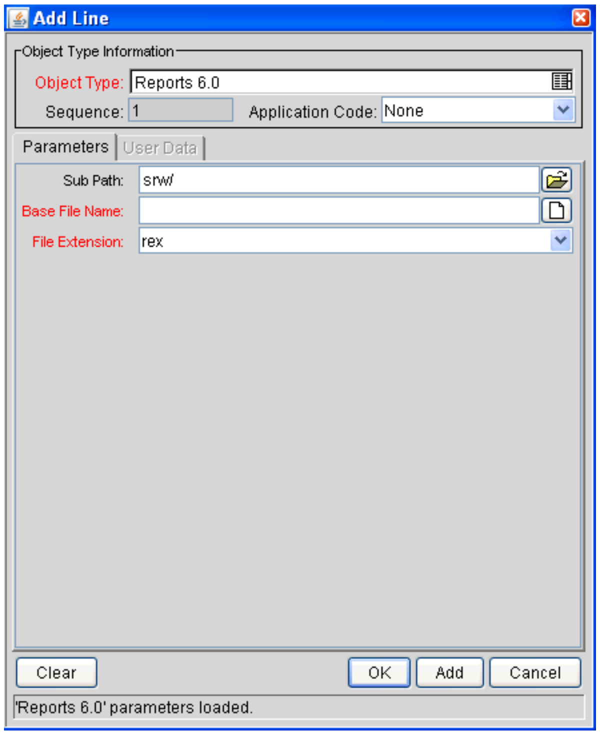
Reports 6.0 Object Type

The Reports 6.0 object type moves Oracle Reports 6.0 files from one instance to another. If a `rex` file is being moved, then an `rdf` file is generated at the destination. This migrates the report to the Oracle Account defined as the destination in the **Host** tab of the PPM Environment Workbench.

You can modify this object type to include steps to interact with your version control system.

"Figure 3-6. Reports 6.0 object type sample data" on the next page shows the default screen when adding a package line that uses the Reports 6.0 object type. "Table 3-7. Reports 6.0 object type field descriptions" on the next page provides field descriptions for the object type.

Figure 3-6. Reports 6.0 object type sample data



Object Type: Reports 6.0
Sequence: 1
Application Code: None
Sub Path: srw/
Base File Name:
File Extension: rex
'Reports 6.0' parameters loaded.

Table 3-7. Reports 6.0 object type field descriptions

Field Name (*Required)	Description
Sub Path	Subpath (relative to base path) to the Reports 6.0 files
*Base File Name	Name of the file to be migrated
*File Extension	File extension of the file to be migrated—`rex` or `rdf`

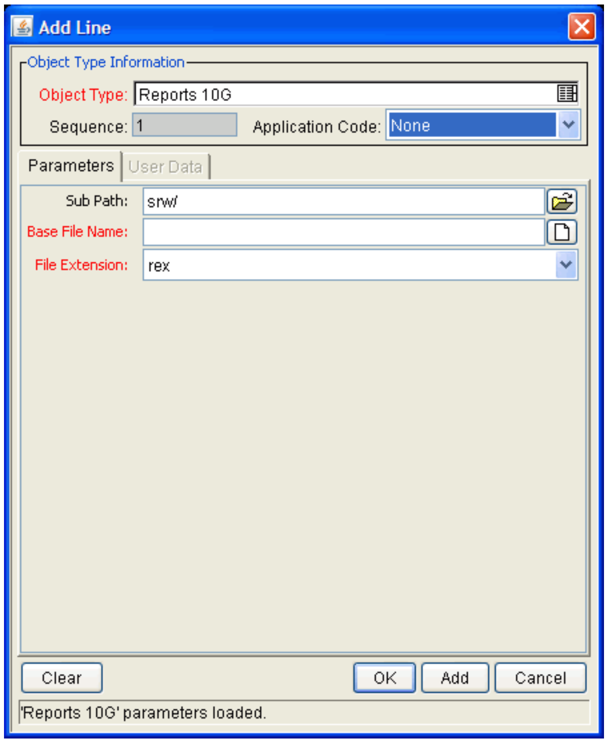
Reports 10G Object Type

The Reports 10G object type moves Oracle Reports 10G files from one instance to another. If a `rex` file is being moved, then an `rdf` file is generated at the destination. This migrates the report to the Oracle Account defined as the destination in the **Host** tab of the PPM Environment Workbench.

You can modify this object type to include steps to interact with your version control system.

"Figure 3-7. Reports 10G object type sample data" below shows the default screen when adding a package line that uses the Reports 10G object type. "Table 3-8. Reports 10G object type field descriptions" below provides field descriptions for the object type.

Figure 3-7. Reports 10G object type sample data



Object Type: Reports 10G
Sequence: 1
Application Code: None
Sub Path: srw/
Base File Name:
File Extension: rex
'Reports 10G' parameters loaded.

Table 3-8. Reports 10G object type field descriptions

Field Name (*Required)	Description
Sub Path	Subpath (relative to base path) to the Reports 10G files
*Base File Name	Name of the file to be migrated
*File Extension	File extension of the file to be migrated—`rex` or `rdf`

SQL Loader File Object Type

The SQL Loader File object type migrates the Oracle SQL Loader control file from one Oracle instance to another. The object type then optionally runs SQL Loader to load the data into an Oracle database at the destination instance.

"Figure 3-8. SQL Loader File object type sample data" below shows the default screen when adding a package line that uses the SQL Loader File object type. "Table 3-9. SQL Loader File object type field descriptions, continued" on page 30 provides field descriptions for the object type.

Note: The SQL Loader File object type does not, by default, use any of the fields in "Table 3-9. SQL Loader File object type field descriptions, continued" on page 30 that are not required. These fields are provided to allow users to customize and add to the SQL Loader call within the command steps.

Figure 3-8. SQL Loader File object type sample data

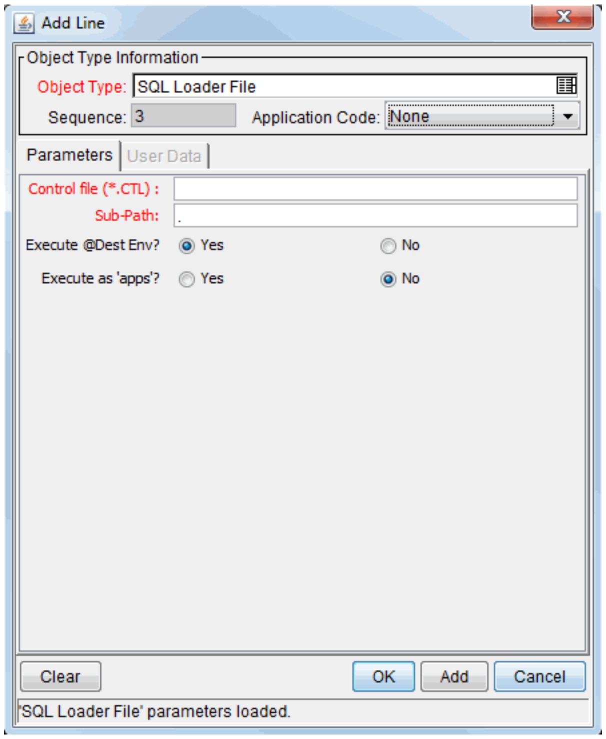

Object Type: SQL Loader File
Sequence: 3
Application Code: None
Control file (*.CTL) :
Sub-Path: .
Execute @Dest Env? [✓ Yes] No
Execute as 'apps'? Yes [✓ No]
'SQL Loader File' parameters loaded.

Table 3-9. SQL Loader File object type field descriptions

Field Name (*Required)	Description
Control file (.CTL)	SQL Loader control file name.

Table 3-9. SQL Loader File object type field descriptions, continued

Field Name (*Required)	Description
*Sub-Path	Subdirectory (relative to the base directory) for the environment where the control file is located.
Execute @Dest Env	If set to **Yes,** SQL Loader automatically loads data at the destination instance.
Execute as 'apps'	If set to **Yes,** and if Deployment Management Extension for Oracle E-Business Suite is installed, SQL Loader uses the username and password for the APPS (main database) user to load the data at the destination instance. If Deployment Management Extension for Oracle E-Business Suite is not installed, this field can still be used to differentiate between a central database login and an application-specific one.
Log file (Hidden by default)	Name of the log file generated by SQL Loader.
Bad file (Hidden by default)	Name of the file containing bad rows.
Data file (Hidden by default)	Data file name, if an external data file exists.
# of logical recs to skip (Hidden by default)	Number of logical records to skip.
Use Direct path (Hidden by default)	Option to use direct paths.
Max. num of discards (Hidden by default)	Maximum number of records that can be discarded before the load stops.
Discard file (Hidden by default)	Name of the file where discarded records are to be placed.
Extents file Hidden by default)	File from which to allocate extents.

Table 3-9. SQL Loader File object type field descriptions, continued

Field Name (*Required)	Description
Max # of errors (Hidden by default)	Maximum number of errors allowed before the load stops.
# of logical recs to load (Hidden by default)	Number of logical records to load.
Parameter file (Hidden by default)	Name of the file that contains the parameter specifications.
Run in silent mode (Hidden by default)	Option to suppress messages (header, feedback, errors, discards) during the run.

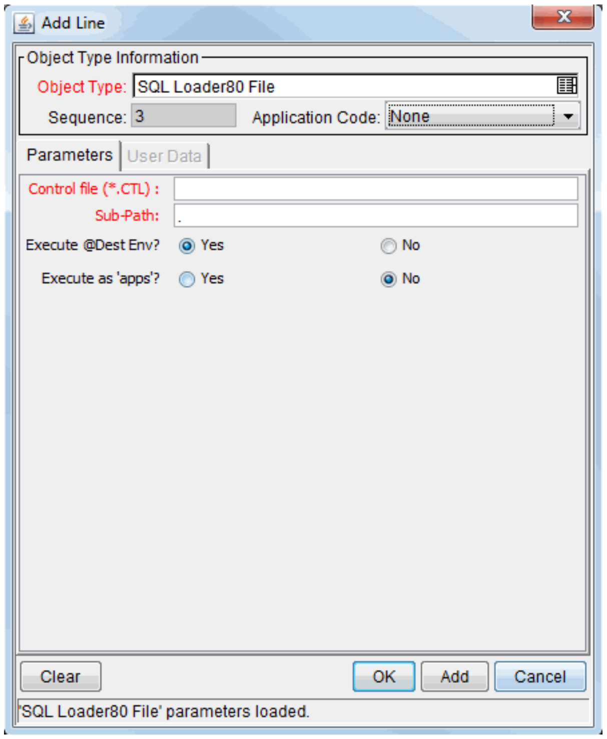
SQL Loader80 File Object Type

The SQL Loader80 File object type migrates the Oracle SQL Loader control file from one Oracle instance to another. The object type then optionally runs SQL Loader to load the data into an Oracle database at the destination instance.

"Figure 3-9. SQL Loader80 File object type sample data" on the next page shows the default screen when adding a package line that uses the SQL Loader80 File object type. "Table 3-10. SQL Loader80 File object type field descriptions, continued" on page 32 provides field descriptions for the object type.

Note: The SQL Loader80 File object type does not, by default, use any of the fields in "Table 3-10. SQL Loader80 File object type field descriptions, continued" on page 32 that are not required. These fields are provided to allow users to customize and add to the SQL Loader call within the command steps.

Figure 3-9. SQL Loader80 File object type sample data



Object Type: SQL Loader80 File
Sequence: 3
Application Code: None
Control file (*.CTL) :
Sub-Path: .
Execute @Dest Env? [✓ Yes] No
Execute as 'apps'? Yes [✓ No]
'SQL Loader80 File' parameters loaded.

Table 3-10. SQL Loader80 File object type field descriptions

Field Name (*Required)	Description
Control file (.CTL)	SQL Loader control file name.
*Sub-Path	Subdirectory (relative to the base directory) for the environment where the control file is located.
Execute @Dest Env	If set to **Yes,** SQL Loader automatically loads data at the destination instance.
Execute as 'apps'	If set to **Yes,** and if Deployment Management Extension for Oracle E-Business Suite is installed, SQL Loader uses the username and password for the APPS (main database) user to load the data at the destination instance. If Deployment Management Extension for Oracle E-Business Suite is not installed, this field can still be used to differentiate between a central database login and an application-specific one.

Table 3-10. SQL Loader80 File object type field descriptions, continued

Field Name (*Required)	Description
Log file (Hidden by default)	Name of the log file generated by SQL Loader.
Bad file (Hidden by default)	Name of the file containing bad rows.
Data file (Hidden by default)	Data file name, if an external data file exists.
# of logical recs to skip (Hidden by default)	Number of logical records to skip.
Use Direct path (Hidden by default)	Option to use direct paths.
Max. num of discards (Hidden by default)	Maximum number of records that can be discarded before the load stops.
Discard file (Hidden by default)	Name of the file where discarded records are to be placed.
Extents file (Hidden by default)	File from which to allocate extents.
Max # of errors (Hidden by default)	Maximum number of errors allowed before the load stops.
# of logical recs to load (Hidden by default)	Number of logical records to load.
Parameter file (Hidden by default)	Name of the file that contains the parameter specifications.
Run in silent mode (Hidden by default)	Option to suppress messages (header, feedback, errors, discards) during the run.

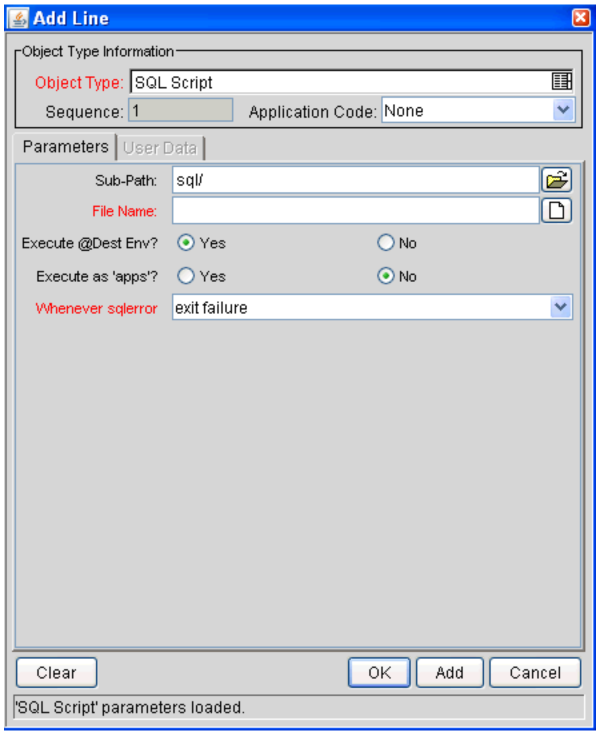
SQL Script Object Type

The SQL Script object type moves one or more SQL or PL/SQL scripts from one instance to another, and optionally executes the scripts at the destination. If you are running Oracle Applications with the APPS account as the default username, then the SQL Script object type has the ability to execute the script as APPS after the script has been migrated to the destination environment.

You can modify this object type to include steps to interact with your version control system.

"Figure 3-10. SQL Script object type sample data" below shows the default screen when adding a package line that uses the SQL Script object type. "Table 3-11. SQL Script object type field descriptions" on the next page provides field descriptions for the object type.

Figure 3-10. SQL Script object type sample data



Object Type: SQL Script
Sequence: 1
Application Code: None
Sub-Path: sql/
File Name:
Execute @Dest Env? [✓ Yes] No
Execute as 'apps'? Yes [✓ No]
Whenever sqlerror: exit failure

'SQL Script' parameters loaded.

Table 3-11. SQL Script object type field descriptions

Field Name (*Required)	Description
Sub-Path	Subpath (relative to base path) to the SQL script.
*File Name	Name of the SQL script to be migrated.
Execute @Dest Env	Option to execute the SQL script.
Execute as 'apps'	If set to **Yes,** and if Deployment Management Extension for Oracle E-Business Suite is installed, SQL Loader uses the username and password for the APPS (main database) user to load the data at the destination instance. If Deployment Management Extension for Oracle E-Business Suite is not installed, this field can still be used to differentiate between a central database login and an application-specific one.
*Whenever sqlerror	Whether the object type should **exit failure** or **continue** if an error is encountered while the script is executing.