PeopleSoft

EnterpriseOne Package Management 8.9 PeopleBook

September 2003

EnterpriseOne
Package Management 8.9 PeopleBook
SKU REL9EPK0309

Table of Contents

Overview of Package Management

The documentation suite for Configuration Planning and Setup is designed for Configurable Networking Computing (CNC) specialists, ERP 9.0 system administrators, and network and server administrators. This documentation is intended for users who have completed the initial ERP 9.0 installation and defined the standard data sources, path codes, and environments. Use the Configuration Planning and Setup documentation to make changes or additions to the configuration setup after the initial installation.

The Configuration Planning and Setup suite consists of the following guides:

- *Configurable Network Computing Implementation Guide*. This guide is written primarily for CNC specialists and contains information about the following topics:
  - Working with middleware
  - Understanding and verifying data sources
  - Creating path codes and environments
  - Working with Object Configuration Manager
  - Understanding the different modes of processing
  - Configuring ERP 9.0 for a typical customer
- *System Administration Guide*. This guide is written primarily for ERP 9.0 system administrators and contains information about the following topics:
  - Setting up data replication
  - Setting up printers
  - Working with servers
  - Setting up user profiles and security
  - Working with data dictionary and vocabulary overrides
  - Understanding transaction processing
  - Working with media objects and imaging
  - Using the universal table browser
  - Understanding ERP 9.0 naming conventions
  - Working with the jde.ini file
- *Package Management Guide*. This guide is written primarily for ERP 9.0 system administrators and others who manage custom modifications to the ERP 9.0 environments. The *Package Management Guide* contains information about the following topics:
  - Package management planning and setup
  - ERP 9.0 modification rules
  - Object management
  - Package build
  - Deployment
  - Multi-tier deployment

- *Server and Workstation Administration Guide*. This guide is written primarily for network administrators and contains information about the following topics:
    - Snapshot (multi-client installer)
    - Server administration
    - Troubleshooting the workstation and the server

Although every attempt has been made to organize the information in the Configuration Planning and Setup suite according to related tasks, you might find that the documentation for the various tasks that you need to perform appears in more than one guide. For example, the documentation for setting up path codes, environments, and data sources is in the *Configurable Network Computing Implementation Guide*, while the documentation for building and deploying packages is in the *Package Management Guide*.

The Configuration Planning and Setup suite is the central location for all CNC-related tasks except for the following:

- Initial installation of ERP 9.0. See the *ERP 9.0 Installation Guide.*
- ERP 9.0 upgrade and cumulative updates. See the *ERP 9.0 Upgrade Guide.*
- Network infrastructure and third-party software setup and maintenance. J.D. Edwards does not provide documentation for third-party software or hardware; the applicable software or hardware vendor provides this information.

You do not need to understand the installation process completely to perform configuration planning and setup tasks. However, to use the Configuration Planning and Setup guides you must understand what the installation accomplishes.

ERP 9.0 Roles

During an ERP 9.0 implementation, both customers and consultants typically fill the following roles:

- CNC consultant and CNC administrator
- Custom solution consultant and application developer
- Application consultant and application project leader
- Hardware, network, and third-party software consultant and administrator

Typically, both consultants and customers participate in an ERP 9.0 implementation. Consultants perform the following roles:

- CNC consultant
- Custom solution consultant
- Application consultant
- Hardware, network, and third-party software consultant

Customers perform the following roles, which parallel the consultant roles:

- CNC administrator
- Application developer
- Application project leader

- Hardware, network, and third-party software administrator

After the implementation, the consultants typically have fewer responsibilities. Therefore, customers must receive adequate training for the roles that they fill.

CNC Consultant and CNC Administrator

The CNC consultant and CNC administrator install ERP 9.0 and set up environments, users, security, distributed processing, and data replication. They also are responsible for setting up version control and testing various CNC configurations. The CNC consultant and CNC administrator control the deployment of ERP 9.0 software throughout the company.

Custom Solution Consultant and Application Developers

ERP 9.0 custom solution consultants resolve business issues by developing applications. Their primary responsibilities include designing the modifications with upgrades in mind; and developing, testing, and introducing the customized software. While the CNC administrator performs the version control functions that build and deploy software, the customer solution consultant must help develop the internal procedures that document the application development cycle for your business.

Application Consultants and Application Project Leaders

After ERP 9.0 is installed, configured, and deployed, the application consultants continue in their role as product experts, where they might be called on to troubleshoot problems that arise. Although application consultants do not implement the CNC configurations, they must understand how ERP 9.0 handles distributed processing, data replication, environments, and so on, because these application issues influence the CNC decisions.

Hardware, Network, and Third-Party Software Consultants and Administrators

Implementing ERP 9.0 includes many tasks that are outside the scope of J.D. Edwards services. Third-party consultants provide these services. They might also work as CNC consultants, network architects, custom modification consultants, and so on.

The Package Management Guide

This guide describes how to set up and maintain processes to develop and deploy custom modifications that are created with the ERP 9.0 tools.

This guide describes how to set up an environment in which you can deploy custom modifications that were made with ERP 9.0 Development tools. It also provides information about applying ERP 9.0 modification rules, transferring objects, checking out development objects, and working with the data dictionary.

If you have already set up your environment and are comfortable with these concepts, you can skip to the information that describes how to create and schedule packages and how to deploy packages to workstations and servers.

Packages

The purpose of a package is to group ERP 9.0 software modifications so that they can be deployed to workstations. A package describes where to find the components that you want to deploy to workstations. A package can contain everything needed to run ERP 9.0 (as in an initial installation for a new workstation) or only updates or changes to existing applications.

Following are the three package types:

- Full
- Partial
- Update

The full package includes all ERP 9.0 applications and is the same as the former Standard package. Users who need the full suite of ERP 9.0 applications should receive this type of package.

The partial package is a minimum configuration of ERP 9.0. It requires much less disk space than a full package. Users who install this type of package can load at runtime only the applications that they need, rather than receiving many unused ERP 9.0 applications with the initial installation.

The update package allows you to update, add to, or refresh your existing full or partial package with changed objects. You can deploy an update package only to a workstation that already has ERP 9.0 installed on it. The objects in the update package replace those objects on the workstation. All other objects on the workstation are left untouched. The advantage of this type of package is that you can quickly deploy software changes or enhancements.

Deployment Programs

ERP 9.0 offers several deployment programs, each with its own specific purpose and advantages. The method that you select depends mainly on the type of package that you want to deploy. You can use the following types of programs to deploy ERP 9.0:

- Workstation Installation
- Package Deployment
- Multi-tier Deployment

Understanding Workstation Installation

The Package Deployment program (P9631) is used to deploy full and partial packages. Update packages cannot be deployed using this program.

P9631 retrieves the items that are specified in the package. A full or partial package is like a bill of materials or a kit with instructions. To include all of the necessary components, the Package Deployment program deploys the package to the local computer. The user always initiates the deployment on a workstation.

Package Deployment

The Package Deployment program (P9631) enables the ERP 9.0 administrator to specify the date and time that a package is made available to users or groups. The administrator can specify whether the package is mandatory or optional. If a package is mandatory, users who receive the package will be unable to access ERP 9.0 until they install the package.

Users who receive a scheduled package can install the package immediately after they log on to ERP 9.0 on the specified date. If they decide to install the package, the installation program launches, and the package loads. The user can also choose to install the package later or to decline the installation altogether (unless the package is mandatory).

Alternatively, the push installation feature enables the administrator to schedule a package that will be automatically *pushed* from the deployment server to the workstations at the scheduled time, without requiring any interaction with the user.

Understanding Server Packages

A server package is a set of ERP 9.0 objects that you have grouped together with a common SAR number or any version control tracking number. Objects consist of specification records, source files, and header files. Compiled objects are created on the enterprise servers. When you define a package, you can install it on multiple servers and multiple path codes.

All ERP 9.0 application development takes place on a workstation, and objects are stored in a central location, either on the deployment server or an enterprise server. ERP 9.0 allows you to store business applications on an enterprise server. Developers and system administrators can push a group of objects, called a server package, from the central objects data source to a replicated server path code.

Although server packages are not the same as the workstation packages, they are built and deployed using the same applications: Package Assembly (P9601), Package Build (P9621), and Package Deployment (P9631). When you create an update package for workstations, you should create a corresponding server package to ensure that you are deploying the same objects to your enterprise server that you deploy to the workstations.

Understanding Multi-Tier Deployment

Multi-tier deployment enables ERP 9.0 workstations to install software from more than one deployment location and more than one deployment server. J.D. Edwards recommends that you consider multi-tier deployment if your site has more than 50 workstations performing ERP 9.0 software installations per day, or if you are deploying ERP 9.0 software across a WAN connection.

Package Management Planning and Setup

Managing modifications requires a practical version control plan for tracking changed objects. You can avoid many software problems by tracking changed objects.

To more easily plan and track your development and to simplify version control, you should build and deploy packages only as often as necessary. If you perform many development changes, you should build and deploy packages on a set schedule to ensure that everyone involved knows when objects are due to be completed and when you are going to build and deploy the package.

Implementing version control might require a staff of information technology professionals. For example, J.D. Edwards has several hundred developers and a complex Configurable Network Computing (CNC) configuration. To manage version control for multiple developers, the product version control group consists of the following:

- One manager to oversee coordination within the department
- One supervisor to coordinate the package builds, coordinate object transfers, and troubleshoot problems
- Two server specialists to build server packages
- Four technical specialists to build workstation packages, perform object transfers, and run automated testing before releasing the package to Quality Assurance
- One night operator to build workstation packages, build server packages, and clear build errors

Working with Recommended Path Codes

If you are not planning any development projects, you need only three path codes (sets of central objects): PY9, PD9, and JD9. If you plan to modify the software extensively, you also should create a development path code (DV9).

Because each path code requires version control maintenance, you should create only the path codes that you really need. Even when you make extensive software modifications, you should have only the following four path codes:

PY9 A path code that contains a practice set of objects that you test during conference room pilot before you transfer objects to production. Use this path code to deploy quick corrections or make minor modifications to objects that you will transfer to production. You also can use this path code to test modifications that were made in the development path code before you transfer the objects to the production path code.

PD9 The production path code from which just-in-time installations and production server objects are deployed. After you test software changes in PY9, transfer them to PD9, and then deploy the changes to your enterprise servers and workstations.

JD9 The set of pristine objects that is included with J.D. Edwards software. You should not make changes to this path code other than to apply changes that J.D. Edwards documents. Use this path code to compare J.D. Edwards standard software to any custom software that you have implemented in other path codes. You should keep a copy of this path code so that you have an

unchanged copy of ERP 9.0 in case you need to undo any of your changes.

DV9 The path code that you use for routine development. After successfully testing the objects that you develop, transfer them to your PY9 path code using the Object Transfer program, and distribute them to your users using the package build deployment process.

All path codes share the same Object Librarian tables, the same system data source, and, usually, the same data dictionary. The only tables that are distinct for each path code are the central objects and specifications tables (tables that begin with F987), the Versions List table (F983051), the Processing Option Text table (F98306), and the User Overrides Table (F98950).

Disabling Just-In-Time Installation

Choose one of the following methods to disable JITI:

- Use the Work with Environments program (P0094) to disable JITI for the environment. When JITI is disabled, users who sign on to that environment can access only those applications for which they have specifications. J.D. Edwards recommends that you use this method during the cumulative installation process when you update your production central objects with new J.D. Edwards changes.
- Use application security to disable JITI for a particular application, for an end user, or for a group profile. When a user accesses an application for which the specifications do not reside on the user's workstation, the system first reviews P0094 to determine whether JITI is disabled for the entire environment. If JITI is on for the environment, the system determines whether application security prevents the user from using JITI. Application security has a field called "not allowed to install."

Although you can disable JITI for an environment, the system still uses JITI to copy data dictionary items to the global tables. The structure of the data dictionary prevents you from disabling JITI for it.

See Also

- *Security* in the *System Administration Guide* for more information about setting up application security.

Suggested Package Names

J.D. Edwards recommends that you maintain two versions of each package—an A and a B version—so that you can alternate between these versions when you build packages. The advantage of this approach is that users always have a package available to them, even when you are building the latest version of that package. For example, package PRODB would be available to users while you are building PRODA. Then, after you release PRODA, you would build the next package into PRODB, and so on.

If you are using both full and partial packages, you will have four packages for each path code. This setup gives you two full packages (A and B) for production and two partial packages (A and B) for production, as illustrated in the following table:

PD9FA Standard Production Full A

PD9FB Standard Production Full B

PD9PA Standard Production Partial A

PD9PB Standard Production Partial B

Update packages might be named as follows:

PD9UA Production Update Package 1

PD9UB Production Update Package 2

PD9UA Production Update Package 3

PD9UB Production Update Package 4

Ensuring the Integrity of the Production Environment

As soon as you transfer objects into your PD9 path code, end users can access the changes. Therefore, you should test the modified objects before you transfer them to the production path code.

After you transfer objects to the production path code, they are immediately deployed to end users under the following circumstances:

- When a user is set up for just-in-time installation (JITI), the system automatically deploys the object to the user's workstation the first time that the user attempts to access the object.

  Note
  JITI affects users who are using a partial package, as well as those who received an update package containing an application without specifications. For more information about JITI, see *Understanding Just-in-Time Installation*.

- When you build an update or partial package that includes a business function build for that package, the system builds the business function and then globally links it with all other business functions in the check-in location for the path code.

To ensure that the modifications that you transfer to your production path code are not immediately available to end users, avoid using partial packages or update packages that contain applications with no specifications. Also, do not transfer business functions into the production path code until you are ready to deploy because, during a package build, a global build of business functions automatically includes the new functions. When you transfer changes into the production path code, they will not be available to users until you build a full or update package.

Understanding the Development Processes

This topic includes information that will help you understand the development processes that J.D. Edwards recommends. It includes information about developing long-term enhancements and short-term fixes.

Working with the Normal Development Process

The following lists provide an overview of how you should perform your normal development cycle.

In the DV9 path code, complete the following tasks:

- Make modifications
- Test the modifications
- Transfer the objects to PY9

In the PY9 path code, complete the following tasks:

- Build the package
- Test the modifications
- Deploy server objects to the CRP path code on the enterprise server, and then test the objects
- Schedule the package
- Transfer the objects to PD9

In the PD9 path code, complete the following tasks:

- Build the package
- Schedule the package
- Deploy the server objects to the PD9 path code on the enterprise server, and then test the objects

► To work with the normal development process

1. Check out your objects from the DV9 path code, modify them, test them, and check them back in.
2. Use the SAR system (or any numbering system) to track your changes.

   Always use a SAR number when you check in the objects.
3. If the objects need to reside on the logic server, transfer them to the DV9 path code on that server.
4. Test the objects by comparing them to the objects on the server.
5. Use a SAR number with Object Transfer to transfer the objects to the PY9 path code.

   Use the check-out log to confirm the transfer (optional). The objects are not in production, but they are now available for you to build a test package in the PY9 path code.
6. Build a full, partial, or update package.

7. Test the newly built but unreleased package in the PY9 path code.

   You test the package only by comparing it to workstation processes, not to server processes. Although the name of this package will probably be PY9U1 (update package number 1 for the CRP path code), it is a test package because you have not released it to your users.

8. Schedule the update package to deploy to a test machine and test it in an environment that contains CRP objects with CRP data.
9. Deploy server objects to the PY9 path code on the enterprise server and test them.

   If you prefer, you can build the server package and schedule the deployment at the same time that you build and schedule the workstation package. Building these packages simultaneously can save you time, although this method puts a greater load on the server.

10. Schedule the new package to deploy to CRP users.
11. Use a SAR number with Object Transfer to transfer the object to the PD9 path code.

    Use the check-out log to confirm the transfer (optional). The objects are now in the production environment and are available for you to build a package in the PD9 path code.

    Note
    If just-in-time installation is enabled, users can access the objects immediately.

12. Build a full, partial, or update package for client workstations.
13. Perform a server package build.

    You can transfer the server package now or wait until it has been tested on a workstation.

14. Schedule the new package to deploy to end-user workstations.
15. Deploy the server objects to the PD9 path code on the enterprise server and test them.

    If you prefer,you can build the server package and schedule the deployment at the same time that you build and schedule the workstation package.

Developing Short-Term Changes

Sometimes you need to make a simple change to an application that is undergoing major enhancement work in the DV9 path code. When an object has been modified in the DV9 path code, you might encounter unpredictable results if you make a simple change and quickly deploy it. Therefore, you should make the change in both the PY9 and the DV9 path codes, and deploy the change via the PY9 path code. This method allows you to deploy the change quickly to users without interfering with the major enhancement work in the DV9 path code

Deployment Methods

After you have made software changes, the method that you use to deploy those changes to the workstations on your enterprise depends on factors such as the type of package that you typically build and the needs of your users.

Cumulative and Noncumulative Update Packages

When you use a cumulative update strategy for deploying packages, you have one package that you add to, rebuild, and re-release to users. You do not create a new package each time you have a modification that you want to deploy. To use a cumulative package, follow these steps:

1. Change the package status in order to add the new modifications.
2. Add the changed or new objects to the package.
3. Redeploy the package.

When you use a noncumulative update strategy, you create and deploy a different package each time you add or change an object. For example, if you deploy one modification a week for 10 weeks, you would have 10 different packages, each containing only the software change for that week.

Recommendations for Sites Using Partial Packages with JITI

For sites that use partial packages and JITI, J.D. Edwards recommends that you adopt a noncumulative update package strategy. Each week that you build a new update package that contains modifications that cannot be deployed through JITI (such as business function changes), you should also rebuild the partial package for that week to ensure that the partial package contains the latest changes. When you want to deploy a package to a new workstation (or completely refresh any workstation), you can install this partial package rather than installing an update package.

Recommendations for Sites Using Full Packages with JITI

For sites that use full packages with JITI, J.D. Edwards recommends that you adopt a cumulative update package strategy. Each week that you need to deploy a change, add that object to the existing update package, and then rebuild and schedule the package.

The advantage of this strategy is that you do not need to rebuild your full package each week. By using this strategy to deploy packages to a new workstation (or to completely refresh any workstation), you must install the full package and then install one cumulative update package.

The disadvantage of this strategy is that the update package might become so large that the deployment time increases. You must determine when to rebuild the full package for new workstations and allow existing users to install the new update package.

Recommendations for Developers

Developers must use full packages. You can deploy update packages to them, but developers do not use partial packages because workstations that contain partial packages never receive development objects.

When installing a full or update package, workstations that contain compilers automatically receive development objects. Developers receive source code, headers, and libraries. The installation process builds the objects into the business function library and dynamic link library (DLL), and thereby preserves any business function changes that the developer made.

Multitiered Deployment and Just-in-Time Installation

JITI can be used in remote locations that are using multitiered deployment to install packages. However, you might find that performance time for the JITI is unacceptable. In this case, you can initially install full packages and use update packages to deploy software changes.

Comparing Deployment Methods

Each deployment method has strengths and limitations. To help you decide which method is right for your needs, key points about the different methods are presented below:

- A new user loading a new machine should use the Install Manager program to load a full or partial package, plus any update packages that you have instructed users to load since the last package build. Therefore, you need a manual tracking system to track which update packages must be applied after installing a particular package.
- All update packages must use the Package Deployment program (P9631) to be scheduled to workstations unless you are using the Push Installation feature.
- The full and partial packages can also use Package Deployment if ERP 9.0 is already loaded on a machine. (You can use the Push Installation feature to deploy to a machine that does not have ERP 9.0 installed.)
- If workstation disk space is a concern, consider deploying a partial package that provides the minimum configuration of ERP 9.0 and yet allows users to load only the applicable ERP 9.0 applications. In this case, you could use either the Install Manager program or the Package Deployment program to deploy the partial package.
- Use the Silent Installation program to submit a Workstation Installation request through command line arguments. Do not use this program for an initial installation.
- Use the Multitier Deployment process to install from more than one deployment location. You should consider this method if you have more than 50 workstations performing ERP 9.0 software installations per day.
- Use the Package Deployment program (P9631) when you need to push objects in a server package from the central objects data source to enterprise servers.

Deploying Various Types of Modifications

An understanding of which types of objects (and therefore modifications) can be deployed through each package type will help you choose the appropriate package for the changes you deploy.

Partial packages include the following:

- All DLLs for consolidated business functions
- Specifications for only those applications that are necessary for logging in and accessing ERP 9.0 Explorer
- All icons
- Helps, data, and ERP 9.0 foundation

The following table illustrates which types of changes are installed with a full package, an update package with specifications, and an update package with no specifications.

Modification	Full Package	Update Package with Specs	Update Package with No Specs
Applications			
Imbedded event rules	X	X	X
Vocabulary overrides (FDA text)	X	X	X
Data structure	X	X	X
Processing options (report)	X	X	X
Business Functions			
C Language source/include/object (if a compiler exists)	X	X	
Consolidated business function DLLs	X	X	
Data structure	X	X	
Table event rules	X	X	
Named event rules	X	X	
Batch Applications			
Report	X	X	X
Imbedded event rules in a report	X	X	X
Report data structure	X	X	X

Modification	Full Package	Update Package with Specs	Update Package with No Specs
Report vocabulary overrides	X	X	X
Report processing options	X	X	X
Versions and processing option values (depends on processing options)	X	X	X
Imbedded event rules in versions	X	X	X
Processing option templates	X	X	X
Business Views			
Added or changed fields	X	X	X
Tables			
Structure (specifications)	X	X	
Indexes	X	X	
Joins	X	X	
Miscellaneous			
Helps stored on the server			
Helps stored on the workstation	X	X	
Generic text data structure	X	X	
Data dictionary items			
Foundation code (required for full and partial packages, optional for update packages)	X	X	X
Foreign languages	X	X	X
Non-ERP 9.0 objects (custom items can be deployed through any package type)	X	X	X
Replicated local data (required for full and partial packages, optional for update packages)	X	X	X
New icons	X	X	

For an update package with JITI types, the following changes are installed:

Applications

- Imbedded ER
- Vocabulary overrides (FDA text)
- Data structure
- Processing options (report)

Batch Applications

- Report
- Imbedded ER in a report
- Report data structure
- Report vocabulary overrides
- Report processing options
- Versions and processing option values (depends on processing options)
- Imbedded ER in versions

Business Views

- Added or changed fields

Miscellaneous

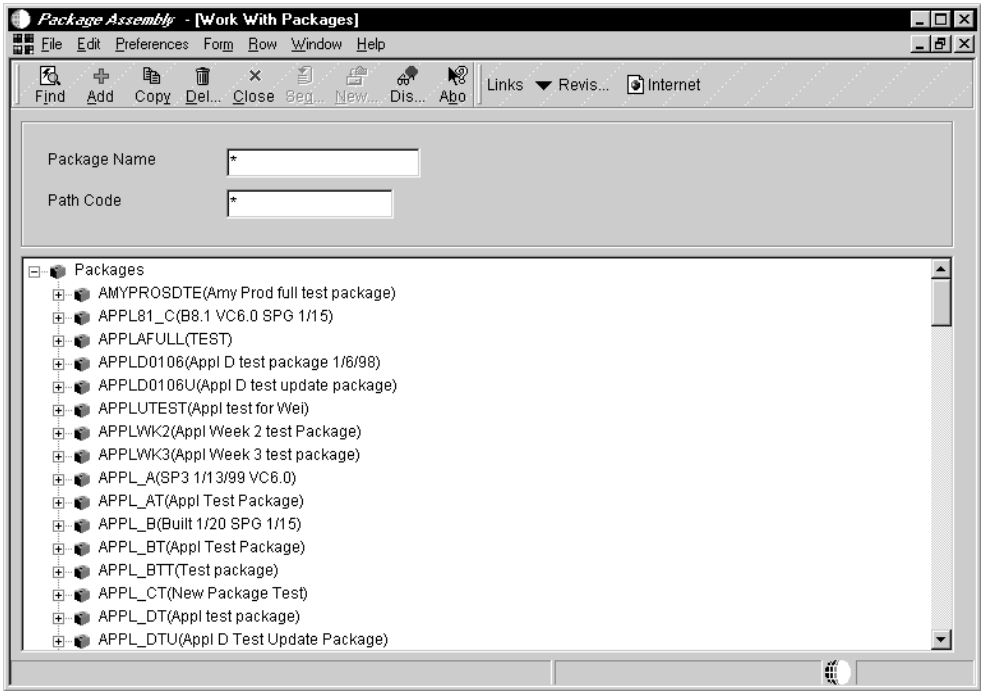
- Foundation code (required for full and partial packages, optional for update packages)
- Foreign languages
- Non-ERP 9.0 objects (custom items can be deployed through any package type)
- Replicated local data (required for full and partial packages, optional for update packages)

Deploying Data

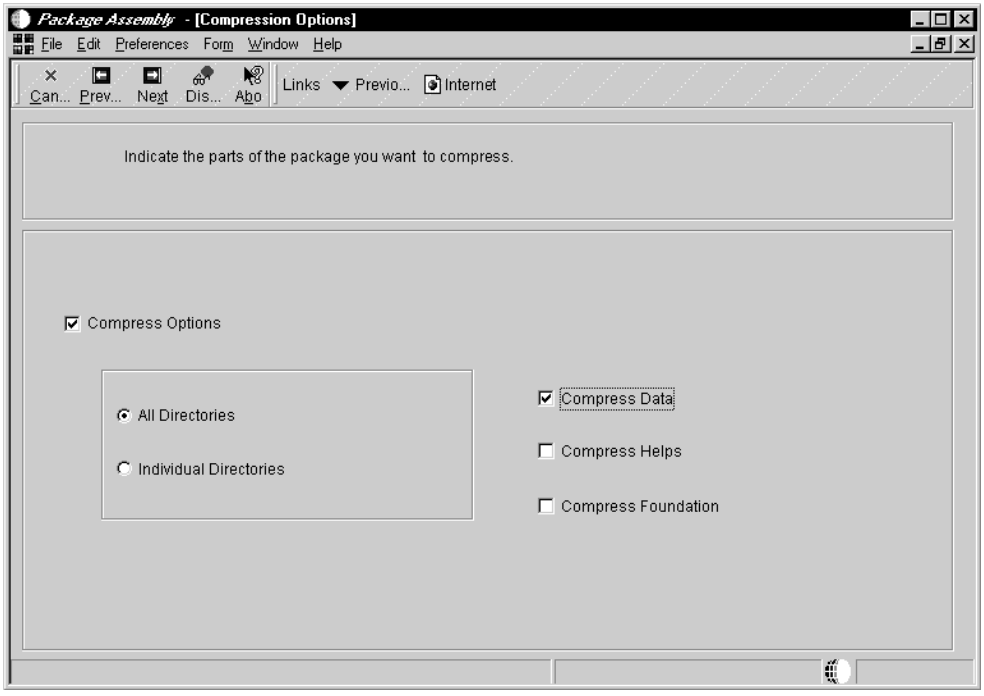
This topic describes how to deploy a new replicated local database and a special database for store-and-forward users.

► To deploy a new replicated local database

From the Package and Deployment Tools menu (GH9083), choose Package Assembly.



1. On Work With Packages, choose the package and choose Define Build from the Row menu.
2. Proceed through the Package Assembly Director forms until you reach the Compression Options form.



3. On Compression Options, turn on only the Compress Options and Compress Data options.

   These options recompress the database so that anyone who installs this package will receive current data.

Note
To verify that compression was successful, review the size and date of the JZ file that appears in the PeopleSoft/B9/*pathcode*/package/data folder. If the file is 1KB or smaller, the compression failed.

Deploying a Special Database for Store-and-Forward Users

Store-and-forward users require J.D. Edwards Supported Local Databases that are much larger than the databases required for other users. Therefore, you should build an update package with one line item for the database for store-and-forward users.

Because store-and-forward users cannot connect to the network to install their applications using just-in-time installation, they must install full packages. Therefore, store-and-forward users should use an environment that has just-in-time installation turned off. Instruct your store-and-forward users to install the normal full package, and then install the update package that was created for store-and-forward users.

► To deploy a special database for store-and-forward users

From the Package and Deployment Tools menu (GH9083), choose Package Assembly (P9601).

1. On Work with Packages, choose the package, and then choose Define Build from the Row menu.
2. Proceed through the Package Assembly Director forms until you reach the Compression Options form.
3. On Compression Options, turn on the Compress Data option.
4. Schedule the package to deploy to all workstations that will be performing store-and-forward transactions.

ERP 9.0 Modification Rules

Because the ERP 9.0 Development Tools are comprehensive and flexible, you can customize certain aspects of business solutions and applications without making custom modifications. J.D. Edwards refers to this concept as *modless modifications*, which are modifications that you can perform easily without the help of a developer. You can perform modless modifications on the following:

- User overrides
- User defined codes
- Menu revisions
- All text
- Processing options values
- Data dictionary attributes
- Workflow processes

This flexibility improves efficiency and provides distinct advantages, such as the ability to do the following:

- Export grid records to other applications, such as a Microsoft Excel spreadsheet
- Resequence a grid on a different column
- Change grid fonts and colors
- Control major system functions using processing options

Although J.D. Edwards makes tools that are as flexible and robust as possible, you might need to modify ERP 9.0 more extensively. To ensure that the modifications perform like modless modifications and to provide a seamless and predictable upgrade to the next release level, verify that any software modifications that you make comply with the rules and standards that J.D. Edwards recommends..

To ensure a smooth upgrade, you should prepare for the upgrade before you make any custom modifications. If you plan modifications properly, you can minimize the tasks that you need to perform following an upgrade. Planning usually reduces the time required to upgrade your software, which reduces disruption to your business and the overhead cost of the upgrade.

ERP 9.0 tracks all custom modifications as you check them into the server. Before you perform an upgrade, you can run the Object Librarian Modifications report (R9840D) to see a list of the changed objects.

ERP 9.0 consists of control tables, such as menus, user defined codes, versions, and the data dictionary, and transaction tables, such as Address Book Master (F0101). J.D. Edwards provides control tables with data that you can modify, while transaction tables contain your business data.

During an upgrade, both sets of tables go through an automatic merge process. The system merges control tables with new J.D. Edwards data and converts transaction tables to the new specifications without change to your existing data. For the object specification merges (such as business views, tables, data structures, processing options, event rules, and applications), the system merges the specifications or replaces them, depending on the rules that are defined in ERP 9.0.

What an Upgrade Preserves and Replaces

If your business needs require custom modifications to ERP 9.0 software, use the following general definitions to ensure a smooth and predictable upgrade. These definitions describe which modifications the upgrade process preserves and which modifications it replaces.

- Preserve

  During an upgrade, ERP 9.0 automatically merges your modifications with the new J.D. Edwards applications that are shipped with the upgrade, and that you do not lose your modifications. If a direct conflict exists between your specifications and J.D. Edwards specifications, the upgrade process uses your specifications. When no direct conflict exists, then the upgrade process merges the two specifications.

- Replace

  The upgrade does not merge your modifications with new J.D. Edwards applications and, therefore, the new software replaces your modifications. You must recreate your modifications after the upgrade completes.

Run the Object Librarian Modifications Report (R9840D) before the upgrade process to identify the objects that you modified.

General Rules for Modification

The following general modification rules apply to all ERP 9.0 objects:

- When adding new objects, use system codes 55 - 59.

  ERP 9.0 uses reserved system codes that enable it to categorize different applications and vertical groups. When you use system codes 55 through 59 for your custom modifications, ERP 9.0 does not overlay your modifications with J.D. Edwards applications.

- Do not create custom or new version names that begin with ZJDE or XJDE.

  These prefixes are reserved for standard version templates that J.D. Edwards includes with the software, and these prefixes do not preserve your custom versions in case of a naming conflict. You can copy the pristine versions to create new templates or versions.

- For upgrades, build a package from the last modified central objects set and perform backups of your development server, central objects, and Object Librarian data sources so that you can access those specifications for comparison or for troubleshooting purposes.

Interactive Applications

Do not delete controls, grid columns, or hyper items on existing ERP 9.0 applications. Instead, hide or disable them. ERP 9.0 might use these items for calculations or as variables, and deleting them might disable major system functions.

What an Upgrade Preserves and Replaces

The following table shows the interactive application elements that are preserved or replaced during an upgrade.

Object	Preserved	Replaced	Comments
New applications	X		You can either create an application from scratch, or copy an existing application using the Copy feature in Application Design Aid. This feature enables you to copy all of the application specifications, including event rules. If you use the Copy feature to copy an existing application for some modifications, during an upgrade your new application does not receive any changes that J.D. Edwards might have made to the original application.
New hyper items added to existing forms	X		
New controls added to existing forms	X		
New grid columns added to existing forms	X		
Style changes	X		Style changes include fonts and colors. New J.D. Edwards controls have the standard base definitions. If you adjust the style, you need to also adjust the styles for any new controls that you added to an application.
Code-generator overrides	X		
Data dictionary overrides	X		
Location and size changes	X		In a subsequent release of the software, J.D. Edwards might place a new control in the same place that you have placed a custom control. In this case, the new control appears on top your custom control. This situation does not affect the event rules or the functions of the application. After the upgrade, you can use Application Design Aid to rearrange the controls.
Sequence changes for tabs or columns	X		The upgrade process adds new J.D. Edwards controls to the end of your custom tab sequence. You can review the tab sequence after an upgrade.
Custom forms on existing ERP 9.0 applications		X	Instead of adding custom forms to existing ERP 9.0 applications, create a custom application using system codes 55 through 59, and then place the custom form on that custom application. You can then add to existing applications form exits and row exits that call your custom forms within your custom applications. System performance is not adversely affected when you call an external application from a row exit instead of from a form within the application.

Reports

The following rule applies to Report Design Aid specifications:

Do not delete objects on existing ERP 9.0 reports. Instead, hide them. ERP 9.0 might use these objects for calculations or as variables, and deleting them might disable major system functions.

What an Upgrade Preserves and Replaces

The following table shows the report elements that are preserved or replaced during an upgrade.

Object	Preserved	Replaced	Comments
New reports	X		You can either create a report from scratch or copy an existing report using the Copy feature in Report Design Aid. This feature enables you to copy all the report specifications, including event rules. If you use the Copy feature to copy an existing report for some modifications, during an upgrade your new report does not receive any changes that J.D. Edwards updates might have made to the original report.
New constants added to existing reports	X		
New alphabetical variables added to existing reports	X		
New numeric variables added to existing reports	X		
New data variables added to existing reports	X		
New runtime variables added to existing reports	X		
New database variables added to existing reports	X		
New data dictionary variables added to existing reports	X		
Style changes	X		Style changes include fonts and colors. New J.D. Edwards controls have the standard base definitions. If you have adjusted the default style, you need to also adjust the styles for any new controls that you

Object	Preserved	Replaced	Comments
			added to a report.
Location and size changes for objects	X		In a subsequent release of the software, J.D. Edwards might place a new object, such as a control, in the same place as you placed a custom object. In this case, the objects display right next to each other. This situation does not affect the event rules or the functions of the report in any way. After the upgrade, you can use Report Design Aid to rearrange the objects.
Data dictionary overrides	X		
Custom sections on existing ERP 9.0 reports		X	Instead of adding custom sections to existing reports, use Report Interconnect and connect to a new custom report that uses system codes 55 through 59. System performance is not adversely affected when you call a report through report interconnections.

Application Text Changes

The following table shows the application text that is preserved or replaced during an upgrade.

What an Upgrade Preserves and Replaces

The following table shows the application text elements that are preserved or replaced during an upgrade.

Object	Preserved	Replaced	Comments
Overrides performed in Application Design Aid	X		
Overrides performed in Report Design Aid	X		
Overrides performed in Interactive Vocabulary Override	X		
Overrides performed in Batch Vocabulary Override	X		

Table Specifications

An upgrade merges your table specifications from one release level to the next.

What an Upgrade Preserves and Replaces

The following table shows the table specification elements that are preserved or replaced during an upgrade.

Object	Preserved	Replaced	Comments
New Tables	X		
Custom Indices to ERP 9.0 tables	X		
Columns added to or removed from existing		X	This includes changing field length, field type

removed from existing ERP 9.0 tables			field length, field type, and decimal position. Instead of adding a new column to an existing J.D. Edwards table, use a tag table with system codes 55 through 59.

For custom tag files, be aware of data item changes in the J.D. Edwards data dictionary. In subsequent releases, J.D. Edwards might change certain attributes of a data item, such as its size, that might affect data integrity and how data is stored in the database. For this reason, you might need to use the Table Conversion tool to convert the tag file data to the new release level. For base J.D. Edwards files, the upgrade process automatically applies the data dictionary to the new release level. An upgrade preserves custom indices for the custom tag files.

Control Tables

Control tables contain user defined codes (UDCs), menus, and data dictionary items. An upgrade merges your control tables from one release level to the next using the change table process, which uses your control tables, not ERP 9.0 tables, as the basis for the data merge.

What an Upgrade Preserves and Replaces

The following table shows the control table elements that are preserved or replaced during an upgrade.

Object	Preserved	Replaced	Comments
Data dictionary custom changes	X		This includes changes to row, column, and glossary text. The upgrade process uses your data dictionary as the base, and in case of a conflict with ERP 9.0 data items, your changes override. Create new data items using system codes 55 through 59.
User defined codes	X		The upgrade process merges any new, hard-coded ERP 9.0 values. (Values owned by J.D. Edwards are stored in systems 90 and above, and H90 and above.) The process also reports any ERP 9.0 hard-coded values that conflict with your custom values.
Menus	X		In case of a conflict with ERP 9.0 base menus, your custom changes override.
Columns added or removed from existing ERP 9.0 control tables		X	

Business Views

Do not remove columns from existing business views. Changing business views that applications use can cause unpredictable results when you run the application. If you need to hide columns, do so within the application design using either Application Design Aid or Report Design Aid. Deleting a few columns from a business view does not significantly improve system performance.

What an Upgrade Preserves and Replaces

The following table shows the business view elements that are preserved or replaced during an upgrade.

Object	Preserved	Replaced	Comments
New custom business views	X		
New columns, joins, or indexes added to existing ERP 9.0 business views	X		
Columns removed from J.D. Edwards business views		X	

Event Rules

During the upgrade process, ERP 9.0 checks for customized event rules that conflict with new event rules that the software installs. If a conflict exists, ERP 9.0 disables the custom event rules and appends them to the end of the new event rules. The following table shows which custom event rules are preserved and which are disabled.

What an Upgrade Preserves and Replaces

The following table shows the event rule elements that are preserved or replaced during an upgrade.

Object	Preserved	Replaced	Comments
Custom event rules for custom applications, reports, and tables	X		
Custom event rules for custom business functions	X		
Custom event rules on a new custom control	X		
Events for System applications, reports, and tables that do not have any System event rules attached to the same event	X		
Events for System business functions that do not have any System event rules attached to the same event	X		
Events for System applications, reports, and tables that have existing event rules attached to the same event		X	An upgrade disables and appends your custom event rules to the end of the standard event rules. During the merge process, the system prints the Object Librarian Modifications report (R9840D) to notify you of any event rules that the upgrade process disabled.
Events for System business functions that have event rules attached to the same event		X	An upgrade disables and appends your custom event rules to the end of the standard event rules. During the merge process, the system prints the Object Librarian Modifications report to notify you of any event rules that the upgrade process disabled.

To restore your custom event rules to System objects, highlight and drag the event rules back to the proper place in the event and enable them. Prior to an upgrade, perform the following tasks:

- Run the Object Librarian Modifications report to identify modified applications
- Print the event rules for the modified application so that you can see the logic for the event when you restore custom event rules

Data Structures

What an Upgrade Preserves and Replaces

The following table shows the data structure elements that are preserved or replaced during an upgrade.

Object	Preserved	Replaced
Custom forms data structures	X	
Custom processing options data structures	X	
Custom reports data structures	X	
Custom business functions data structures	X	
Custom generic text data structures	X	
Modifications to existing System forms data structures		X
Modifications to existing System processing options data structures		X
Modifications to existing System reports data structures		X
Modifications to existing System business functions data structures		X
Modifications to existing System generic text data structures		X

To bring forward to the next release level the custom modifications that you made to System data structures, run the Object Librarian Modifications report (R9840D) to list all of the modified data structures, and use this report as a guide when you manually enter data structure changes.

Business Functions

For any new custom business functions (BSFNs), create a new (custom) parent DLL to store your custom modifications. See the *System Development Tools Guide* for details.

To bring your custom changes forward to the next release level, run the Object Librarian Modifications (R9840D) report to list all of the modified business functions, and use this report as a guide when you manually enter the business function changes.

Always use the standard API (jdeCallObject) to call other business functions from within a business function. Not following this and all other standards will cause problems.

To determine whether the source code of existing base business functions has been modified, use a third-party source-compare tool, such as Microsoft WinDiff. To determine modifications to APIs within business functions, see the online help feature for the most current information about APIs.

What an Upgrade Preserves and Replaces

The following table shows the business function elements that are preserved or replaced during an upgrade.

Object	Preserved	Replaced	Comments
New custom business function objects	X		
Modifications made to existing System business function objects		X	NER BSFNs can be modified

Versions

For new custom versions, create a new version with a name that does not begin with XJDE or ZJDE. See the *ERP 9.0 Foundation Guide* for details.

What an Upgrade Preserves and Replaces

The following table shows the versions elements that are preserved or replaced during an upgrade.

Object	Preserved	Replaced
Non-JDE versions	X	
Version specifications	X	
Processing option data	X	
All ZJDE and XJDE version specifications		X
All processing option data for XJDE versions		X

In addition, processing option data is copied but not converted for non-JDE versions that use J.D. Edwards processing option templates. A warning is issued at runtime, and some data may be lost.

Also, event rule modifications for custom versions of JDE templates are not reconciled with the J.D. Edwards parent template.

Object Management

By industry standards, an *object* is a self-sufficient entity that contains data as well as the structures and functions used to manipulate the data. An object is a reusable entity that is based on software *specifications*. A specification is a complete description of a system object. Each object has its own specification, which is stored on both the server and the workstation.

The system stores objects in the following two places:

- A central-storage server, where you keep your central objects. When a developer checks out a central object, it is replicated on the workstation and then converted back into a central object when the developer checks it in.
- Workstations and servers, which store run-time objects. Run-time objects are replicated objects that actually run the system.

The following conceptual information describes how objects are moved between the central objects location and the object destinations. For information about how to check objects in or out, add objects to a project, or get objects without checking them out, see *Object Management Workbench* in the *System Development Tools Guide*.

Understanding Object Movement

When you perform any of the following tasks, objects move between the central objects location and the object destination:

- Check objects in or out
- Add existing objects to a project
- Perform a get object from the Object Management Workbench program (P98220)
- Run the Workstation Installation program

  During the workstation installation, all objects and the J.D. Edwards Supported Local Database, which contains replicated data, are copied from the package to the workstation. The system copies objects in only those packages for which you included specifications. If the package does not include specifications, objects are not replicated; instead, they are installed on the workstation through just-in-time installation.

Unless otherwise noted, object movement is the same when you check objects in or out, add objects to a project, or perform a get object. The following table shows the objects and specifications that move when you check in, check out, add, or get each type of object:

Table (object type TBLE)	The following objects move: • Table and table event rule specifications • Source files (*.c) • Table header files (*.h) • Object files (*.obj) • Table event rule include files (*.hxx) The table header is not the same as the actual table that resides in a database. The table itself is created through Table Design Aid when you generate it. The Workstation Installation program copies this table to the workstation if the table is stored in the J.D. Edwards Supported Local Database.
Business view (object type BSVW)	Specifications move.
C business function (object type BSFN, source language C)	The following objects move: • Specifications • Source files (*.c) • Header files (*.h) • Object files (*.obj)
Business function data structure (object type DSTR)	Specifications move.
Embedded event rules	You cannot check out embedded event rules. Embedded event rules move when you check out the object in which the event rule is embedded. For example, if embedded event rules are attached to a table, interactive application, or batch application, when you move the table or application, the specifications for the embedded event rules move with it.
Business function event rule (object type BSFN, source language NER)	Business function event rules (object type BSFN) can be checked out. When business function event rules are checked out, the .h file moves to the include directory, the .c file moves to the source directory, the .obj moves to the obj directory, and the local specifications are updated. The following objects move: • Specifications • Source (*.c) • Header (*.h) • Object (*.obj)
Media object data structure (object type GT)	Data structure specifications move.

Interactive application (object type APPL)	The following specifications move: • Application • Form • Form data structure • Embedded event rules
Batch application (object type UBE)	Report and event rule specifications move. You must check in and check out the version separately from the report.

See Also

- *Object Management Workbench* in the *ERP 9.0 Development Tools Guide* for more information about checking objects in and out, adding objects to a project, and performing a get object

Replicating Data Dictionary Items

The data dictionary resides in relational tables on a server. This set of tables is the publisher data dictionary where you make all data dictionary changes that you want to replicate to servers and workstations. ERP 9.0 stores the publisher data dictionary in the following tables:

- Data Item Master (F9200)
- Data Field Display Text (F9202)
- Data Item Alpha Descriptions (F9203)
- Data Dictionary - Error Message Information (F9207)
- Data Field Specifications (F9210)
- Data Dictionary - Smart Fields (F9211)
- Media Object storage (F00165)

Understanding Files Created By the Build Process

This section describes the files that ERP 9.0 creates during the build process for the workstation and the various server platforms.

Workstation Build

Business function dynamic linked libraries (DLLs) on workstations are grouped by related business functions. This grouping limits the size and number of procedures that are contained in each DLL. Grouping prevents memory allocation errors and avoids the platform limitations that can occur when you export many procedures from the same DLL.

The production environment PD9/bin32 directory contains the DLLs that are created on the workstation. All of the business function source files are in the PD9/source directory.

Files Created by a Business Function Build

When you build a single business function through the Object Librarian, the Business Function Builder program uses the make (*.mak) file that you provided, builds the business functions into their respective DLLs, and creates the following files:

- Source file (*.c)
- Header file (*.h)
- Object file (*.obj)

You must use the jdecallobject API to call a business function from a business function.

Business function event rules create the following files:

- OBJNAME.c
- OBJNAME.h
- OBJNAME.obj

Table event rules create the following files:

- OBJNAME.c
- OBJNAME.hxx
- OBJNAME.obj

Server Package Builds

Server package builds are used to move path code objects from the deployment server to enterprise server platforms. Server package builds are initiated by creating either full or update packages during Package Assembly. (Partial packages are not allowed for server packages.) Once you have assembled the package, you must check the server checkbox for Package Definition, and select the relevant servers from the list of available servers in the screen that follows. Once package definition is complete and the package has been activated, highlight the package and select Submit Build from the Row menu to start the server package build. When the server package build has completed successfully, you can deploy the server package to activate path code objects.

To assemble a server package, use the foundation, database, and object information in package assembly to generate build information, specification files, and business function source for .c, .h, and .hxx files to place into a staging area assigned during package assembly. Once the server package build has generated these objects and placed them in the staging area, the system transfers the objects to each of the servers specified in the package definition. Server package build then directs the servers receiving the server packages to compile the business function source code and generate the corresponding business function DLLs.

UNIX Server Build

This topic describes the files that the system creates when it builds business functions on a Unix server.

Files Created by a Business Function Build

When building business functions, the following groups of source files are actually compiled:

- Business function event rules
- Table event rules
- C business function event rules

When building business functions, the following file types are supplied to the build process:

- Source files (.c)
- Header files (.h, .hxx)

When building business functions, the build process creates the following file types:

- Object files (.o)
- Make files (.mak)
- Shared libraries (.sl, .so)

Shared libraries for business functions, which are equivalent to a DLL for a Windows NT workstation, are consolidated. Therefore, one shared library is created for each parent DLL in the Object Librarian – Status Detail table (F9861). If you are creating custom business functions, use a custom parent DLL instead of one of the parent DLLs that J.D. Edwards provides.

Where Business Functions Are Stored

On UNIX platforms, shared libraries for business functions are grouped by related business functions. This grouping limits the size and number of procedures that are contained in each shared library. Grouping prevents memory allocation errors and avoids platform-specific limitations in the number of procedures that you can export per shared library. Subordinate to the environment PD7334 directory is a source directory. This source directory contains subdirectories for each shared library that is created on the enterprise server. The directory structure looks like the following:

```
PD7334

         source

            CAEC

            CALLBSFN

            CCORE

            CDESIGN

            CDIST
```

```
CFIN

CHRM

CMFG

JDBTRIG
```

Each subdirectory contains the business function source files that belong to the shared library. All shared libraries are installed in the PD7334/bin32 directory. The naming convention for the shared libraries is `lib`, followed by the name of the shared library subdirectory, followed by `.sl` (for HPUX) or `.so` (for AIX). For example, `libccore.sl`.

Specification Files

Specification files must be consolidated into individual files before you transfer them to a server for translation. A specification file is a collection of two files, grouped by the same prefix (RDATEXT, for example). The two files have different suffixes (.ddb for the data file, and .xdb for the index file), and together these files comprise a complete specification file.

Windows NT Server Build

This topic describes the files that the system creates when it builds business functions on a Windows server.

Files Created by a Business Function Build

When building business functions, the following groups of source files are actually compiled:

- Business function event rules
- Table event rules
- C business function event rules

When building business functions, the following file types are supplied to the build process:

- Source files (.c)
- Header files (.h, .hxx)

When building business functions, the build process creates the following file types:

- Object files (.o)
- Make files (.mak)
- DLLs (.dll)

Business function DLLs are consolidated, just as they are on the UNIX platform or workstation. Therefore, one shared library is created for each parent DLL in the Object Librarian – Status Detail table (F9861). If you are creating custom business functions, use a custom parent DLL instead of one of the parent DLLs that J.D. Edwards provides.

Where Business Functions Are Stored

On Windows NT platforms, business function DLLs are grouped by related business functions. This grouping limits the size and number of procedures that are contained in each DLL. Grouping also prevents memory allocation errors and avoids platform-specific limits of exported procedures per DLL.

Subordinate to the environment PD9 directory is a source directory. This source directory contains subdirectories for each DLL that is created on the enterprise server.

The directory structure looks like the following:

```
PD9

          source

             CAEC

             CALLBSFN

             CCORE

             CDESIGN

             CDIST

             CFIN

             CHRM

             CMFG

             JDBTRIG
```

Each subdirectory contains all of the business function source files that belong to the DLL. All DLLs are installed in the PD9\bin32 directory. They have the same name as the DLL subdirectories, except that they have the .dll suffix.

Specification Files

Specification files must be consolidated into individual files before you transfer them to a server for translation. A specification file is a collection of two files, grouped by the same prefix (RDATEXT, for example). The two files have different suffixes (.ddb for the data file, and .xdb for the index file), and together these files comprise a complete specification file.

AS/400 Server Build

This topic describes the files that the system creates when it builds business functions on an AS/400 server.

Files Created by a Business Function Build

When building business functions, the server package build creates the following file types in a library with the package name in the QSYS file system:

- *MODULES - Object files
- *USRSPC - User spaces hold information about which .c files each business function DLL contains
- *SRVPGM - Server programs are the DLLs on the AS/400
- *FILE – Contains only logs about compiled business functions

Where Business Function Source Members Are Stored

New with ERP 9.0, the business function source and header locations are standardized across all server platforms when performing server package builds. AS/400 business function source and headers are now transferred to the Integrated File System (IFS). Server package build transfers objects to the following subdirectories under the server package directory in the IFS for the AS/400.

```
PD9
      include
      pack
      source
            CAEC
            CALLBSFN
            CCORE
            CDESIGN
            CDIST
            .
      spec
      text
```

`PD9/include` – This is the location where .h and .hxx source files are located. These objects are taken from the server and built on.

`PD9/pack` – The pack directory contains compressed objects used for moving server packages among servers of the same platform.

`PD9/source` – Contains subdirectories that include the business function DLL names. Each subdirectory contains .c source for the business functions that are compiled and linked into the DLL.

`PD9/spec` – Contains specification files for the path code. A specification file is a collection of two files grouped by the same prefix (RDATEXT for example). The suffixes .xdb and .ddb describe a specification file.

`PD9/text` – Contains build text and status files (.txt and .sts) for business function DLLs and specification files. The text files contain information about the server package build transfer for business function DLL and specification files. The text files also contain build directives for creating business function DLLs. The status files for specification files indicate if a server package build was successful in transferring specs into the parent package ‘spec’ subdirectory. The status files for business function DLLs indicate which .c source files were successfully compiled and linked.

Note
After an ERP upgrade (for example, migrating from Xe to B9), existing AS/400 server path codes must be rebuilt with the server package build to avoid problems building server package updates and manually re-linking business functions using the LINKBSFN program.

Specification Files

Specification files must be consolidated into individual files before you transfer them to a server for translation. A specification file is a collection of two files, grouped by the same prefix (RDATEXT, for example). The two files have different suffixes (.ddb for the data file, and .xdb for the index file), and together these files comprise a complete specification file.

Correlating Replicated and Central Objects

The following table shows the correlation between replicated objects that are stored in specification tables and central objects that are stored in a relational database that has a binary large object (BLOB).

Replicated Object	Central Object and Description
DDTABL	Table Header table (F98710). Contains one record for each table.
DDCLMN	Table Columns table (F98711). Contains one record for each column in a table.
DDPKEYH	Primary Index Header table (F98712). Contains one record for each table index.
DDPKEYD	Primary Index Detail table (F98713). Contains one record for each column in an index.
BOBSPEC	Business View Specifications table (F98720). Contains one record for each business view.
GBRLINK	Event Rules – Link Table (F98740). Contains one record for each event that has event rules for applications, reports, or tables (Named Event Rule links are stored in the Business Function F9862).
GBRSPEC	Event Rules – Specifications Table (F98741). Contains one record for each line of event rules.
DSTMPL	Data Structure Templates table (F98743). Contains one record for each business function, processing option, form interconnection, report interconnection data structures.
FDATEXT	Forms Design Aid Text Information table (F98750). Contains override text for applications.

FDASPEC	Forms Design Aid Specification Information (F98751). One record for every column, grid line, button, hyperitem, control, and so on, in the application.
ASVRHDR	Forms Design Aid/Software Versions Repository Header Info. (F98752). One record for each application (if the application has processing options, that information is also stored in the record).
ASVRDTL	Forms Design Aid/Software Versions Repository Detail Info. (F98753). One record for each form (includes references to the data structures).
RDATEXT	Report Design Aid Text Information (F98760). Override text for batch reports.
RDASPEC	Report Design Aid Specification Info (F98761). One record for each section, column, sort, constant, and so on, in Batch Reports and Versions.
JDEBLC	JDEBLC – Behavior Information (F98762). One record for each function in BSFN.
CGTYPE	Stored in specification format only. Code Generator Form Types.
DDDICT: One record for each Data Dictionary item that has been just-in-time installed DDINDEX: obsolete in B73.2 - indexes for dictionary DDTEXT: Data Dictionary text	• F9200, Data Item Master • F9202, Data Field Display Text • F9203, Data Item Alpha Description • F9207, Data Dictionary - Error Message Information • F9210, Data Field Specifications (ERP 9.0)
NEXTID	Next ID Master table (F98701). Local record of next IDs assigned to each workstation.
GLBLTBL	Cache Information From Data Dictionary and Table Specifications. Contains runtime table and override information. Built dynamically the first time a table is used.
SMRTTMPL	Data structure required field information.

Performing Backups and Restoring Objects

You can back up development objects on workstations and servers as frequently as necessary.

J.D. Edwards recommends that you perform the following backups:

- Developer workstation backups: Developers at J.D. Edwards do not back up their ERP 9.0 directory to the server because of server-space concerns. Instead, they check in the objects they are working on every eight hours, or however often needed to avoid rework.

Unless you have unlimited disk space on a file server to allow developers to back up their entire path code directory, you must use the check-in process as your backup method. If you follow the recommended development process, developers will know that they are allowed to check in unfinished or malfunctioning applications to the DEV path code.

- End-user workstation backups: End users should not have unreplicated data on their machines except store-and-forward transactions that have not yet been uploaded to the transaction tables. You should establish a policy that all store-and-forward users must upload their transactions before they leave work for the day. If a user is unable to perform this upload, he or she should back up the local database to a subdirectory on a file server. The name of the subdirectory should be the user's name.
- Development Server backups: At J.D. Edwards, the IT department backs up both the development file server (normally the deployment server) and necessary databases (central objects, Object Librarian, and data dictionary). When a developer needs to restore a particular object from backups, a database administrator restores the export to a path code called Restore. The developer checks out the object from Restore, ensures that the object functions as expected, and checks the object into the normal development path code.
- Deployment Server backups: In most cases, you do not need to back up the entire server nightly. However, under certain conditions, you might need to back up the following directories nightly:
  - The DEV path code, if you are modifying objects, building new packages, or updating the database that is delivered during a workstation installation.
  - HELPS, if you are modifying help files.
  - MEDIA OBJ, if your media objects reside on the deployment server. ERP 9.0 data sources in Oracle or SQL Server should be backed up nightly if your system data or any other important data is stored on the deployment server.
- Enterprise Server backups:
  - Back up the DBMS nightly. J.D. Edwards recommends that you use the backup tool that your database vendor provides.
  - Back up ERP 9.0 objects by backing up the entire ERP 9.0 directory. Also back up the PROD and DEV path codes and the JDE.INI file. Path codes are updated when the Version Control administrator deploys an object that was modified by developers who are authorized to access the Server Package application, and by end users who create new batch versions that will be executed on the server.

See Also

- *Server and Workstation Administration Guide* for more information about backing up and restoring

Copying Records Between Data Sources

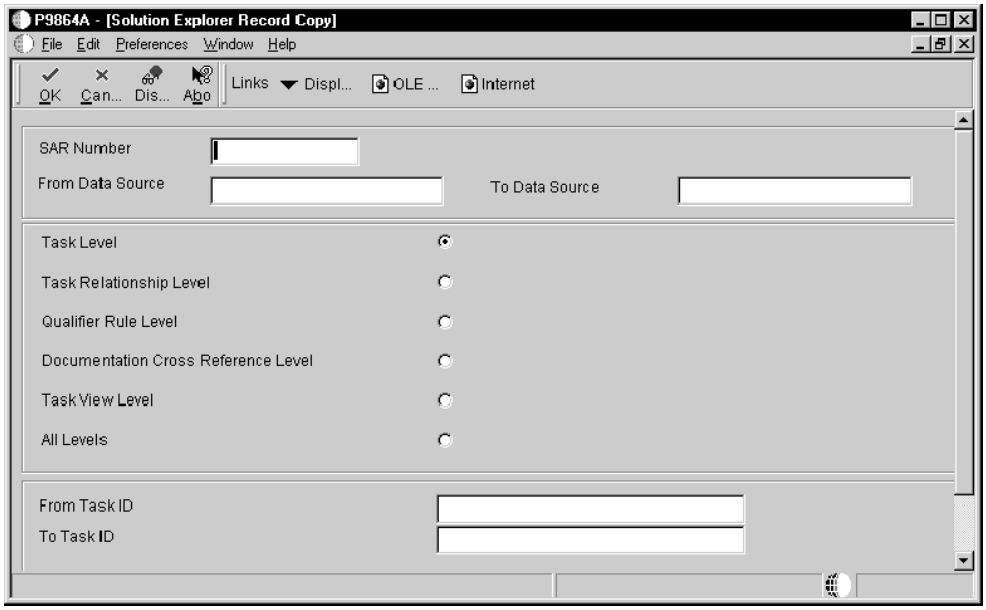
For ERP 9.0, you use the Object Management Workbench program (P98220) to copy control table records from one data source to another. However, to copy Solution Explorer records, you must use the Solution Explorer Record Copy program (P9864A). For example, you might need a separate set of UDCs, menus, or data dictionaries for your production and CRP environments. In this case, you can use the Solution Explorer Record Copy program to copy each change that you make in the CRP environment to the production environment.

You can copy control table records in Solution Explorer from one data source to another data source. You can either copy all control table records, or you can copy the following records individually:

- Tasks
- Task relationships
- Qualifier rules
- Documentation indices
- Task views

► To copy records between data sources

From the Content Management task view, choose Solution Explorer Record Copy (P9864A).



1. On Solution Explorer Record Copy, complete the following fields:

   - SAR Number

     Type the SAR number for the group of tasks that you want to move. If the tasks are not associated with a SAR, type an arbitrary number in this field.

   - From Data Source

     Type the data source from which you are copying the records.

   - To Data Source

     Type the data source to which you are copying the records.

2. Click one of the following options and then complete the relevant fields that appear at the bottom of the form.

- Task Level

  Choose Task Level to transfer a single task or a range of tasks. ERP 9.0 transfers all tasks in the range between the From Task ID and To Task ID. If you want to transfer a single task, enter the task ID in both the From and To fields.

  When you copy Solution Explorer tasks from the Task Level, the records are copied into all task relationships and task views in which they appear in the source data source.

- Task Relationship Level

  Choose Task Relationship Level to transfer tasks that are associated with a parent task (task node). Complete the following fields to transfer tasks at this level:

  - Parent Task -Type the parent task of the node or task that you want to transfer.
  - Child Task - Type a child task. If the child task is a node, ERP 9.0 transfers the node and all of its children.

  c. Task View - Type the task view for which you want to copy this task relationship. ERP 9.0 copies the task relationship only to the task view that you specify, even if this task relationship exists in other task views.

- Qualifier Rule Level

  Choose Qualifier Rule Level to transfer one or more qualifier rules. ERP 9.0 transfers all qualifier rules within the range that you specify.

- Documentation Cross Reference Level

  Choose Documentation Cross Reference Level to transfer documentation index records that are associated with task IDs. Type a range of task IDs to transfer a range of documentation indexes. ERP 9.0 transfers all indexes within the range that you specify.

- Task View Level

  Choose Task View Level to transfer all tasks, task relationships, qualifier rules, and documentation indexes that are associated with a task view. When you transfer these records within a given task view, ERP 9.0 displays only those tasks within the view that you specify.

- All Levels

  Choose All Levels to transfer all Solution Explorer tasks, task relationships, qualifier rules, documentation indexes, and variants. Check Clear To Data Source if you want to delete the contents in the target data source before you copy the new records from the source data source.

Package Build

As ERP 9.0 applications, business functions, and other objects change, you need to make those changes available to users within your enterprise. You might also need to set up a new workstation with ERP 9.0 software.

Packages allow you to deploy software changes and new applications to your users, or to install ERP 9.0 on a workstation for the first time. After you have defined and built a package, you can deploy it using the Client Workstation Installation program or Package Deployment program (P9631).

Understanding Packages and Deployment

You may need to update or set up a workstation or an enterprise, logic, or application server with ERP 9.0 software for a variety of reasons, such as:

- You want to set up a workstation for a new ERP 9.0 user or group.
- You need to deploy custom solutions to all users or only to selected users.
- You have created a new path code for development purposes, and you need to deploy it.
- You need to rapidly deploy a software fix to a selected group of affected users.
- Disk space is getting low on some of your workstations, and you need to create a minimum configuration of ERP 9.0. (To avoid this situation, J.D. Edwards recommends that all production users use partial packages.)
- You need to update your servers with custom modifications that you have developed using the ERP 9.0 toolset.

J.D. Edwards provides solutions to meet all of these needs. First, ERP 9.0 allows you to create a package that describes the location of the components that need to be distributed to your workstations. In order to deploy these components, you must define and build a package.

ERP 9.0 gives you the flexibility to choose between three different package types: full, partial, and update. Partial packages are available only for workstations. The package type that you choose depends on your needs.

After you have defined and built your package, it is ready for distribution. Depending on the package type, you can deploy packages through the Client Workstation Installation program or the Package Deployment program (P9631).

Package Deployment enables you to specify the workstations and servers that receive the package, as well as when the package is made available. Packages can be deployed to all computers within the enterprise, a select group of computers, or individual computers. You can schedule a package to be *pushed* from the deployment server to workstations. Push installation requires no interaction with the workstation users.

Understanding Full and Partial Workstation Configurations

Before you begin assembling and building packages, you should understand the two types of workstation configurations that are available in ERP 9.0: full and partial. These terms refer mainly to the amount of disk space that the installed package requires on the workstation. Understanding these two configurations helps you determine the suitable package type (full or partial) for the workstations in your enterprise.

A full workstation configuration requires a larger amount of disk space, and installation times can be longer. A typical full workstation installation takes more than 1.4 GB of disk space and can take 10 to 30 minutes to install, depending on network traffic.

A partial workstation configuration represents a minimum configuration of ERP 9.0 and requires significantly less disk space compared to a full workstation configuration. ERP 9.0 installation times are also dramatically faster for a partial workstation configuration. A typical partial workstation configuration requires approximately 165 MB of disk space and takes 3 to 10 minutes to install, depending on network traffic. However, because ERP 9.0 adds applications to the minimum configuration as the user needs them, users with WAN access might experience unacceptably slow performance with a partial workstation configuration.

The main difference between the full and partial workstation configurations is the amount of specifications that reside on the workstation. A full workstation configuration contains the full suite of ERP 9.0 applications, including those that the user rarely or never uses. The benefit of a full workstation configuration is that all applications are available immediately. The disadvantage is that it requires a large amount of disk space on a workstation.

J.D. Edwards recommends that production users install partial packages because partial packages are much easier to administer and maintain. Production users should install full packages only if the performance time for just-in-time installation is unacceptable on your network's configuration.

J.D. Edwards recommends that development users install full packages because full packages contain development objects that are not included in partial packages.

Both full and partial workstation configurations have advantages, and you must evaluate your needs before you determine which configuration is best for your enterprise. The following table summarizes the benefits and disadvantages of each configuration:

Full Workstation

Advantages:

- This configuration includes the full suite of ERP 9.0 applications (all specifications and foundations). Developers have local access to all ERP 9.0 objects.
- After initial client workstation installation, network traffic is lower than with a partial workstation configuration because all applications are installed at once.

Disadvantages:

- This configuration requires a larger amount of disk space on the workstation than the partial workstation configuration.
- Initial installation time is much longer than it is for a partial workstation configuration.

Partial Workstation

Advantages:

- This configuration requires much less disk space on the workstation than a full workstation configuration.
- Initial installation times are much faster for a partial workstation configuration.
- Only those objects that are required to begin using ERP 9.0 are deployed during client workstation installation. All other applications are delivered through just-in-time installation when the user first enters an application.

Disadvantages:

- Network traffic increases slightly when first-time users load applications.
- Performance for just-in-time installation over a WAN might be slower than over a LAN.
- Developers cannot use partial packages because development objects are not delivered.

Understanding Just-in-Time Installation

A partial workstation doesn't have any applications that reside on it. Instead, the system retrieves each application at runtime and loads it on the workstation the first time that you select that application from the ERP 9.0 Explorer menu. Loading happens only once; the next time that you select the same application, it is still loaded on the workstation. This process is called *just-in-time installation* (JITI). JITI applies only to applications. Business functions cannot be installed through JITI.

JITI works both when no applications are installed and when your workstation receives a partial or update package that does not contain specifications. Update and partial packages that contain specifications do not require the JITI process.

When you choose an option from a menu in ERP 9.0 Explorer, the runtime engine determines whether application specifications reside on your workstation. If local specifications do not exist, the just-in-time installation flag is set to Y in the Environment Master program (P0094), and your security profile allows the application to be installed, the specifications are transferred from the central objects data source for the path code that is associated with your environment. In this way, needed applications are installed just-in-time, rather than when the user receives the package or during the initial installation time.

The process for installing an update package is almost the same. When you receive an update package that contains a changed application without the new specifications, ERP 9.0 first determines whether that specifications for the application reside on your workstation. If so, it deletes from your workstation old versions of that application. Then, the next time that you select that application from the ERP 9.0 Explorer menu, ERP 9.0 loads the new version of that application.

When a package includes applications without specifications, at the time of execution only the following related objects are deployed:

- Interactive or batch application specifications
- Imbedded event rules for the application
- Processing option templates, data structures, and related business views

The following objects are not deployed, and, therefore, must be included in the package if they have been modified:

- Business functions and their data structures
- Generic text data structures
- Table event rules, which are included with tables
- Named event rules
- New icons

Understanding Package Types

After you determine whether you want to have a full or partial workstation configuration, you can build the following types of packages:

- Full
- Partial
- Update

Full Packages

Full packages are static, point-in-time snapshots of the central objects for the path code on which the package is based. A full package contains everything developers need to run to develop in ERP 9.0. Specifically, a full package includes a full set of TAM specification files, a full set of DLLs, all of the .c and .h files for business functions and tables, and an INF file that defines where the foundation, data, helps, and packages are located. Select this package type when you want to create a full workstation configuration.

The main advantage of a full package is that every ERP 9.0 application for which users are licensed is available to them. Because specifications reside on the workstation, information is processed locally, which eliminates network traffic. In contrast, a partial package requires the just-in-time installation process to download objects when they are used the first time. When you deploy a full package, a few specifications and tables might be installed just-in-time but the impact on network performance is insignificant.

Full packages are primarily for initial ERP 9.0 installations and are normally deployed via the Client Workstation Installation program. You can also use the Package Deployment program (P9631) to install a full package on a computer on which ERP 9.0 is already installed.

Full packages can also include development objects such as business function source files, object files, and header files. A user who needs to load development objects has the option to do so at deployment time.

Partial Packages

Partial packages provide a minimum configuration of ERP 9.0. Use partial packages when you want to create a partial workstation configuration. A partial package essentially contains only the specifications that allow users to launch ERP 9.0 Explorer. Specifically, a partial package starts with an empty set of TAM files, and then the system adds the specifications for the objects that are included in the package. Also included in a partial package is a full set of DLLs and an INF file that defines where the data, help files, and package objects are located.

Applications are loaded through just-in-time installation the first time that the user selects an application from the ERP 9.0 Explorer menu. The selected application is loaded from the central objects data source for that path code.

Like full packages, partial packages are primarily for initial ERP 9.0 installations and are normally deployed via the Client Workstation Installation application, although you can use Package Deployment to install a partial package on a computer on which ERP 9.0 is already installed.

Update Packages

An update package enables you to update, add to, or refresh an existing full or partial package with changed objects such as a database, application objects, or help files. This package type is well suited for quickly deploying software changes and corrections.

Unless a package includes applications that do not have specifications, the update package is a point-in-time copy of your central objects for a particular path code. If the update package contains an application without specifications, only that application is *dynamic*; the rest of the package is static. Dynamic means that the system copies the applications directly from the central objects data source at the moment that the user first selects the application.

After a user receives an update package and then signs on to ERP 9.0, the system loads the user's workstation with all objects in the package. The objects in an update package replace those same objects on the workstation. All other objects on the workstation remain the some.

Applications in a package can be secured through Security Workbench, which verifies whether the user has permission to install an application. Only those users who are authorized to load the application are able to do so. Therefore, your update package can contain a varying assortment of applications that apply to several diverse users or groups, but only the authorized users or groups can receive the applications that apply to them.

Like full packages, update packages can include development objects such as business function source files, object files, and header files. Update package recipients have the option to load development objects at deployment time.

Because of the way just-in-time installation works, performance across a WAN might be slow if the partial or update package contains only applications without specifications. To improve performance on the WAN, include in the package the specifications for each application.

All update packages require a corresponding parent package on which the update package is based. The parent package is a full or partial package that is updated by update packages. All objects in the update package are merged into the parent package.

Business function objects in the update package are linked to the corresponding objects in the parent package, and new DLLs are created. Similarly, specifications from the update package are merged into the specifications in the parent package.

The parent package concept applies to both workstations and servers. Parent packages for workstations reside on the deployment server, while server parent packages are kept in the build area for the enterprise server.

Understanding Features

In addition to ERP 9.0 objects, you can also add a *feature* to a package. A feature is a set of files or configuration options that must be copied to a workstation or server to support an ERP 9.0 application or another ERP 9.0 function. Like ERP 9.0 objects, features are included

in a package and deployed to the workstations and servers that require the feature components.

For example, you might need to add to a package ActiveX controls, a J.D. Edwards Supported Local Database for the Sales Force Automation feature, ODBC data sources for use with Open Data Access, or Microsoft Windows registry settings.

You define a feature by using the Package Deployment program (P9631). You can then add the feature to a package by using the Package Assembly program (P9601) and the Package Build program (P9621).

Overview of Creating and Deploying a Package

The following is an overview of the steps for creating and deploying a package:

4. Assemble the package.

   During this step, you specify the type of package that you are building and provide a name, path code, and package description. Next, you assemble your package by specifying the objects that you want to include in the package. If you are building a partial or update package, you can specify individual objects to include.

   To simplify the process of assembling a package, the Package Assembly program (P9601) includes the Package Assembly Director, which displays a series of forms that guide you through the steps of naming your package and assembling the objects that you want to include in the package.

5. Define the package build.

   After you assemble the package, you must define the build before you can deploy the package to your workstations and servers. In this step, you specify the following:

   - Build options
   - Build specification options
   - Business functions build options
   - Compression options
   - Build features options

   You also need to specify whether the package is for a workstation, a server, or both. If the package is for servers, you must specify which servers should receive the package.

   Again, to simplify the build process, the Package Build program (P9621) includes the Package Build Definition Director, which displays a series of forms that guide you through the steps of specifying where to build the package, whether to include specifications, whether to compress or build business functions, and so on.

6. Build the package.

   During the actual build process, ERP 9.0 takes the information that you provided when you assembled and defined the package and copies and converts central objects to the package. It also globally builds the business functions that are included in the package and then compresses the package.

7. Schedule the package for deployment to workstations.

If you build an update package (or if you want to redeploy a full or partial package to a workstation that already has ERP 9.0), you must specify the date and time to deploy the package. When you schedule the package, you can indicate whether package installation is mandatory or optional.

At this same time, you can specify whether you want the package to be deployed via push installation, which requires no interaction with the package recipient

Deploy the package to deployment and enterprise servers.

Use the Package Deployment program (P9631) to move any changed objects to the enterprise server.

If you specify a server during the package build definition process, ERP 9.0 automatically creates a corresponding server package in the correct format. If you do not specify a server and define only a workstation package, you should create a corresponding server package. The process is nearly identical to creating a workstation package.

See Also

- *Adding Features to a Package* in the *Package Management Guide* for information about adding features
- *Assembling Packages* in the *Package Management Guide* for details about how to assemble a package
- *Defining Package Builds* in the *Package Management Guide* for information about how to define a package build
- *Using Push Installation* in the *Package Management Guide* for information about push installation
- *Deploying Server Packages* in the *Package Management Guide* for information about deploying packages

How ERP 9.0 Builds a Full Package

The following is an overview of how ERP 9.0 builds a full package:

3. Creates the package build directories.
4. Creates the INF file.
5. Copies the following directories and files from the check-in location to the package name directory:
    - res
    - source (.c files)
    - include (.h files)
    - work
    - make
    - bin32
    - lib32
    - object (.obj files)

Note
If you choose to build business functions with the package build, ERP 9.0 does not copy bin32, lib32, and object (.obj) files because the BusBuild program creates them.

6. Builds the table access management (TAM) specification files from the information in the relational database.
7. Runs the BusBuild program to compile and link the business functions that create the DLLs in the bin32 directory, the objects in the obj directory, and the libraries in the lib32 directory.
8. Compresses the directories.

How ERP 9.0 Builds a Partial Package

The following is an overview of how ERP 9.0 builds a partial package:

1. Creates the package build directories.
2. Creates the INF file.
3. Builds an empty set of TAM specification files.
4. Copies the following directories and files from the check-in location to the package name directory:
   - res
   - work
   - make
   - bin32
   - lib32

Note
If you choose to build business functions with the package build, ERP 9.0 does not copy bin32 and lib32 files because the BusBuild program creates them.

5. Builds the specifications for the objects in the *LITE record.
6. Runs the BusBuild program to compile and link the business functions that create the DLLs in the bin32 directory, the objects in the obj directory, and the libraries in the lib32 directory. (Optional)

Note
If you do not run the BusBuild program when you build the partial package, you must run it separately.

7. Compresses the directories (optional).

How ERP 9.0 Builds an Update Package

The following is an overview of how ERP 9.0 builds an update package.

1. Creates the package build directories.
2. Creates the INF file.
3. For each object in the Software Package Detail table (F9631), retrieves the information from the relational database and adds it to the TAM specification files.
4. Runs the BusBuild program to update the DLLs in the bin32 directory, the objects in the obj directory, and the libraries in the lib32 directory.

Understanding Server Packages

All ERP 9.0 application development takes place on workstations. The development objects themselves are stored on a single deployment server and are managed by the Object Management Workbench. ERP 9.0 allows you to partition business applications to an enterprise server. To ensure that modifications and enhancements that are developed on the workstation are reflected on the server, you must build a server package that contains those modifications and enhancements.

Through the same Package Assembly (P9601) and Package Build (P9621) programs that enable you to create workstation packages, you can assemble, define, and build server packages that push objects from the central objects location to enterprise servers. A server package is a group of specification records, source files, and header files. Compiled objects are created on the enterprise servers. After defining and building a server package, you can install it to enterprise, logic, or application servers by using the Package Deployment program (P9631).

In a development environment, developers can create server packages whenever they create or modify an object and need to transfer that object to the enterprise server. In a production environment, an ERP 9.0 system administrator should create server packages.

A server package is essentially the same as a client workstation package, with the following exceptions:

- Server packages do not include ERP 9.0 help information or a J.D. Edwards Supported Local Database.
- Foundation code is not typically deployed as part of a server package.
- Some sets of TAM specifications, such as those used in form design (that is, interactive engine specifications), are not used on servers, and therefore are not included in a server package.
- Some business functions are not built on the server, and therefore are not included in a server package.
- Partial packages are not available for servers.

Under the following circumstances, you should create a separate server package that corresponds to the workstation package:

- When business functions are not compatible across platforms. For example, a business function that is compiled on a Windows NT workstation will not run on a UNIX server. Also, not all business functions are compiled on the server.

- When data alignment is different across platforms. TAM files that are built on the workstation will run on the server only if you convert them. Also, integer representation (the way numbers are recognized and identified) is different on servers than on the workstation.
- When ASCII to EBCDIC conversion must take place for the AS/400.

In addition, server platforms are not always compatible. For example, each enterprise server platform has its own internal storage for specifications, which are not compatible across platforms. Also, each enterprise server platform uses different compilers and has different object representation, so business functions are not compatible across platforms.

Understanding the Server Package Build Process

Although creating a server package is identical to creating a client workstation package, you create them for different purposes. The main purpose for building a server package is to enable ERP 9.0 administrators to build TAM specifications and business functions on the enterprise servers.

The Package Build program (P9621) provides the following benefits for building server package builds:

- Provides complete integration with client workstation packages
- Allows administrators to build a package on multiple servers simultaneously
- Enables administrators to build individual package components simultaneously on the server
- Allows you to build a package on one enterprise server and deploy to another server of the same type
- Creates history records that enable monitoring from the client workstation, so that the administrator can determine which packages have been built
- Creates compressed files and loads them onto the deployment server for easier mastering to a CD
- Supports both full and update packages
- Provides restart capabilities for packages that do not build successfully

You can install the following objects on an enterprise server:

- Business functions
- Business views
- Data structures
- Tables (installation does not create the table in a database; it only pushes the specifications and table header files to the server)
- Batch application specifications (both templates and versions)
- Application specification records

Because server packages are assembled and defined in the same way as client workstation packages, you can assemble a server package, using the Package Assembly program (P9601), and build the server package, using the Package Build program (P9621), at the same time that you assemble and build client workstation packages.

After you assemble the server package and define the package build, the following events occur:

1. After the build process starts, ERP 9.0 creates packed TAM files.
    2. On the local server, ERP 9.0 submits a batch application for the server package.
    3. This batch application calls a business function, which, in turn, calls the server package engine.
    4. The build engine uses the records that the Package Assembly and Package Build programs create to do the following:
        - Initialize directories on each server.
        - Begin transferring packed TAM files. As each packed file is transferred, an unpack process starts on the server.
        - Transfer all business function source and header files to the server. At this time, the build engine reads the Object Librarian Master Table (F9860) to determine the DLL in which each module belongs. For the JDBTRIG library, a special function is called to direct the trigger library in which the module belongs. In this case, the system does not use the Object Librarian Master Table .
        - Start a build master process on the server when the source files for all business functions are transferred. This build master starts one or several individual build processes simultaneously. Each DLL has its own build process. The JDE.INI file indicates the number of processes that can run simultaneously.
        - Move the process to another server, if one was specified during the package build process. The process transfers and builds all components on that server.
        - Check the status of each build piece on each server after the build process has begun on all servers. History records are updated as the statuses change.
        - Compress the package components and transfer the compressed files on the deployment server when the building is complete. This happens only if you specify that the system compress the files when it builds the package. This process is repeated for all servers.

JDE.INI Settings for Server Package Builds

If your server package includes business functions, the BSFN BUILD section of the JDE.INI file applies to the package. The following example illustrates a typical entry for UNIX-based servers.

```
[BSFN BUILD]
BuildArea=/u17/i5745669/b73.3/appdev/packages
DebugFlags=-g -y D_DEBUG -DJDEDEBUG
InliningFlags=
DefineFlags=-DKERNEL -DPRODUCTION_VERSION -DNATURAL_ALIGNMENT
CompilerFlags=-Aa +w1 +z -c
OSReleaseLevel=+DAportable
LinkFlags=-b -z
LinkLibraries=
SimultaneousBuilds=0
DoCompression=1
```

Setting	Value	Purpose
Build Area=	/usr/PeopleSoft/b9/packages	Indicates the location on the server where the package will be built.

Optimization Flags=	+02 (default for HP 9000) -02 (default for RS/6000 and Sun)	Varies, depending on hardware. The system uses these compile flags to build business functions in Release mode. You should not change these flags.
DebugFlags=	-g -y -D_DEBUG _DJDEDEBUG (default for HP 9000) -g -qfulpath -qdbextra -D_DEBUG -DJDEDEBUG (default for RS/6000) -g -D_DEBUG -DJDEDEBUG (default for Sun)	Varies, depending on hardware. The system uses these compile flags to build business functions in Debug mode. You should not change these flags.
InliningFlags=	blank (default)	Indicates whether the AS/400 uses inlining. Enter Yes to turn on inlining on the AS/400. Enter No to turn it off. This flag is blank for non-AS/400 servers.
DefineFlags=	-DKERNEL -DPRODUCTION_VERSION -DNATURAL_ALIGNMENT -D_HPUX-SOURCE (default for HP 9000) -DKERNEL -DPRODUCTION_VERSION -DNATURAL_ALIGNMENT (default for RS/6000) -DKERNEL -DPRODUCTION_VERSION -DNATURAL_ALIGNMENT -D_SUN-SOURCE (default for Sun)	Indicates the Kernel Production Version of the source for HP, RS, and SUN.
CompilerFlags=	-Aa +w1 +z -c (default for HP 9000) -qalign=natural -qflag=I:I -c (default for RS/6000) -qspill=1024 -misalign -KPIC (default for Sun)	Varies, depending on hardware. The spill flag sets the stack space when business functions are compiled. Typically, 1024 is adequate space to compile the delivered business functions.
OSReleaseLevel=	+DAportable -q32 (for AIX)	Indicates the release level for which you are compiling the package. You should not change these flags.
LinkFlags=	-b -z (default for HP 9000) -bl:/<your system directory>/bin32/functlist.imp -bM:SRE -bexpall -brtl -lc -bnentry -L. L/usr/<your system directory>/lib -ljdelib -ljdekrnl -ljdenet -bloadmap:loadmap (default for RS/6000) -G -L$(ORACLE_HOME)/lib (default for Sun)	Varies, depending on hardware. The system uses these flags to link business functions. You should not change these flags.
LinkLibraries=	blank (default)	Indicates the libraries to which business functions are linked. (Applies to Windows NT and AS/400 servers only.)

SimultaneousBuilds=	0 (unlimited) (default) any integer (number of simultaneous builds)	Indicates the number of DLLs that can be built at a time. Zero means that all will be built simultaneously.
Qname=	<queue name>	Applies to AS400 only. Specify a queue name if you want the jobs for building dlls to go to a queue other than the default queue.

See Also

- The *System Administration Guide* for complete information about the JDE.INI file

JDE.INI Settings for Workstations

Following are optional settings that you can enter in the jde.ini file for the workstation.

Building Specifications and Business Functions

If you build full or partial client packages that include both business functions and specifications, add the following setting to the [INSTALL] section of the jde.ini file on the computer that you use to build the packages:

```
WaitForBusbuild=Y
```

When you add this setting, ERP 9.0 builds the specifications and business functions sequentially instead of simultaneously, which can speed up the build process.

Compressing Server Packages

To compress packages that you build on the server, add the [BSFN BUILD] section to the jde.ini file on the workstation and create the following entry:

```
DoCompression=1
```

This setting compresses packages that you built on the server and deploys them to other servers of the same type, such as all AIX Unix servers or NT servers. If the server is of a different type, then you must build a package on that type of server.

When the server builds a compressed package, it stores the compressed files in subdirectories, such as \bin32, \specs, that are subordinate to the following path on the deployment server: \package*package name**server type*\, where *package name* is the name of the package, and *server type* is the type of server for which the package is compressed. The compression process creates a new file called compressed.inf in the *server type* directory. This file includes the information that the system needs to deploy the compressed files. The following table shows the type of compressed files that the process creates for each type of server:

NT	UNIX	AS/400
.cab	.z	SRVPG, SPECS, MODULE, USRSPC

When you deploy the package to another enterprise server, ERP 9.0 reads the compressed.inf file and uses this information to copy the compressed files from the package directory on the deployment server to the enterprise server.

Considerations for Users of the Sun Operating Systems

The Sun Solaris compiler expects a newline character at the end of every source code file that it compiles. If the compiler does not find this newline character, it rejects the line and displays a warning message. In some cases, this line rejection and message might cause the package build to fail.

If you develop custom modifications on Sun servers that use the Solaris operating system, you must ensure that this newline character is present in the compiled source code before you assemble, define, and build packages that contain your modifications. This step helps ensure that the package build process completes successfully.

In some cases, the system automatically adds the newline character, and you do not need to add it manually. If you edit source code in the UNIX environment using an editor such as VI or emacs, these editors automatically add the newline character. Also, all of the source code files for ERP 9.0 business functions include the newline character.

However, editors that are included with PC workstations typically do not add the newline character. Therefore, if you edit source code on a PC workstation and then transfer the file to the server for compiling, verify that the newline character exists in the source code.

Assembling Packages

The first step in building and deploying a package is to assemble the package, which entails selecting the type of package that you want to build, entering a package name and detailed description, and assembling the objects that you want to include in the package. The package name and description appear during workstation installation when the user chooses a package to install.

The Package Assembly Director, which you access from the Package Assembly program (P9601), steps you through the process. During package assembly, the build status is always either In Definition or Definition Complete. After you assemble the package, you can then define it.

Understanding the Package Assembly Director

The Package Assembly Director guides you through the process of specifying or confirming the location where package components can be found, as well as indicating the objects to include in the package. The director always gives you the option to either continue to the next form or go back to the previous form. Also, you can always cancel the assembly process.

The following table summarizes the function of each form in the Package Assembly Director:

Package Assembly Directory form

Use this form to review introductory information about the Package Assembly Director.

Package Information form

Use this form to enter the package name, description, and corresponding path code.

Package Type Selection form

Use this form to indicate whether you are creating a full, partial, or update package.

When you create an update package, you must also indicate the parent package on which the update package is based. For example, if you are creating a package to update your original partial package called APPL_B, you would enter APPL_B as the parent package for your update package.

You can also include object specifications in an update or partial package. If you do not include specifications, the application is installed through just-in-time installation the first time that a user selects that application. That is, the system automatically retrieves the application specifications from the central objects data source the first time that a user selects the application after receiving the package.

Foundation Component form

Use this form to enter the location of the ERP 9.0 foundation. A ERP 9.0 foundation is the code required to run all ERP 9.0 applications. It is required for all full and partial packages. If you do not specify a foundation path for full and partial packages, the system uses the default foundation path. Update packages use the foundation for the parent package unless you specify another foundation.

Help Component form

Use this form to enter the location of the help files. The help component determines where the system looks for the help files for the package. For full and partial packages, if you do not specify a help location, the system uses the default help file path. Update packages do not require help files.

Including a help component in a package does not mean that the help files are installed on the user workstation. The help component tells ERP 9.0 where to find the help files for the package.

Database Component form

Use this form to specify the location of the database to be included in the package. For full and partial packages, if you do not specify a database location, the system uses the default database path. Update packages do not require a database.

Default Object Component form (for full packages only)

Use this form to verify the deployment data source. When you build a full package, the system retrieves the objects that are included in the package from the deployment data source that is associated with the path code that you specified for the package.

Object Component form (For partial and update packages only)

Use this form to specify the individual objects that you want to include in the package. You can add any of the following objects:

- Interactive or batch applications
- Business function modules
- Business views
- Data structures
- Media object data structures
- Table definitions

Features Component form

Use this form to include features in your package. A feature is a set of files or configuration options, such as registry settings, that must be copied to a workstation

	or server to support a ERP 9.0 application or another feature of ERP 9.0.
Language Component form	Use this form to include in your package language specifications for a language other than English.
Package Component Revisions form	Use this form to review the information that you entered on the previous forms. You can modify any or all of your selections on this form.

Accepting Default Values

Many of the forms in the Package Assembly Director have default values and, after verifying that you want to use the default value, you can advance to the next form without entering anything.

Forms determine the default values based on the following:

- Foundation: The default foundation location is the server share path under the path code for the package.
- Help: The default help location is the server share path under the path code for the package.
- Database: The default database location is the server share path under the path code for the package.
- Objects: The default location for full packages is the deployment data source.
- Language: The default language is English.

On forms that have a default value, even if you change or clear the field you can always restore the original default value by clicking the Default button. The Form menu of the Package Component Revisions form also has a Set Default option that restores default values.

If you are building a full or partial package and do not need to specify the objects in that package, the fastest way to define the package is to accept the default locations for the foundation, database, help, and language. This method applies only to full and partial packages. For an update package, if you accept the defaults but do not include any objects, the system creates an empty package.

As you view the forms in the Package Assembly Director, you can accept the default selections by clicking Next. If necessary, you can always make changes at the final Package Component Revisions form.

Verifying a Path Code for Package Build

Before you assemble a package, you can verify that the path code from which the package is built is configured correctly. The verification process tests the environment, machines, and tables before a package is submitted. By verifying your environment, you eliminate the chance that your package build will fail due to configuration issues. This verification can save many hours in building a package.

During the verification process, the program verifies the following:

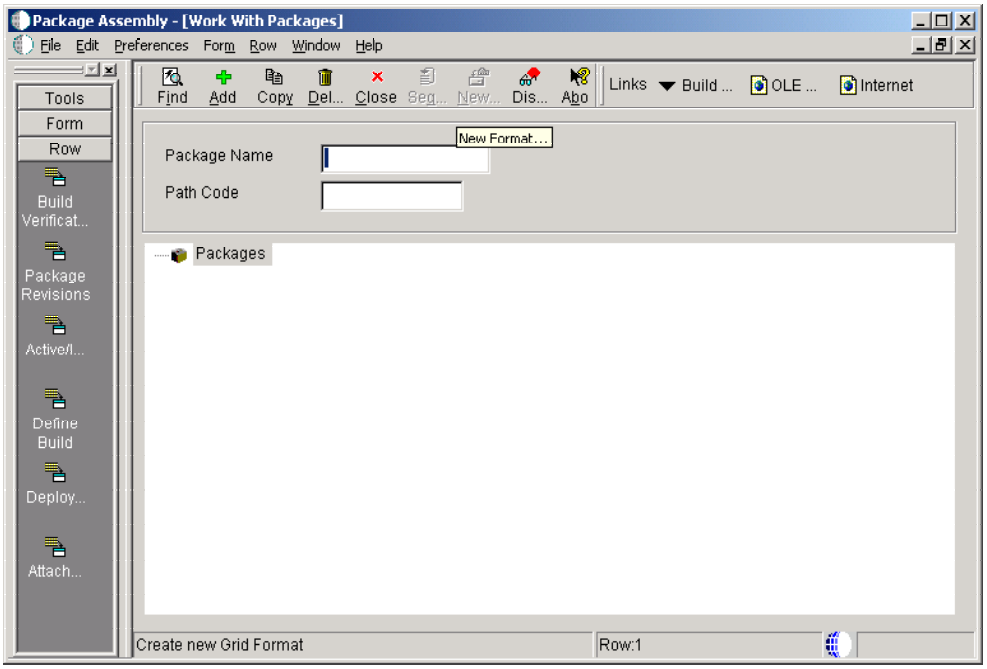
- Disk space is adequate
- Central objects and package build tables are accessible

- User has permissions to create directories on the deployment server and enterprise server
- Required service pack is installed
- Required MDAC is installed
- Machine tables are set up
- Required compiler version is installed
- Enterprise Server port is accessible
- Debug levels of the jde.ini files are adequate for the client and enterprise server

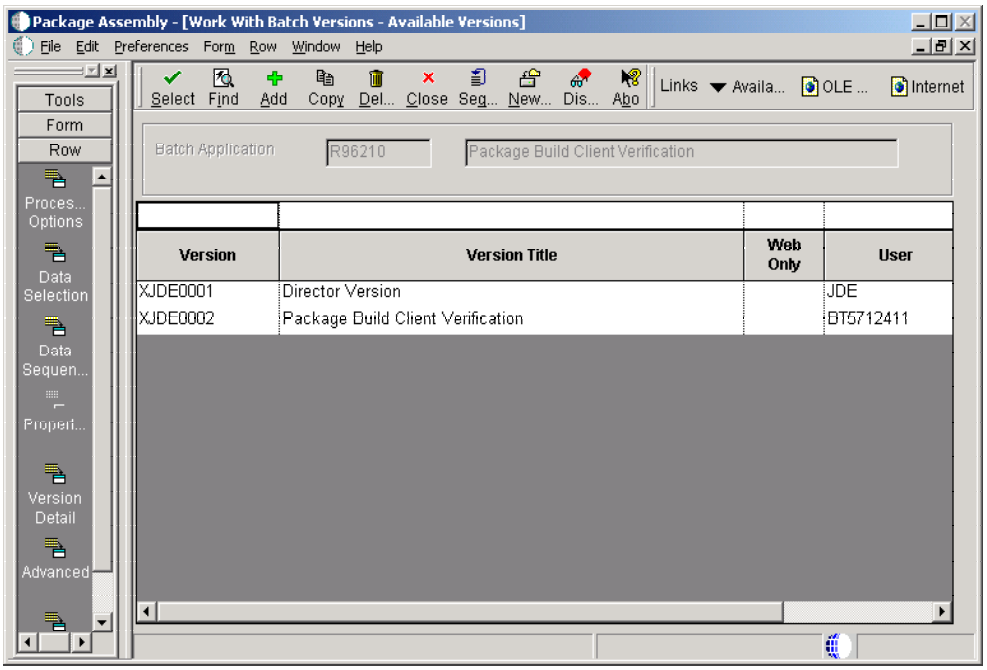
► To verify a path code for package build

From the Package and Deployment Tools menu (GH9083), choose Package Assembly.

The Work with Packages form appears.



1. From the Form Exit menu, choose Build Verification.



2. On the Work with Batch Versions – Available Versions form, click Select.
3. On Version Prompting, click Submit.
4. On Processing Options, complete the following fields:
   - 1. Indicate if this is a Client or Server

     Leave the field blank for a client, or type 1 for a server
   - Path Code

     Enter the Path Code that you want to verify
   - Server Name

     If the machine is a server, enter the server name
   - Compiler Flag

     Enter 1 if there is a compiler on the server
5. Click OK.
6. On Printer Selection, choose the desired printer, and click OK.

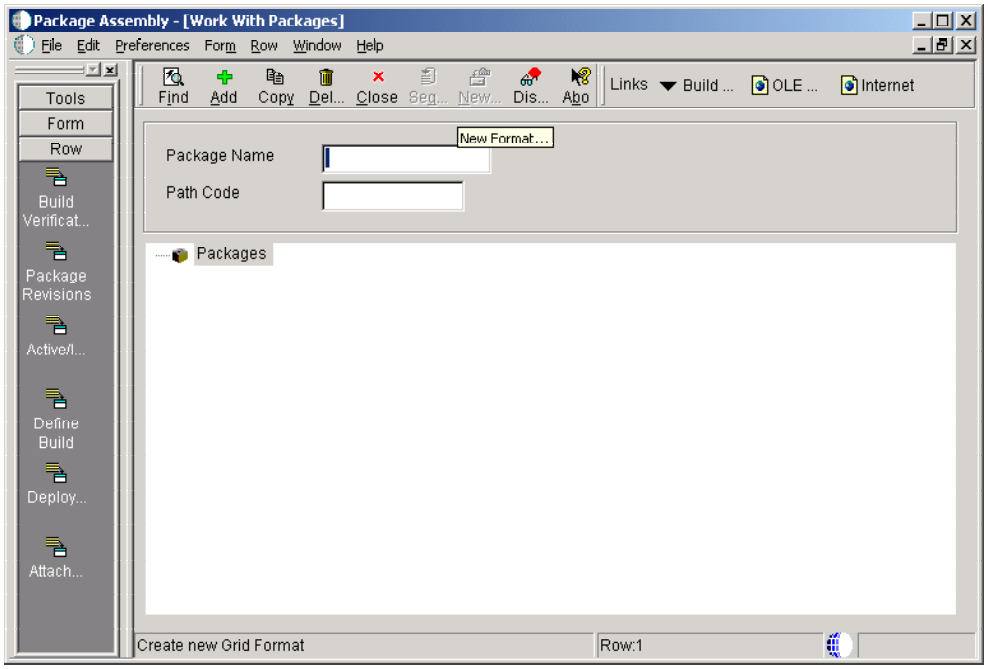
Running the Package Assembly Director

Complete this task to enter default information on each of the main forms of the Package Assembly Director. For details about customizing the information on each of these forms, refer to the relevant subsequent tasks.

▶ To run the Package Assembly Director

From the Package and Deployment Tools menu (GH9083), choose Package Assembly.

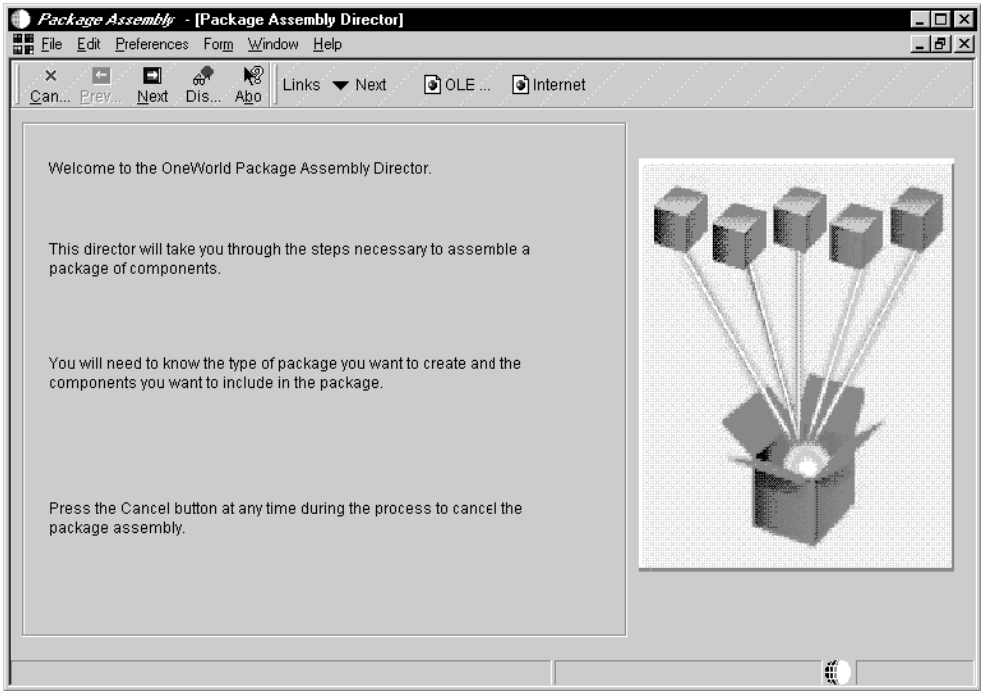
The Work with Packages form appears.



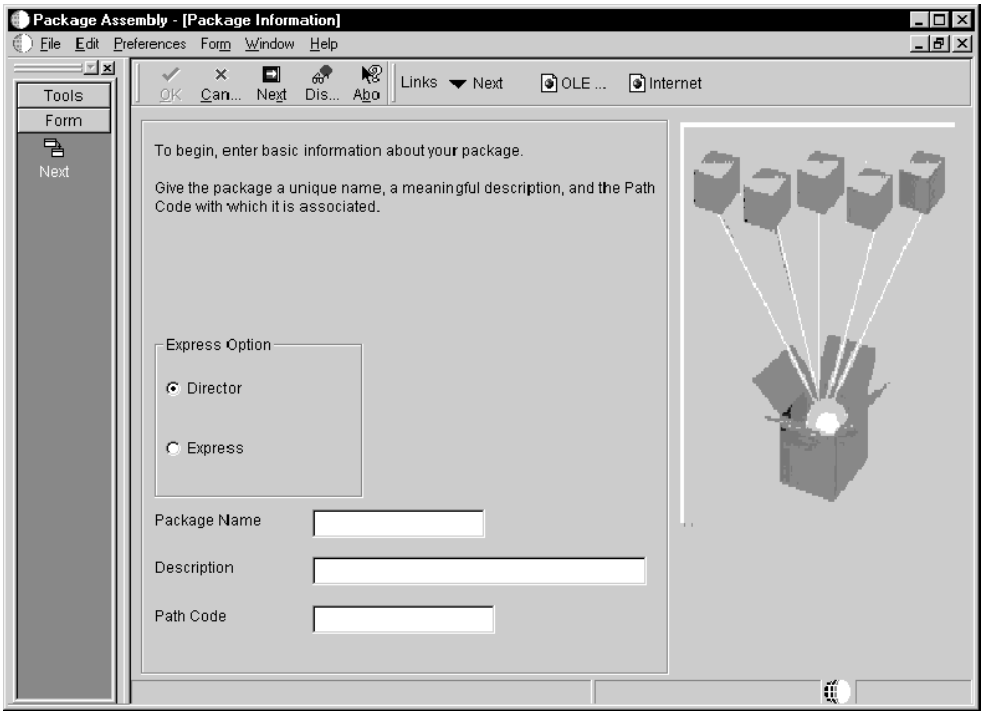
Any previously assembled packages appear on this form. As on any other parent/child form, you can expand or compress the list of packages by clicking the plus (+) or minus (–) symbols to show more or less information.

For any previously assembled packages, underneath the package name you can view the package properties (including package type and current status), as well as the selections for foundation, database, help, and language.

1. To assemble a new package, click Add.



2. On Package Assembly Director, click Next.



3. On Package Information, complete the following fields:
   - Package Name

- Description
- Path Code

Note
Starting with B9, you can build a single foundation package to deploy to all path codes by typing *ALL in this field. This option allows you to create an update package for service packs that can be installed to any pathcode. If you enter *ALL in the Path Code field, the application does not allow you to select, build, or deploy any objects (such as specs, help, business functions, and so on) in the package—only a foundation. The package is built to a directory called "all_packages" located under the release path (for example, B9/all_packages). This package can be deployed to any path code.

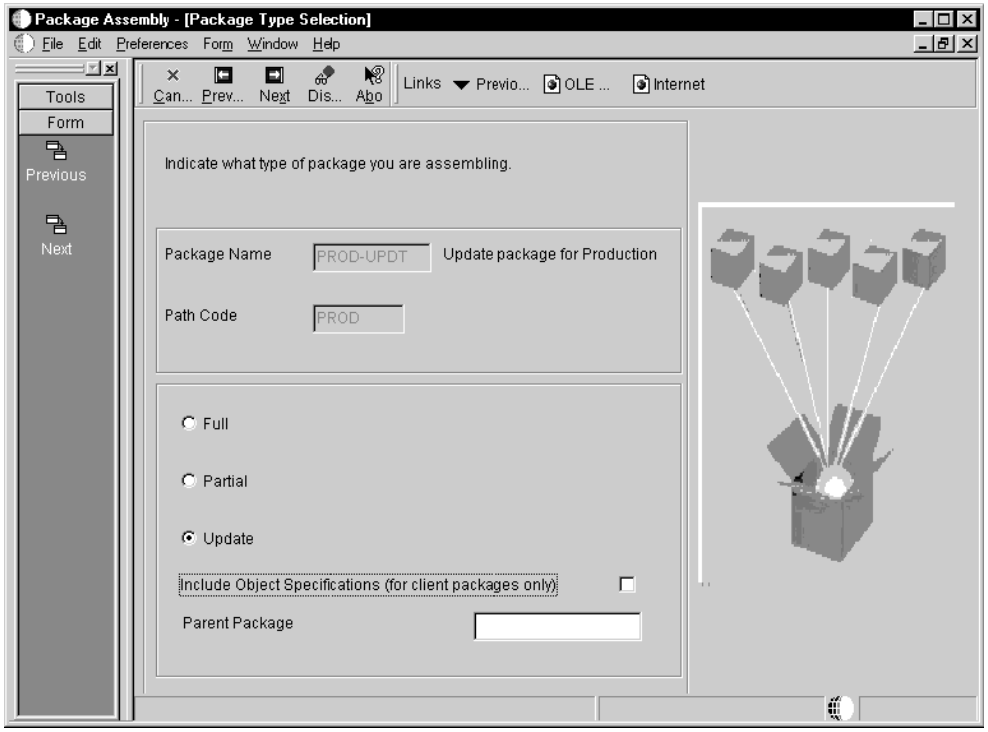
Before you can use this option, you must add an entry to the Path Code Master called *ALL. See the *Configurable Network Computing Implementation Guide* in the *Package Mangement Guide* for more information about how to set up this path code.

4. Click one of the following options:
   - Director

     To customize the package assembly, click this option and then complete the remaining steps for this task.

   - Express

     To accept default values for the package assembly, click this option and then continue with the task *Reviewing the Package Assembly Selections*. Otherwise, proceed to step 5.
5. Click Next.



On Package Type Selection, complete the following fields:

- Full, Partial, or Update

  If you are building a foundation package to the *ALL pathcode, the application automatically selects an Update package.

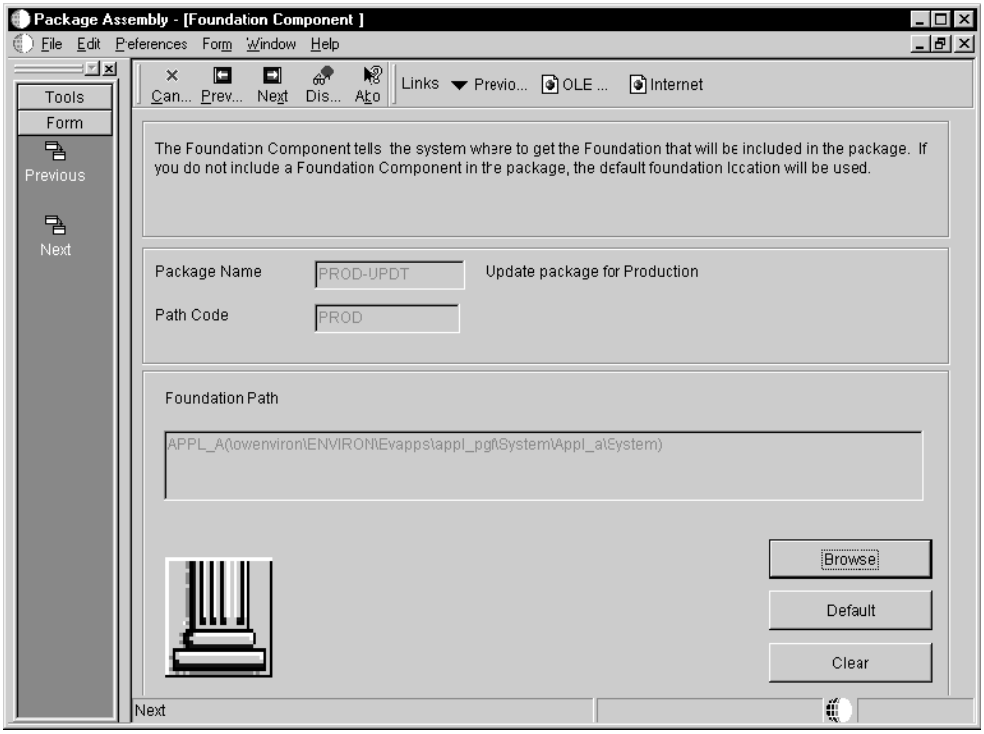
- Parent Package

  (for update packages only)

- Include Object Specifications

  (for client packages only)

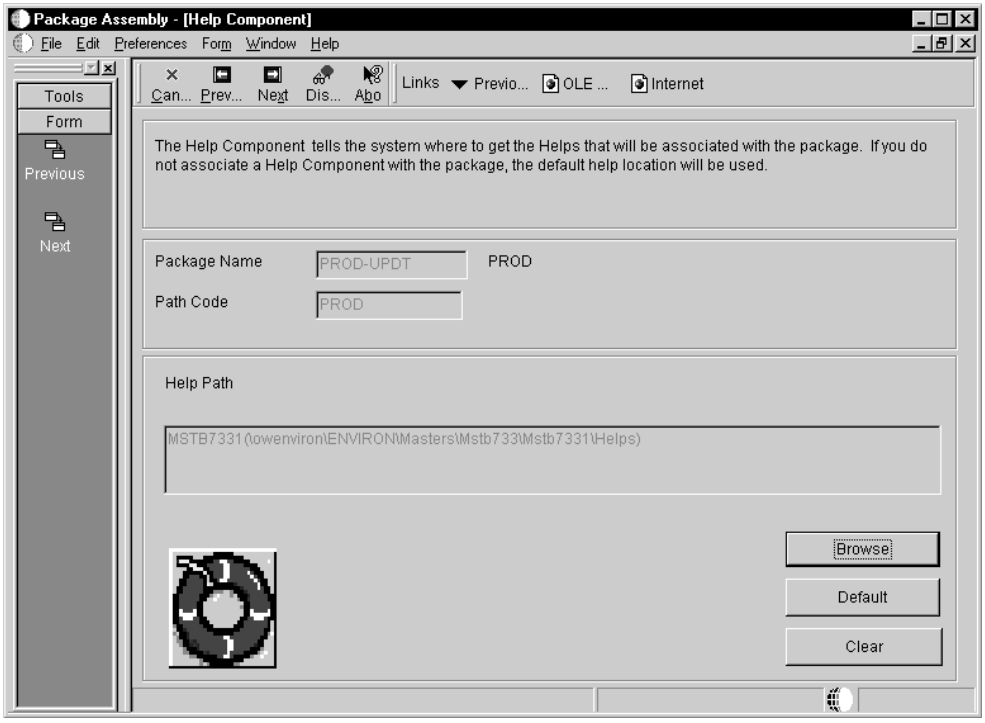
6. Click Next.



7. For full and partial packages, accept the default location by clicking Next, or click Browse to specify another foundation location.

   For more information about entering a foundation, see *Entering a Foundation Location* in the *Package Management Guide*.

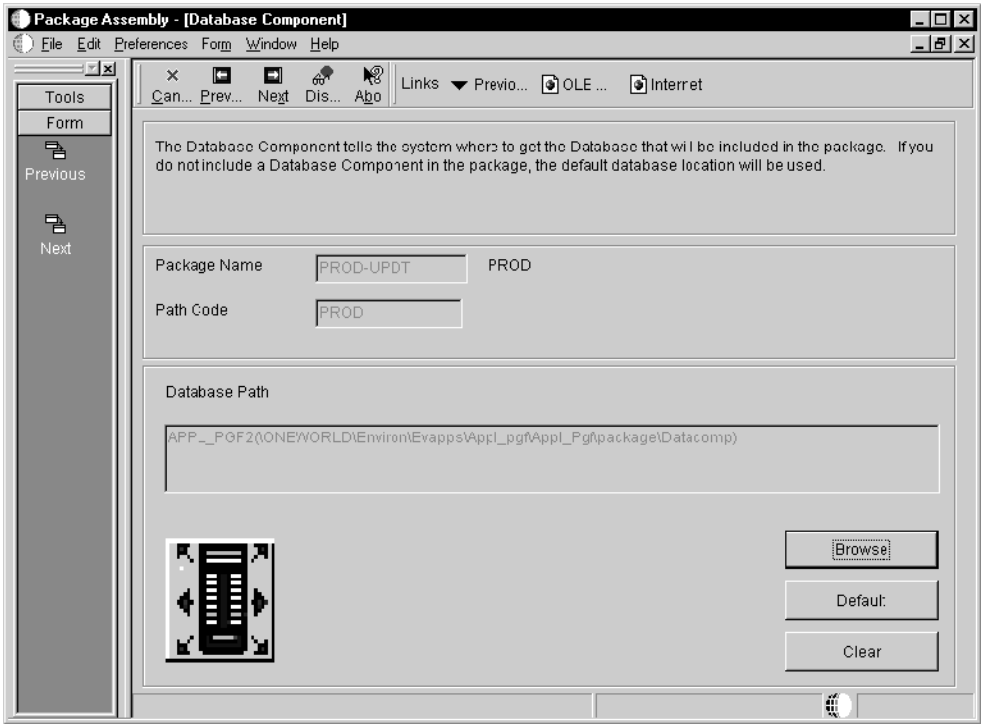
8. For an update, click Clear unless the package includes a Foundation.
9. Click Next.



10. For full and partial packages, accept the default location by clicking Next, or click Browse to specify another help location.

    For more information about entering a help location, see *Entering a Help File Location*.

11. For an update, click Clear unless the package includes a Foundation.
12. Click Next.



13. For full and partial packages, accept the default location by clicking Next, or click Browse to specify another database location.

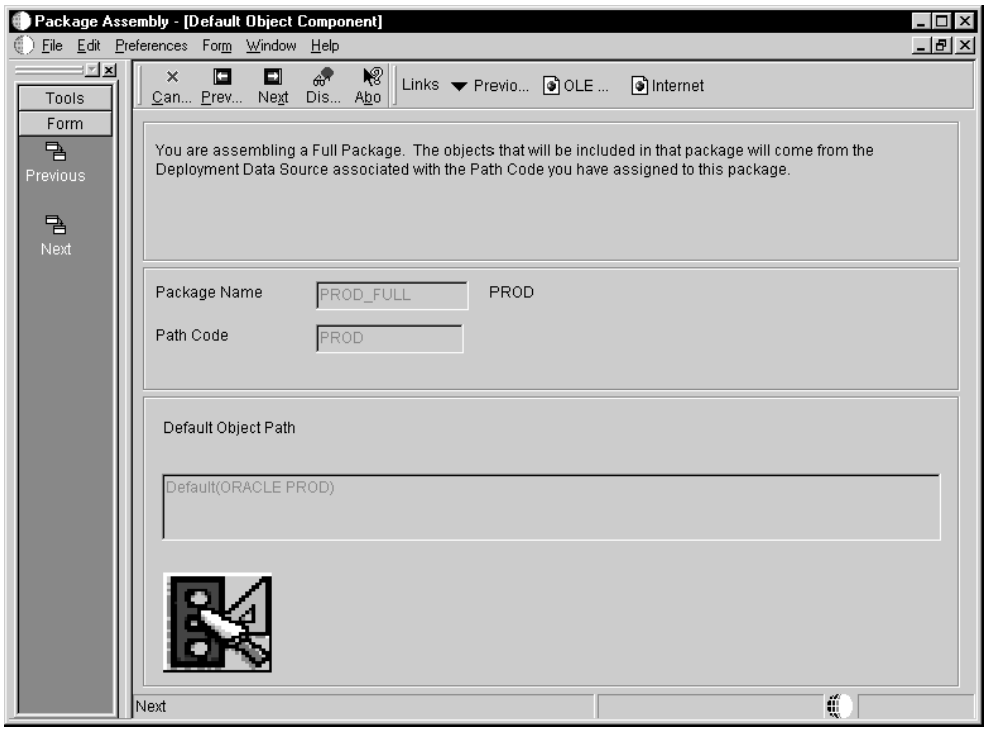
    For more information about entering a database location, see *Entering a Database Location*.

14. For an update, click Clear unless the package includes a Foundation.
15. Complete one of the following:

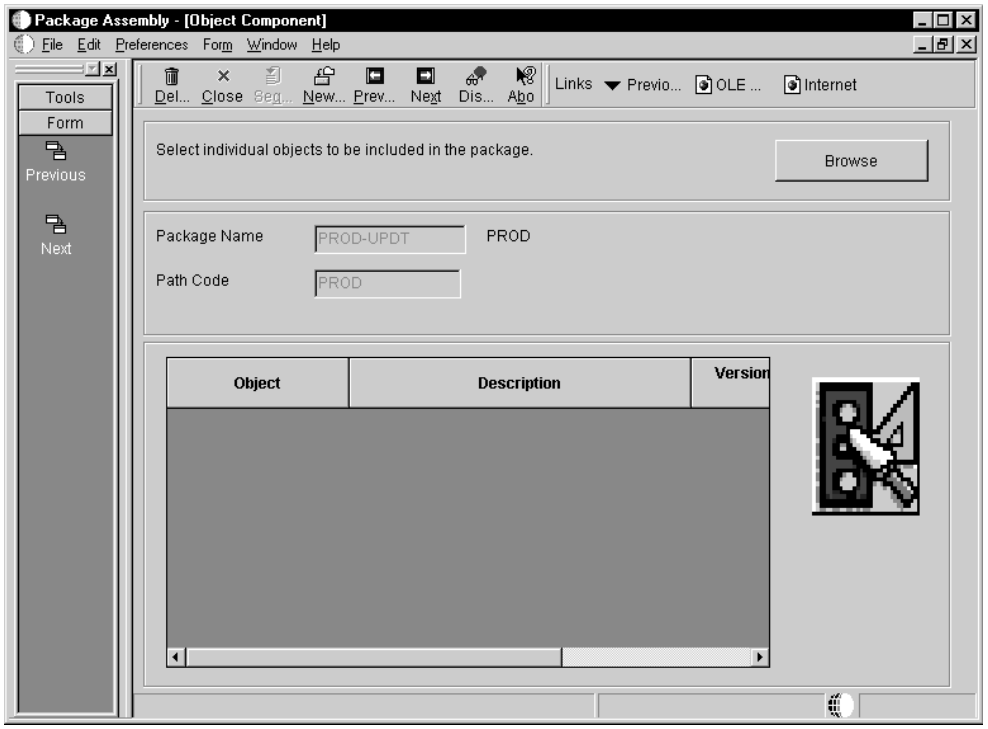
    - If you are assembling a full package, click Next.

      For a full package, ERP 9.0 builds your package from the deployment data source that is associated with the default object path. Verify that the correct location appears on the form and proceed to the next step.

    - If you are assembling a partial or update package, click Next on Database Component, and then proceed to the next step.



When you assemble a partial or update package, the Object Component form appears. This form allows you to specify the individual objects that you want to include in your package. When you revise a previously assembled package, objects that you added earlier also appear.



16. On Object Component, to add an object, click Browse.

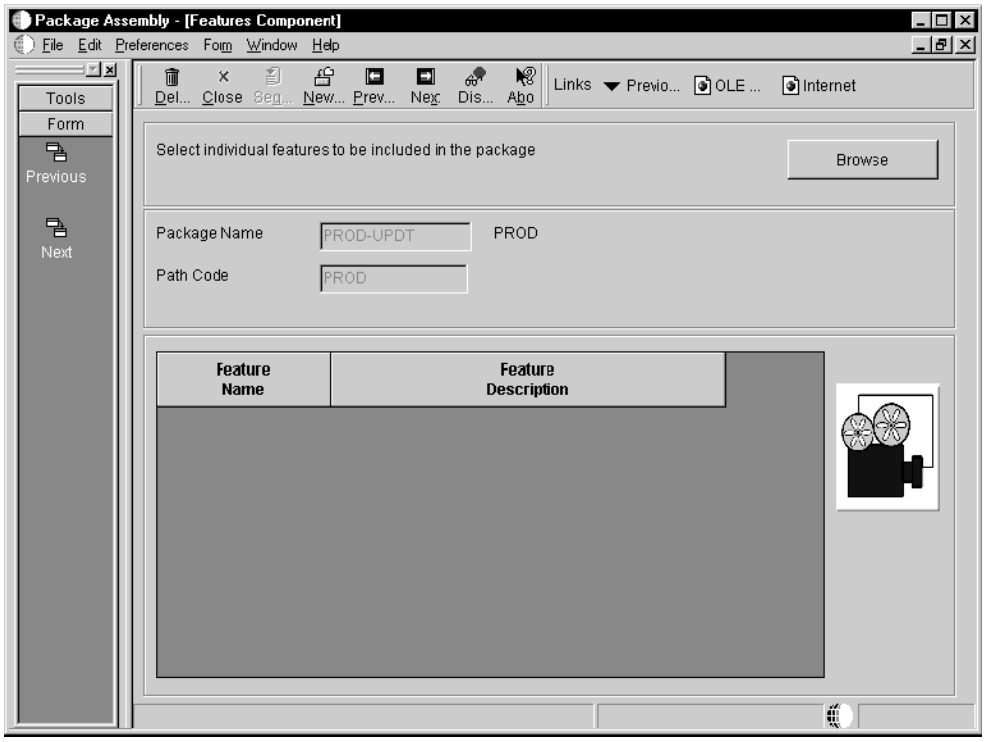
    The Object Component Selection form appears.

17. On Object Component Selection, locate and select the objects that you want to include in your package and then click Close to return to the Object Component form.

    For more information about adding objects, see *Adding Objects to a Package*.

18. On Object Component, click Next.

    The Features Component form appears, on which you can specify the features that you want to include in your package. When you revise a previously assembled package, the system displays the features that you added earlier.



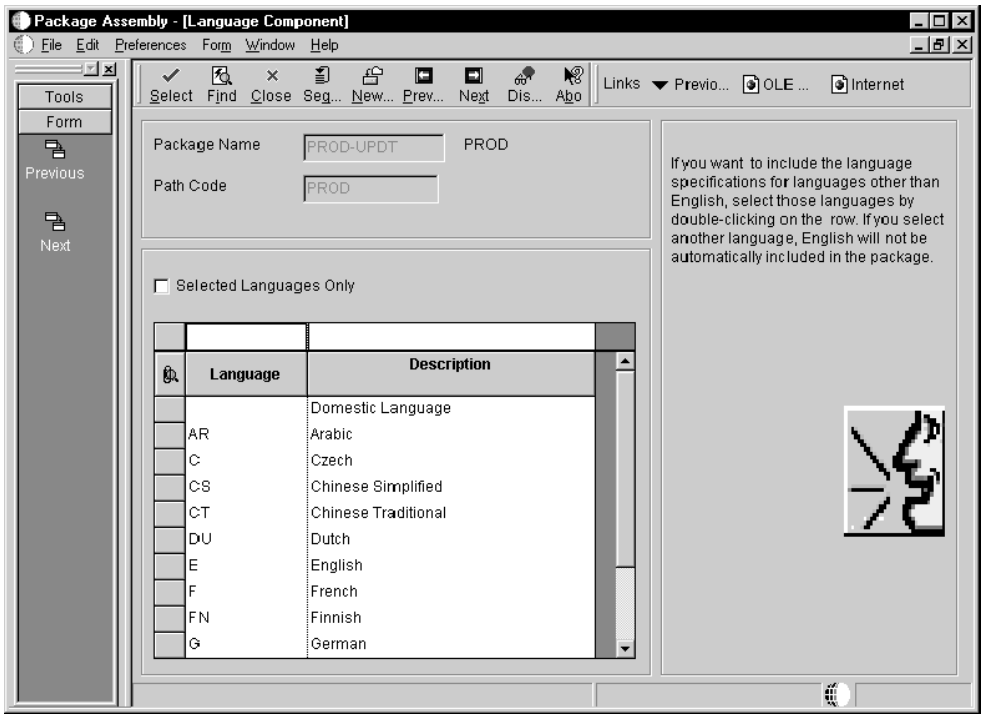
19. To add a feature, click Browse to display the Feature Component Selection form.
20. On the Feature Component Selection form, click find to display a list of features, choose one or more features, and then click the Select button to add the features that you want to include in your package.

    To choose multiple features, press the Ctrl or Shift key while clicking features, and then click Select.

21. Click Close to return to Features Component.

    For more information about adding features, see *Adding Features to a Package*.

22. On Feature Component, click Next.



23. Click Next if English is the only language that you want to configure.
24. To add a language to the language specifications for your package, double-click its row header in the detail area.

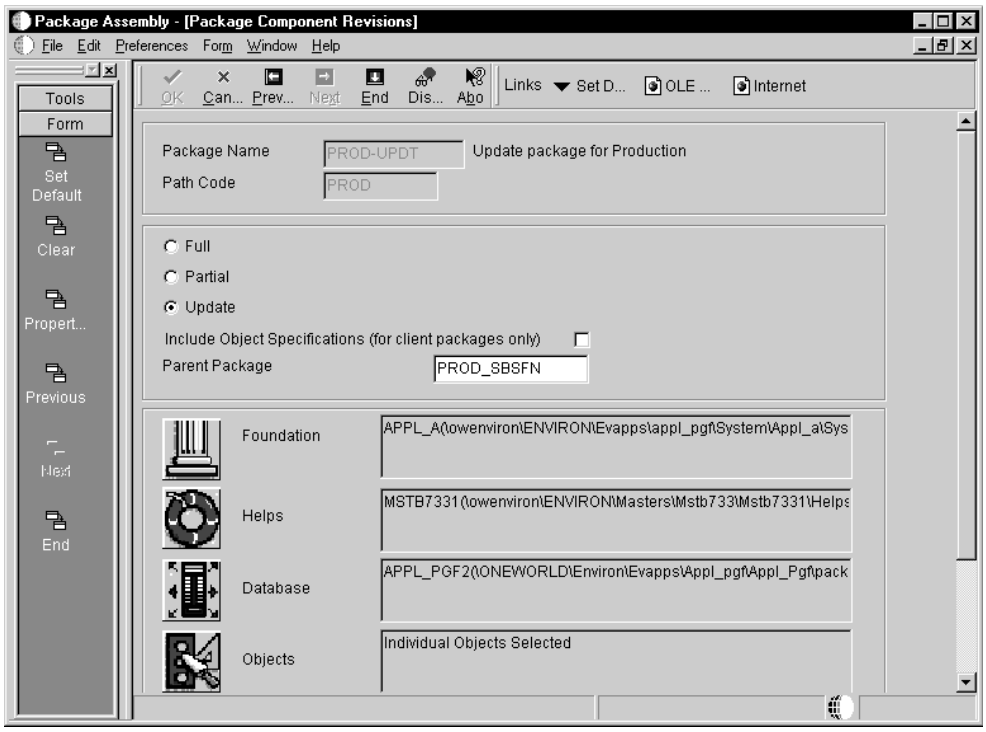
    If you add a language to your package, only that language will be included. For example, if you add French, English will not be included even though it is the default language. To include two languages (such as French and English), you must choose the detail records for both languages. For more information about installing ERP 9.0 in multiple languages, see the *ERP 9.0 Installation Guide*.

25. Click Next.

Reviewing the Package Assembly Selections

The Package Component Revisions form allows you to see, at a glance, the current foundation, help, and database locations, as well as the objects and features in the package and the language selection, if one exists.

► To review the package assembly selections



1. On Package Component Revisions, to change any of the package components, click the icon for the component that you want to change.

   The form for that package component appears.

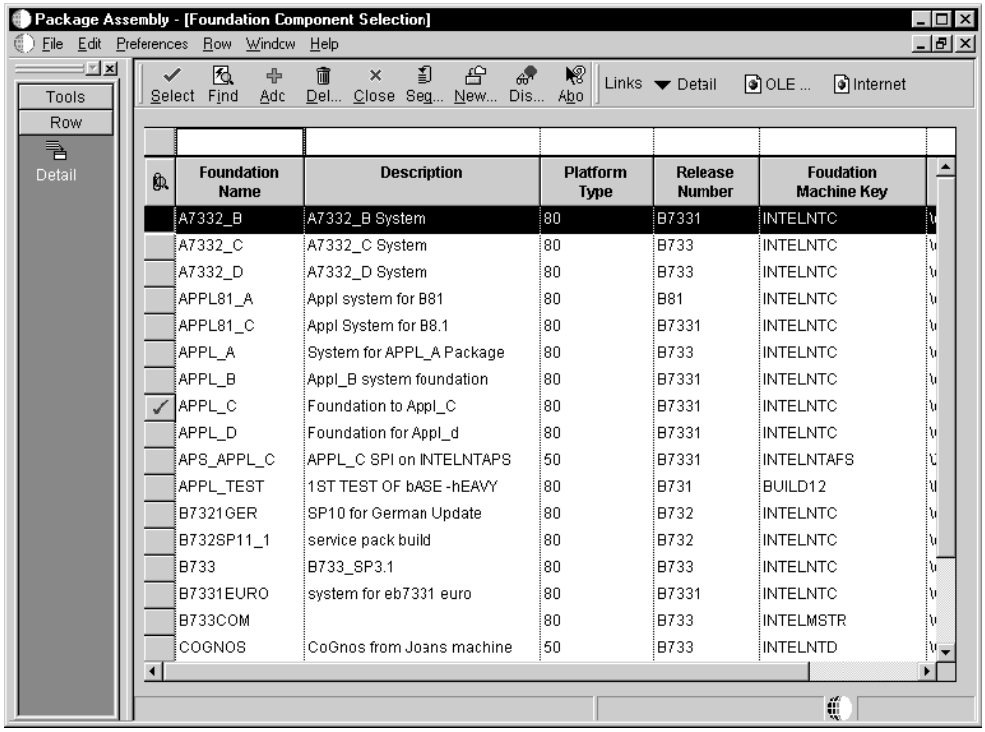
2. When you are finished assembling the package, click End to exit from the Package Assembly Director.
3. Continue with *Activating an Assembled Package*.

Entering a Foundation Location

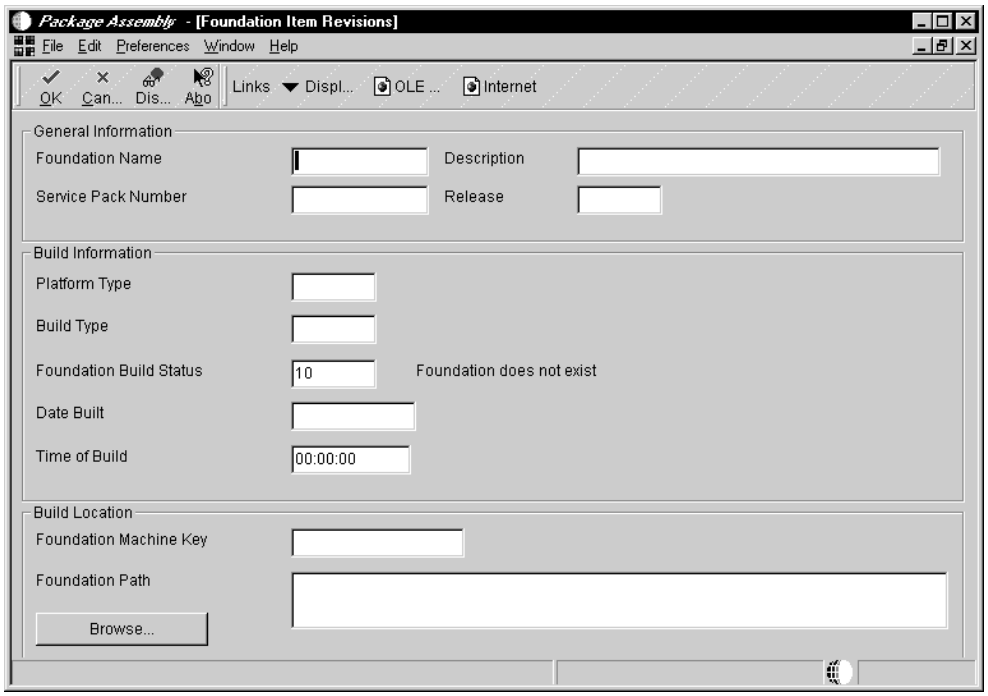
The default foundation location is determined by the release that is associated with the path code for the package. This location is normally the system directory that is at the same directory level as your path code. Enter a foundation location to change the foundation location from the default location that appears on the Foundation Component form.

► To enter a foundation location

1. Access the Foundation Component Selection form by doing one of the following
   - On Foundation Component, click Browse.
   - On Package Component Revision, click the Foundation icon.



2. Locate the existing foundation that you want to use in your package, or, to add a new foundation, Click Add.



3. On Foundation Item Revisions, complete the following fields and click OK:

- Foundation Name
- Description
- Service Pack Number
- Release
- Platform Type
- Build Type
- Foundation Build Status
- Date Built
- Time of Build
- Foundation Machine Key
- Foundation Path

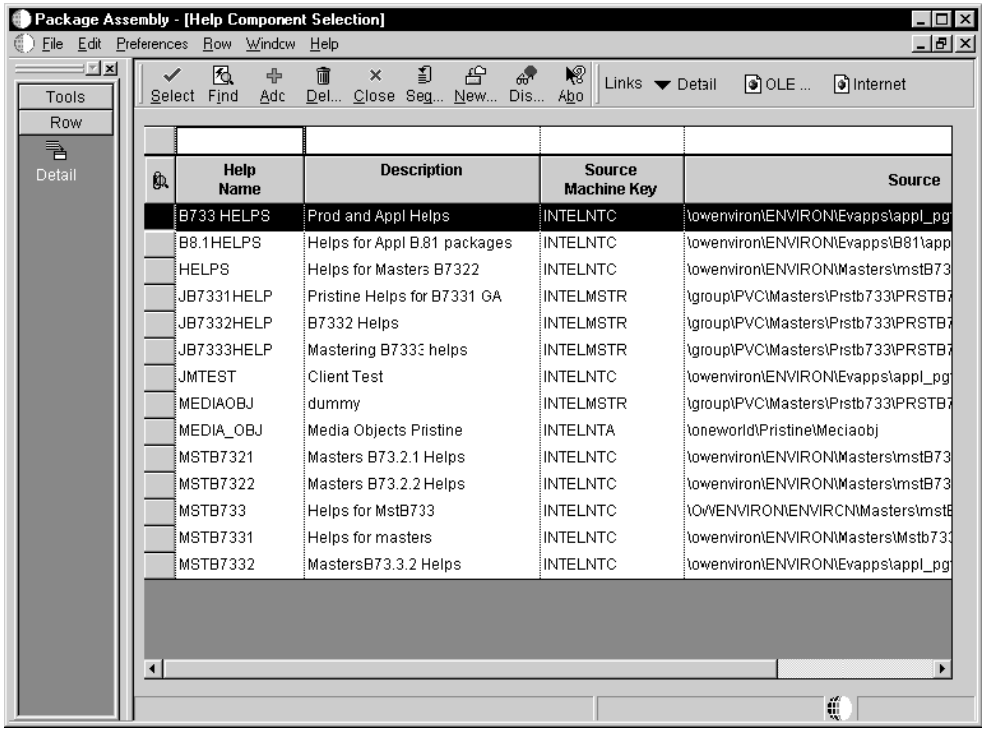
Entering a Help File Location

For all package types, if you do not enter a help location, the package build creates an INF that has help mapped to the default location that is specified in Release Master (one set of help per release). If a help file location is defined for a package, the package build copies the help file from the location that is specified in Help Item Revisions to the package directory.

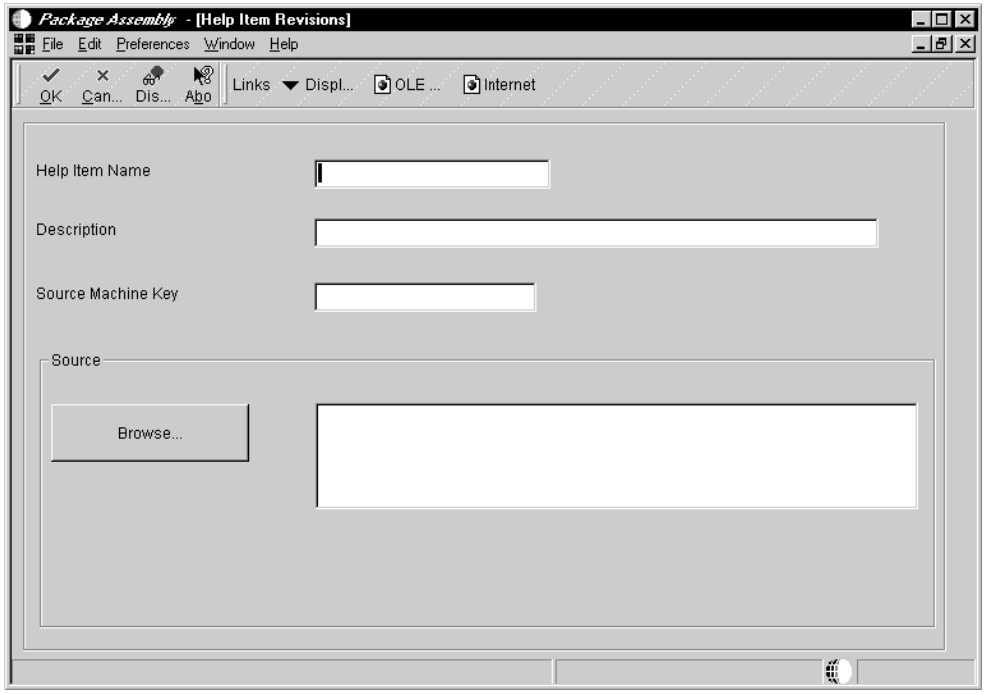
The package INF specifies that help files are located in the package itself, instead of the default location. However, the Workstation Installation program installs help to the workstation based on the user's deployment preference of client or server. The Workstation Installation program updates the JDE.INI file according to the package INF help location and the deployment preferences help location.

► To enter a help location

1. Access the Help Component Selection form by doing one of the following:
   - On Help Component, click Browse.
   - On Package Component Revision, click the Helps icon.



2. Locate the existing help location that you want to use in your package, or, to add a new help location, click Add.



3. On Help Item Revisions, complete the following fields and click OK.

- Help Item Name
- Description
- Source Machine Key
- Source

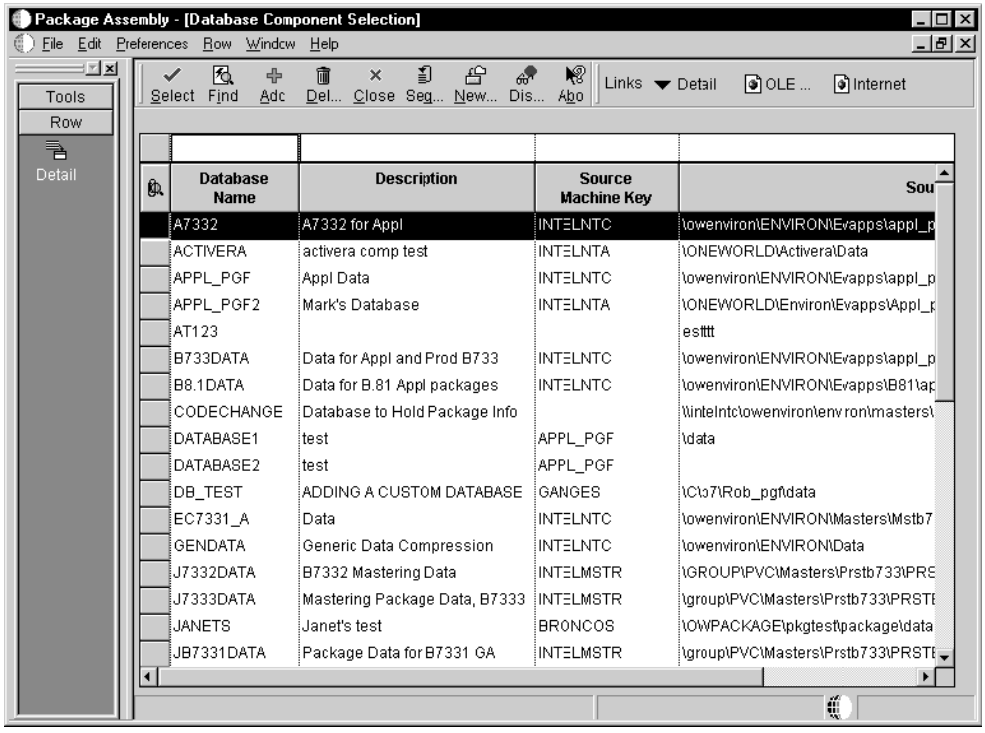
Entering a Database Location

For full and partial packages, if you do not specify a database location, the system uses the default database path (*pathcode*\Packages\Data). Update packages do not require a database.

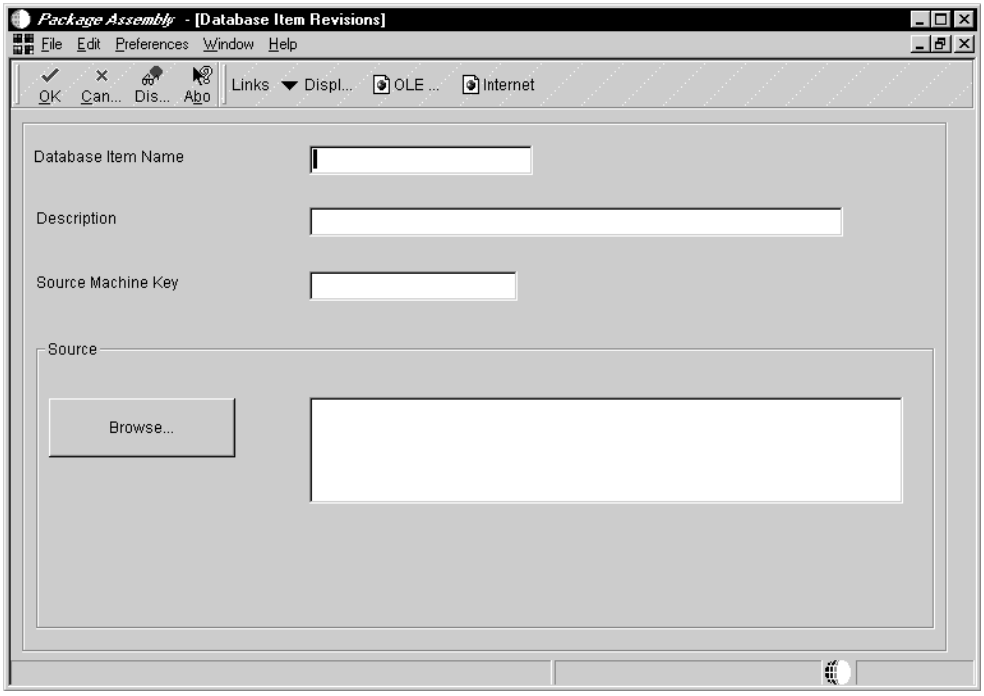
This task describes how to change the database location from the default location that appears on the Database Component form.

► To enter a database location

1. Access the Database Component Selection form by doing one of the following:
   - On Database Component, click Browse.
   - On Package Component Revision, click the Database icon.



2. Locate the existing database location that you want to use in your package, or, to add a new location, click Add.



3. On Database Item Revisions, complete the following fields and click OK.
   - Database Item Name
   - Description
   - Source Machine Key
   - Source

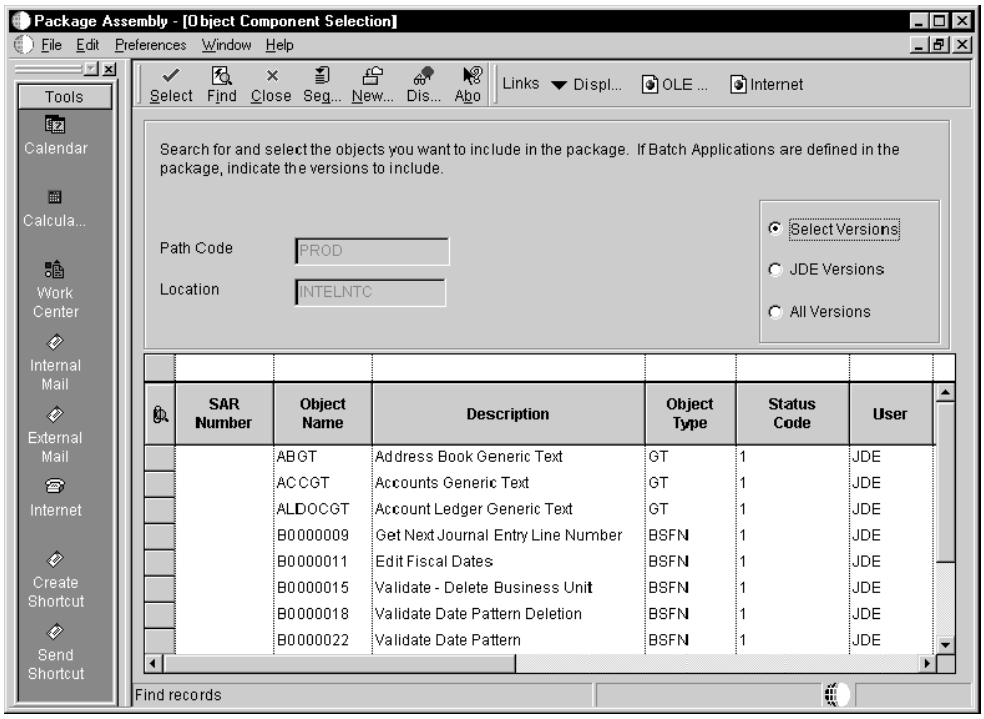
Adding Objects to a Package

If you are assembling an update or partial package, you can select individual objects to include in the package. When you finish adding objects, those objects appear on the Object Component form, the Package Component Revision form, and the Work with Packages form.

This task describes how to add objects to a partial or update package. The procedure is the same whether you are adding objects for the first time or revising a previously-assembled package.

► To add objects to a package

1. Access the Object Component Selection form by doing one of the following:
   - On the Object Component form, click Browse.
   - On the Package Component Revisions form, click the Objects icon to display the Object Component form, and then click Browse.



2. Locate the existing object that you want to include in the package.
3. If you are adding a batch application, click one of the following options and then click Select:

   - Select Versions

     Click this option to select only specified versions of the object.

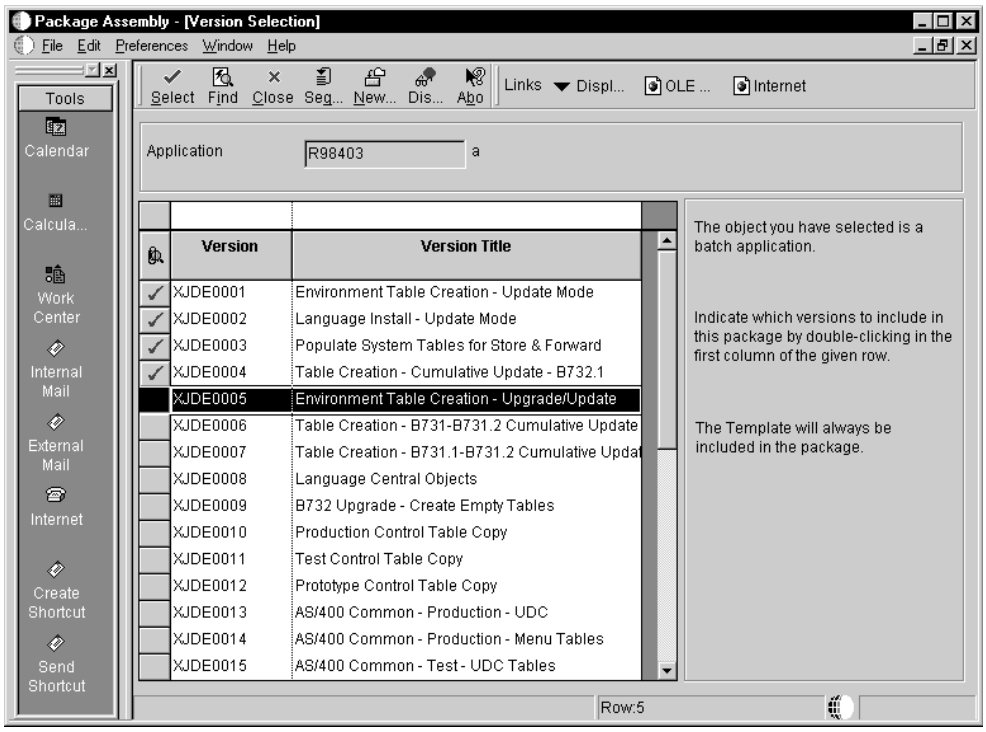
   - JDE Versions

     Click this option to select all J.D. Edwards versions of the object.

   - All Versions - selects all versions of the object

     Click this option to select all versions of the object.

   If you are adding a batch application and clicked Select Versions, the Version Selection form appears.



4. On Version Selection, choose from the list the specific versions that you want to include in the package.

   A check mark appears to the left of each version that you select.

5. When you finish selecting versions, click Close to return to the Object Component Selection form.
6. Repeat this process until you have finished adding objects to your package.
7. When you are finished, click Close to return to the form from which you accessed the Object Component Selection form.

Adding Features to a Package

A feature is a set of files or configuration options, such as registry settings, that is copied to a workstation or server to support a ERP 9.0 application or other ERP 9.0 function. Like ERP 9.0 objects, features are built into a package and deployed to the workstations and servers that require the feature components.

When you finish adding features to the package, those features appear on the Feature Component form, Package Component Revision form, and the Work with Packages form of the Package Assembly Director.

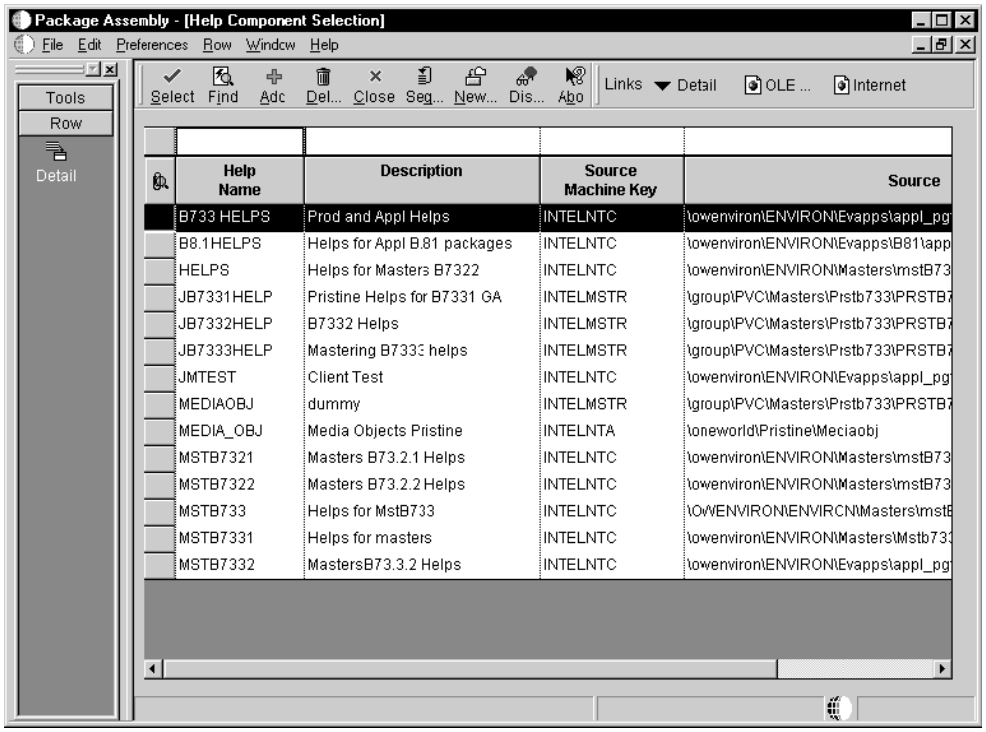
This task describes how to add features to a package. The steps are the same whether you are adding features for the first time or revising a previously-assembled package.

See Also

- *Incorporating Features Into Packages* in the *Package Management Guide* for information about adding features

► To add features to a package

1. To access the Feature Component Selection form, do one of the following:
   - On the Feature Component form, click Browse.
   - On the Package Component Revisions form, click the Features icon to display the Feature Component form, and then click Browse.



2. Find and choose the existing features that you want to include in the package, and then click Select.

   To choose multiple features, press the Control or Select key.

3. If the feature that you want to include has not been defined, you can create the feature definition by clicking Add.

   The Feature Based Deployment Director launches. You can use this feature to create the new feature.

4. Repeat steps 2 and 3 until you have finished adding features to your package.
5. When you are finished, click Close to return to the form from which you accessed the Feature Component Selection form.

Activating an Assembled Package

After you have assembled a package, the package status remains at Assembly. While you define the package, it is inactive. You must specifically change the package status to Assembly-Definition Complete to activate the package. An assembled package cannot be built until the status has been changed to Assembly-Definition Complete. The Assembly-

Definition Complete status indicates that you are finished assembling the package and are ready to begin the build definition process.

You can change the package status at any time until you start the build definition process. That is, even after you have changed a package status to Assembly-Definition Complete, you can change the status back to In Definition if you need to revise the assembled package.

▶ To activate a package

From the Package and Deployment Tools menu (GH9083), choose Package Assembly (P9601).

1. On Work with Packages, choose the package that you want to activate.

   Packages that are currently In Definition are identified by an open box icon, while packages that have an Assembly-Definition Complete status have a closed box icon.

2. Choose Active/Inactive from the Row menu.

You can use this same process to change the status of a complete package back to In Definition.

Note

When you are ready to define the build for the package, follow the steps described in *Defining Package Builds*.

Building a Package

After you activate a package, you can submit the build to start building the package. This process can take several hours, and the time depends on the number of objects in the package.

During the build of a package, the program counts the number of spec records copied from the Central Objects data source to the package. A list of the number of spec records for each object copied is displayed in a report along with the original number of spec records in the parent package. You can then compare these numbers to ensure that the package built successfully. For a client package, the number of records copied should be the same for all the spec files. For a server package, the number of records copied for all but GBRSPEC should be the same as the original number.

▶ To start a package build

1. On Work With Package Build Definition, choose Submit Build from the Row menu when you are ready to initiate the package build.
2. Click one of the following options and click OK:
   - On Screen
   - To Printer

   The form automatically closes and ERP 9.0 begins building the package. Build time varies, depending on the number and size of the items in your package. When the build finishes, the report either appears on the screen or prints, depending on the destination that you specified.

During this process, the program also counts the number of spec records copied from Central Objects to the package.

3. Review the report to make sure that all components in your package were built successfully. Any of the following indicates a problem:
   - Error messages
   - For client packages, check if the number of records copied is not the same for all the spec files
   - For server packages, check if the number of spec records copied is not the same as the original number

     (For GBRSPEC, the number of records copied will not be the same as the original)

4. If the report has any of the problems indicated above, review the error logs for more detail.

Note
If the package build finishes successfully, you can schedule the package for deployment as described in *Deployment*.

Verifying a Package

After you assemble a package, you can verify whether the package can be built successfully. You can use this verification to test the package before you submit the build, or troubleshoot problems with the build process if the package build fails.

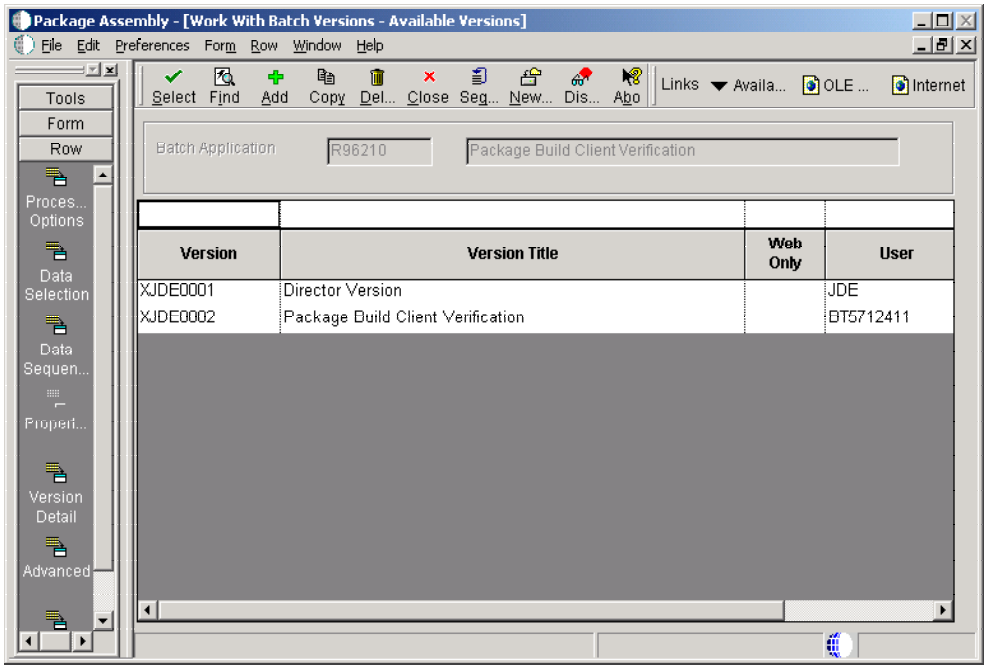
During the verification process, the program verifies the following:

- Disk space is adequate
- Central objects and package build tables are accessible
- User has permissions to create directories on the deployment server and enterprise server
- Required service pack is installed
- Required MDAC is installed
- Machine tables are set up
- Required compiler version is installed
- Enterprise Server port is accessible
- Debug levels of the jde.ini files are adequate for the client and enterprise server

► To verify a package

Choose Package Assembly (P9601) or Package Build (P9621) from the Package and Deployment Tools menu (GH9083).

1. From the Row Exit menu, choose Build Verification.



2. On Printer Selection, choose the desired printer, and then click OK.

   The program verifies the configuration of the environment and package.

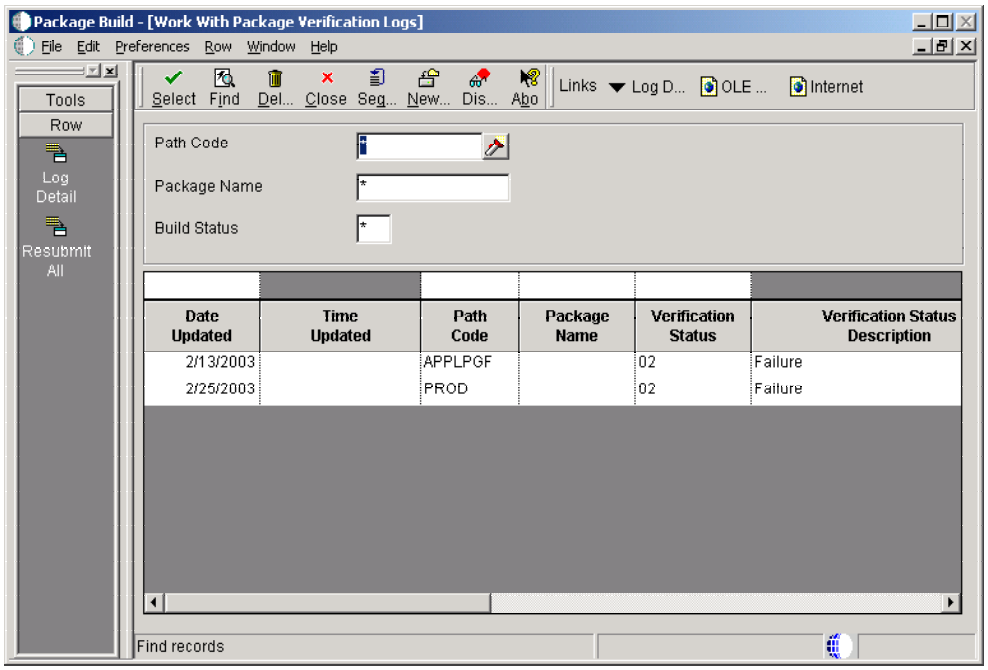
Viewing the Verification Log

When you run the Build Verification, the program creates a detailed report of the verification process. This report lists line-by-line the items that were checked, whether the test succeeded or failed and, if the test failed, a suggestion for how to correct the failure. All the verification logs are stored and can be accessed using the “View Verification” button on the Form Exit menu of the Package Assembly (P9601) or Package Build (P9621) program.

► To view the verification log

Choose Package Assembly (P9601) or Package Build (P9621) from the Package and Deployment Tools menu (GH9083).

1. From the Form Exit menu, choose View Verification.



2. Type the Path Code, Package Name, or Build Status, and click Find.

   The program lists the verification logs that match the criteria, and displays the verification status (Success or Failure).

3. Choose a verification log, and click Log Detail from the Row menu.

Revising an Existing Package

After you have assembled a package, you can easily use the Package Component Revision form to revise any of the components in the package by. You do not need to complete all of the forms in the Package Assembly Director to revise a package.

Before You Begin

- Verify that the status of the package is In Definition. If you try to revise a package that has a definition of Assembly-Definition Complete, the system displays an error message. To change the status of a package, choose Active/Inactive from the Row menu on the Work with Packages form.

► To revise an existing package

From the Package and Deployment Tools menu (GH9083), choose Package Assembly.

1. On Work with Packages, locate the package that you want to revise, either by clicking Find or by expanding the Packages tree.
2. Choose the package that you want to revise, and then choose Package Revisions from the Row menu.
3. On Package Component Revisions, make any necessary changes.

4. When you are finished revising the package definition, click OK to return to the Work with Packages form.

   If any build information exists for the package, ERP 9.0 warns you that your changes will delete the existing build information.

5. Click one of the following:

   - OK, to accept your revisions and delete the existing build information. If you accept the revisions, you should update the build information so that it reflects the changes that you made.
   - Cancel, to delete your revisions and save the existing build information.

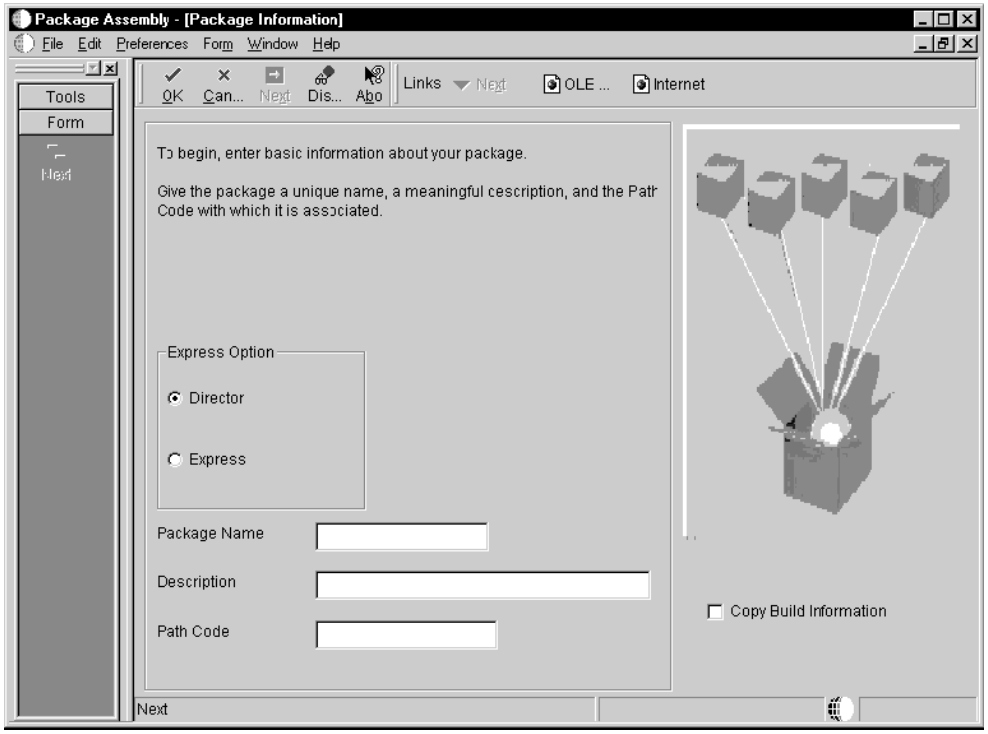
Copying a Package

In some cases, you might prefer to copy an assembled package rather than create a new one. Also, when you make a copy of a package that has been built (that is, it has a status of Build Complete), the original package definition records are copied and added to the new package. Directories (including server directories) from the old package are also copied to the new package.

► To copy a package

From the Package and Deployment Tools menu (GH9083), choose Package Assembly (P9601).

1. On Work with Packages, find and choose the name of the package that you want to copy, and then click Copy.



2. On Package Information, enter the new package name, description, and path code.

   If the package that you are copying has been built (that is, the package has a status of Build Complete), the Copy Build Information option will be available. When you turn on this option, all of the package definition records, directories, and server directories from the original package are copied into the new package.

3. Click OK to return to the Work with Packages form, where you can choose Package Revisions from the Row menu to review and change any of the package components.

Deleting a Package

The Work With Packages form contains an option that allows you to delete existing packages, provided that they are not currently in the process of building. You cannot delete packages that are being built.

When you delete a package, the system deletes all information for that package from the Software Package Header table (F9603) and the Software Package Detail table (F9631). In addition, if the package has been built, the system deletes all specifications for that package from all servers, and it deletes any information in the Software Package Build Header table (F96021) and the Software Package Build Detail table (F9622). The system also deletes any existing deployment records.

► To delete a package

From the Package and Deployment Tools menu (GH9083), choose Package Assembly. The Work with Packages form appears.

1. Find and choose the name of the package that you want to delete.
2. Click Delete.
3. Confirm the deletion by clicking OK when the warning message appears.

Defining Package Builds

After you assemble a package, you must define the package build before you can deploy it to your workstations. The build process reads the central objects data source for the path code that you defined in the package. This information is then converted from a relational format to replicated objects, which are put in the package itself.

The B9 (release name) directory structure looks similar to the following. Directories with an asterisk represent locations to which developers check in development objects.

PD9 (path code name)

    PACKAGE

        PackageA (package name)

            bin32

            include

lib32

make

obj

res

source

spec

work

*bin32

*include

*lib32

make

*obj

*res

*source

work

When you build a package, the directories under the package name are populated. Information for the source and include directories is copied from the location under the path code on the deployment server from which developers check in development objects. Information for all other directories comes from the central objects data source. The bin32, lib32, and obj directories are populated with the output of the business function build process.

Understanding the Build Process

The process that you perform to build a package might take several hours. For this reason, J.D. Edwards recommends that you initiate the actual package build at the end of the working day, if possible. The following is a checklist of the tasks that you need to complete when you build a package.

- Transfer objects.

  Ensure that all of the objects that you want to include in the build have been transferred to the appropriate path code.

- Ensure that the database for the package has the most current replicated data.
- Build a package.

  Build a package using the path code to which objects were transferred.

- Perform a cross-reference build (optional).

Perform a cross-reference build to verify that the cross-reference information reflects the changed objects. This process takes up to 15 hours to complete. You can deploy your package before the cross-reference build has finished.

- Deploy the software to the following:
  - Workstations and servers
  - Tiered deployment locations

See Also

- *Understanding Object Movement* in the *Package Management Guide* for information about how to transfer objects
- *Data Replication* in the *System Administration Guide* for information about verifying replicated data
- *Cross Reference Facility* in the *ERP 9.0 Development Tools Guide* to perform a cross-reference build
- *Deployment* in the *Package Management Guide* to deploy packages

Understanding the Package Build Definition Director

Like the Package Assembly Director, the Package Build Definition Director simplifies and expedites the build definition process by displaying a series of forms that guide you through the process. As with the Package Assembly Director, you can always either click Next to continue to the next form or click Previous to go back to the previous form. Also, you can always cancel the build definition process by clicking Cancel.

The following table summarizes the function of each form in the Package Build Definition Director:

Package Build Definition Director form	Use this form to review introductory information about the Package Build Definition Director.
Package Selection form	Use this form to choose the defined package that you want to build. The status of the package must be Assembly Definition Complete.
Package Build Location form	Use this form to specify whether you want to build the package for the client workstation, one or more servers, or both clients and servers.
Server Selection form	Use this form to specify the server location. (The server location is required when you build a package for a server.)
Build Specification Options form	Use this form to specify whether you want to include all specification tables in the package or only selected tables. The option to build individual specifications is useful if a build fails and your package error log indicates that an individual specification file needs to be rebuilt.
Individual Specification Selection form	Use this form to include only selected tables in the package.
Business Function Options form	Use this form to build business functions. You can also specify the build mode, the severity level at which to interrupt the build process, whether to build business

	function documentation, and whether to clear the output destination before building.
Compression Options form	Use this form to specify package compression and to specify whether to compress directories (all or individual), data, helps, and foundation.
Individual Directory Selection form	Use this form to select the individual directories that you want to compress.
Build Features form	Use this form to enter file set and compression information for features that you added to the package. A feature is a set of files or configuration options, such as registry settings, that must be copied to a workstation or server to support an ERP 9.0 application or other ERP 9.0 function.
Package Build Revisions form	Use this form to review or change any of the options that you have specified for your package.

Building Business Functions During Package Build

As part of the build definition process, you can specify whether you want to build business functions. If so, the system globally builds business functions during the package build process. When you build business functions as part of the package build, the system performs the same process as if you had manually run the BusBuild program (selected Build from the Global Build menu) after you built the package.

The system retrieves source and header information from the package (from the source and include directories), compiles it, and stores it in the bin32, obj, and lib32 directories. The system builds business functions in the package, not on the workstation. If you turn on the Compress Package option, the system compresses the business functions after it builds them. .

The following guidelines apply to the path code, foundation, and destination for the business function build:

- When building business functions, use the path code that you defined in the package.
- The foundation is either the same as the foundation that is included in the package or, for an update package, it is the foundation for the parent package.
- Build output is directed to the bin32, obj, and lib32 directories of the package itself.
- When building a full or partial package, or when building an update package that includes a business function, always build business functions. Otherwise, the consolidated DLLs included in the package will not be current.

  For update packages, the system builds each business function individually. After it builds an individual business function, the system performs a global link for that object and all other objects that are in the same consolidated DLL. The global link affects all objects in the check-in location for the path code of that package.

  Similarly, the system individually builds any business functions that you defined in a partial package and then links these business functions with all objects in the check-in location. For a partial package, the system delivers all consolidated DLLs, not just those that are defined in the package itself.

Note

The build process does not update check-in location directories with the output from the build.

See Also

- *Business Functions* in the *ERP 9.0 Development Tools Guide* for more information about the BusBuild program

Defining a Package Build

This task describes how to define a package build using the Package Build Definition Director.

You can access the Package Build Definition Director either from the Package Assembly menu selection, or from the Package Build menu selection.

The advantage of accessing the director from the Package Assembly menu selection is that the system automatically enters the package name and other information. If you access the director from the Work With Package Build Definition form, you must manually specify the name of the package that you want to build.

Before you launch the Pack Build Definition Director, you can use the Work with Package Build Definition form to review information about any previously-built packages. For example, you can review the properties, build options, business function options, and compression options for the package. As on any other parent/child form, you can click the plus (+) symbol to view more information about the package or click the minus (-) symbol to view less information about the package.

Before You Begin

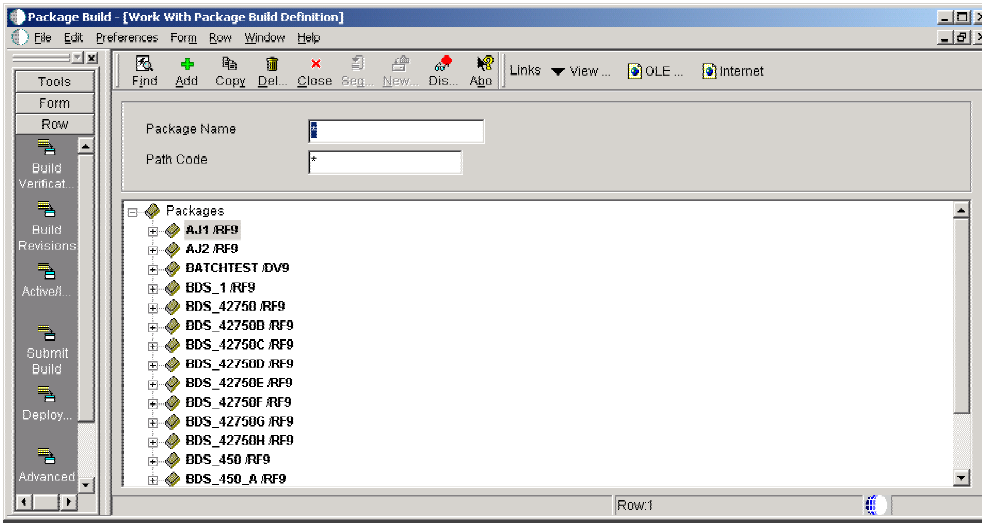
- Assemble your package and verify that the status of the assembled package is Assembly-Definition Complete. For more information, see *Assembling Packages*.
- Verify that Object Configuration Manager (OCM) mappings are correctly set for the Package Build report (R9621) and Server Package Build report (R9622), which the system generates as part of the package build process. For example, if you want the reports to run locally, ensure that the OCM mappings point to Local for the environment in which the package build is running. For more information about setting OCM mappings, see *Object Configuration Manager* in the *Configurable Network Computing Implementation Guide*.

▶ **To define a package build**

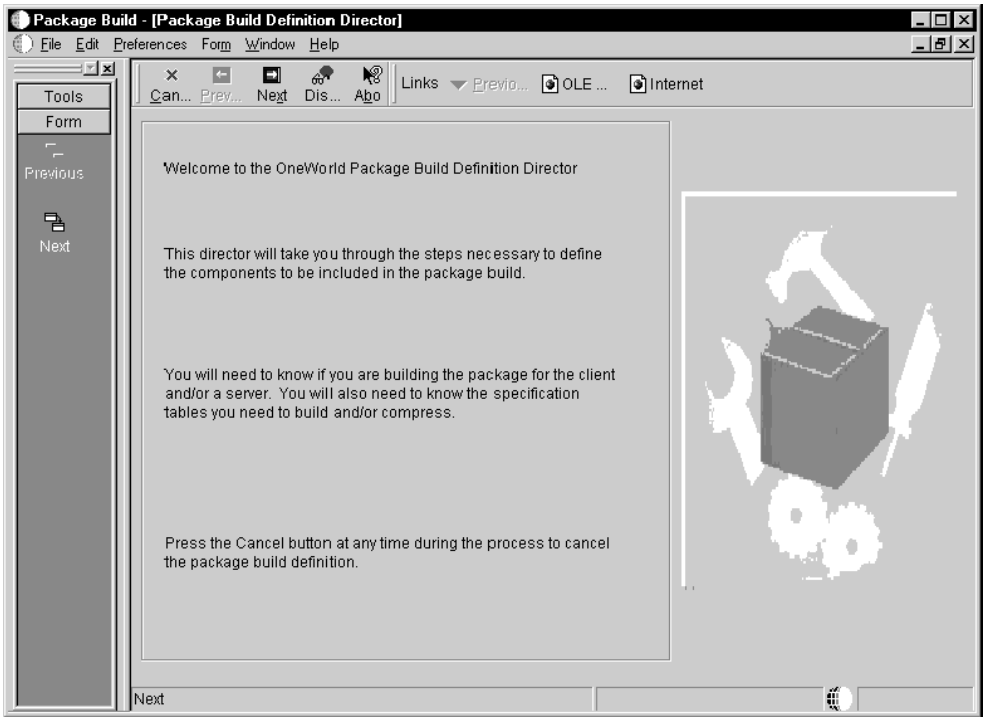
Use one of the following navigations:

From the Package and Deployment Tools menu (GH9083), choose Package Assembly. On Work With Packages, choose a defined package that has a Definition Complete status, and then choose Build Director from the Row menu.

From the Package and Deployment Tools menu (GH9083), choose Package Build.

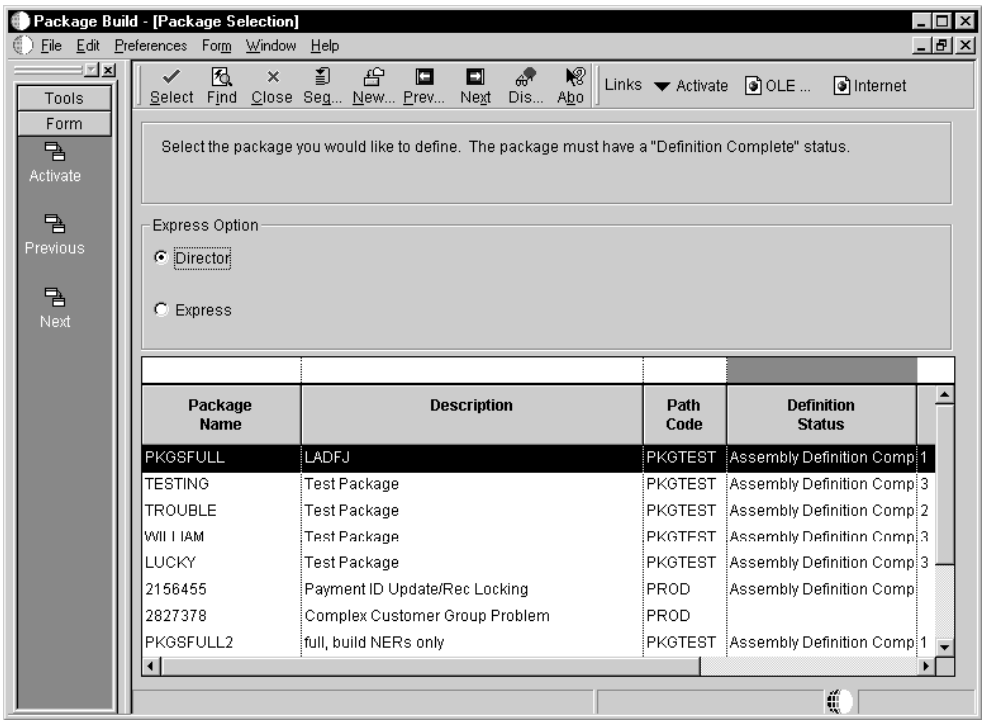


1. On Work With Package Build Definition, click Add to launch the Package Build Definition Director.



If you accessed the director from Work With Packages, skip step 2.

2. On Package Build Definition Director, click Next.



3. On Package Selection, find and choose the defined package that you want to build.

   If the package definition has a status of In Definition, you must change the status to Definition Complete before you build the package. To change the status, choose the package and choose Activate from the Row menu.

4. On the Express Option pane, turn on one of the following options:

   - Director

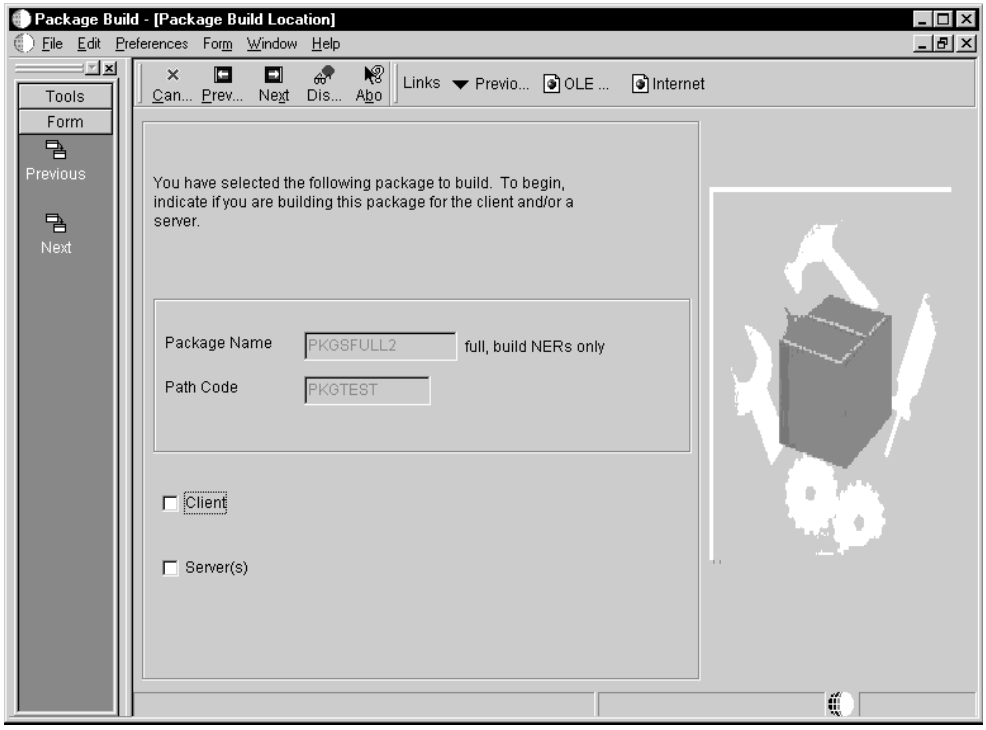
     Turn on this option if you want to customize your package build. Director allows you to navigate the Package Build Definition forms.

   - Express

     Turn on this option if you want to accept the default build parameters. Express allows you to accept the default options for the package build and skip the package build definition forms.

5. Do one of the following:

   - If you chose Express, click Next and continue with the task *To review the package build selections*.
   - If you chose Director, click Next and complete the remaining steps in this task.



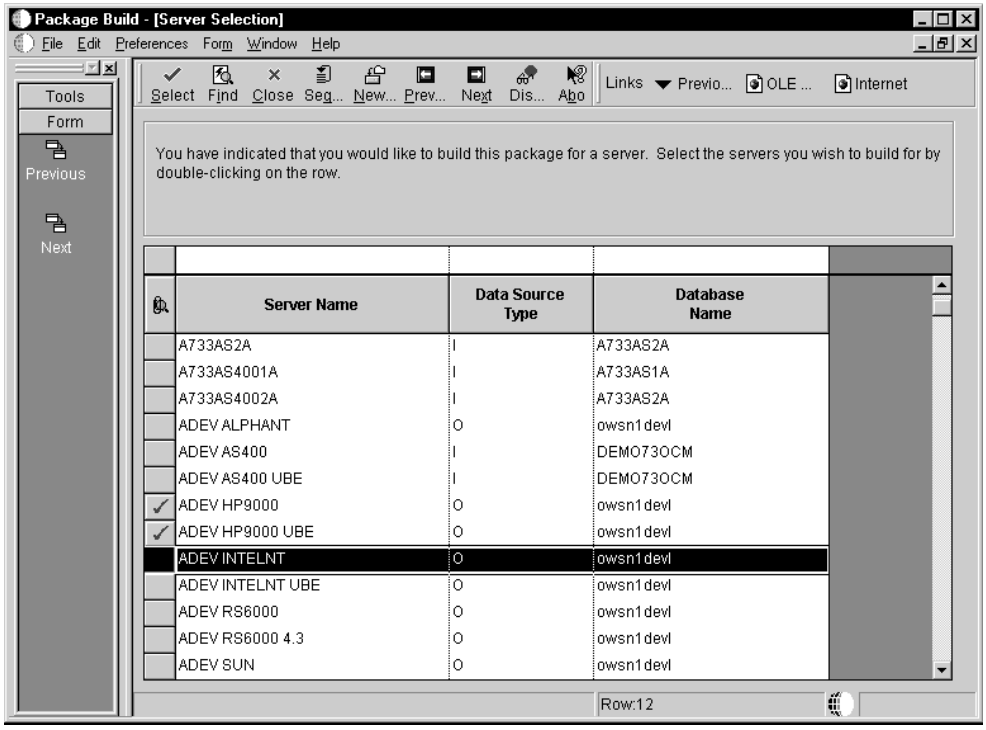
6. On Package Build Location, click one or both of the following options to turn them on, and then click Next:

   - Client
   - Server(s)

   If you are building a partial package, you cannot build it for a server.

   If you are building a package for workstations only, proceed to step 10.

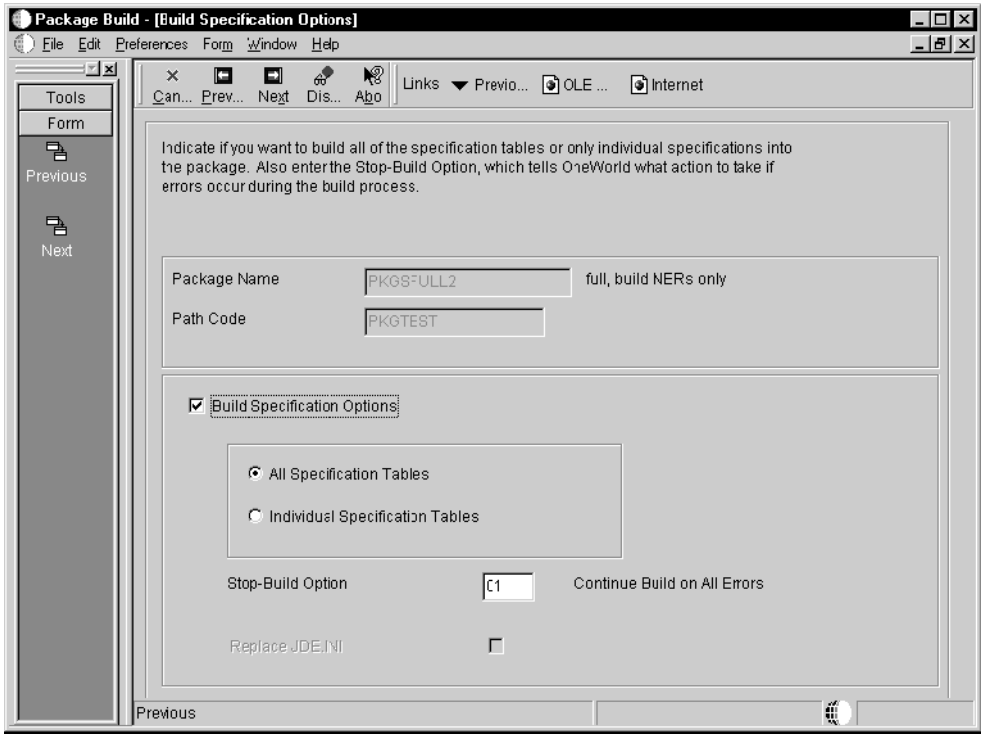
   If you clicked the Server(s) option on the Package Build Location form, the Server Selection form appears. This form enables you to choose the servers on which you want to build the package.



7. To choose a server, click Find, and then double-click the row header of the detail line for the server.

   A check mark indicates your selection. You can choose multiple servers.

8. Click Next.



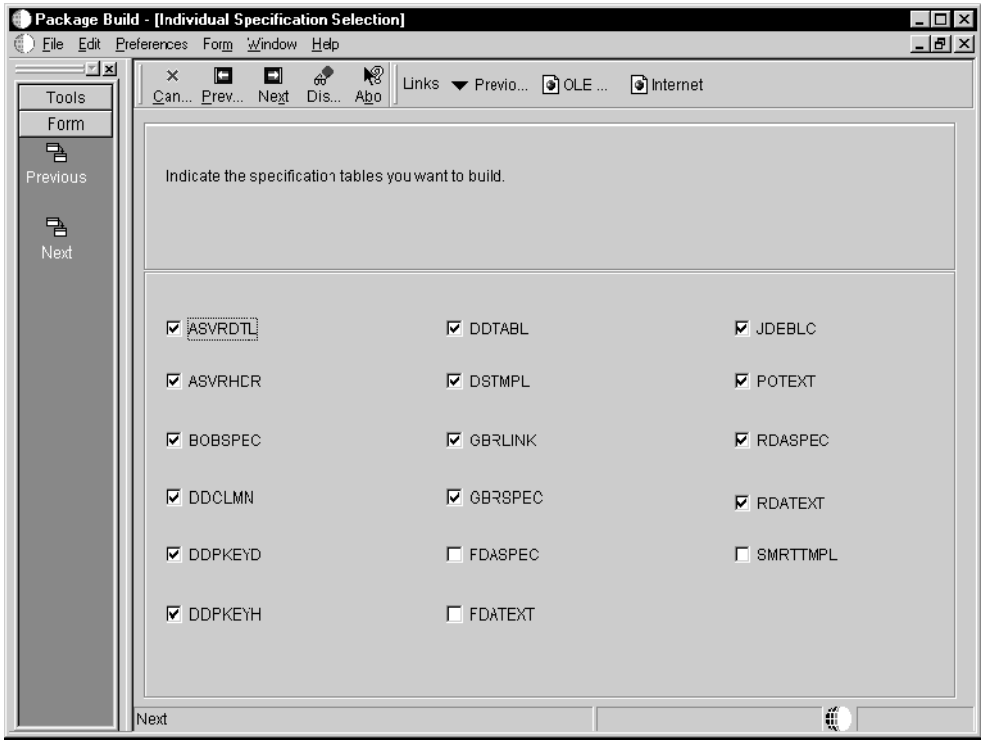
9. On Build Specification Options, click the following option to turn the option on:
   - Build Specification Options
10. Turn on one of the following options:
    - All Specification Tables
    - Individual Specification Tables

    If you choose All Specification Tables, all of the tables listed on the Individual Specification Selection form will be included in the package. If you are building all specification tables, click Next and complete the Business Function Options form.
11. Complete the following fields:
    - Stop-Build Option

      The Stop-Build Option field allows you to indicate the point at which B9 should stop the build. You can continue building on all errors, stop building on specification errors, stop building on business function errors, or avoid compressing when errors exist.
    - Replace JDE.INI

      This field applies to update packages only.
12. Click Next.

    If you chose to build individual specification tables, the Individual Specification Selection form appears.

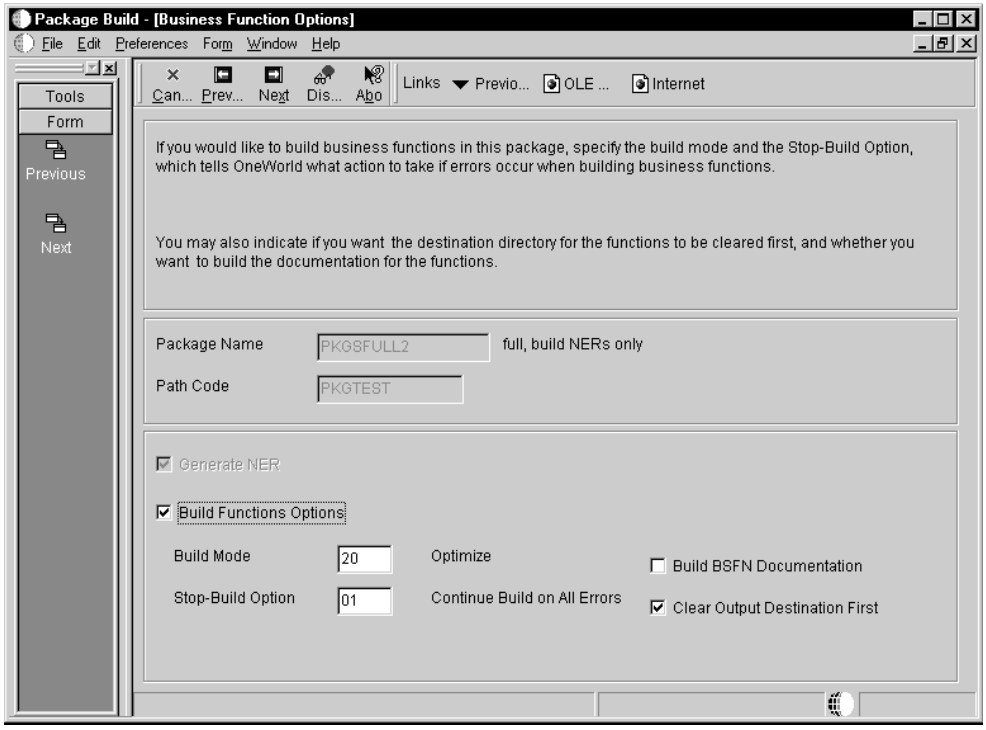


13. To indicate that you do not want to build a specification table, click its option to turn it off.

    You can turn off multiple options.

14. Click Next.

    For a full package or for an update package that includes business functions or tables with table event rules, the Business Function Options form appears.



15. On Business Function Options, complete the following fields:

    - Generate NER

      Turn off this option if you do not want to generate named event rules.

    - Build Functions Options
    - Build Mode

      You can choose from three build modes: debug, optimize, and performance. Debug and performance are for J.D. Edwards developers only. Users should select the optimize mode.

    - Stop-Build Option
    - Build BSFN Documentation

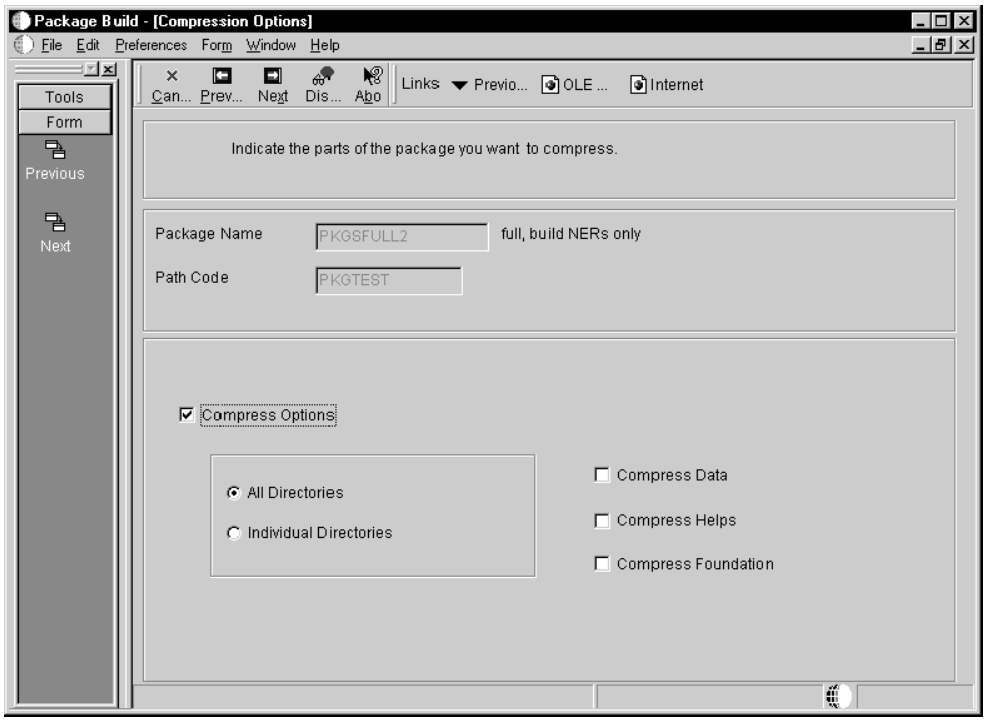
      Verify that this option is turned off.

    - Clear Output Destination

      This option applies to a full package.

16. Click Next.

    If you are building a full or partial package, the Compression Options form appears. If you are building an update package, the Compression Options form does not appear.



17. On Compression Options, click the following option to turn it on:
    - Compress Options
18. Choose one of the following options:
    - All Directories
    - For Server Package

      If the package that you are building will be deployed to a server, you should choose Compress Options only under the following circumstances:

      - You are building the same package for both the workstation and server, and you want to create compressed files for the workstation package
      - You plan to build the package on one enterprise server and deploy it to another enterprise server.

    If you need to compress a server package, click Compress Options and select All Directories. This selection compresses the directories on the enterprise server and copies them to your package on the deployment server in the following directory (the directory on which your server.ini file is located):

    <pathcode>/<package>/<package name>/<operating system type>

    The directories are compressed into file types that are compatible with the type of enterprise server that you are using. For NT servers, the file type is .cab, for UNIX the file type is .z, and for AS400, the file has no extension.

- For Client Package

  When you compress directories, the application objects that are included in the package are automatically compressed. B9 also creates an entry in the package INF

file that indicates whether the foundation, helps, data, and application objects are compressed.

- Compress Data

  Compress Data compresses the J.D. Edwards Supported Local Database that is associated with this package.

- Compress Helps

  Compress Helps compresses the help files that are associated with your package.

- Compress Foundation

  Compress Foundation compresses the foundation that is associated with your package.

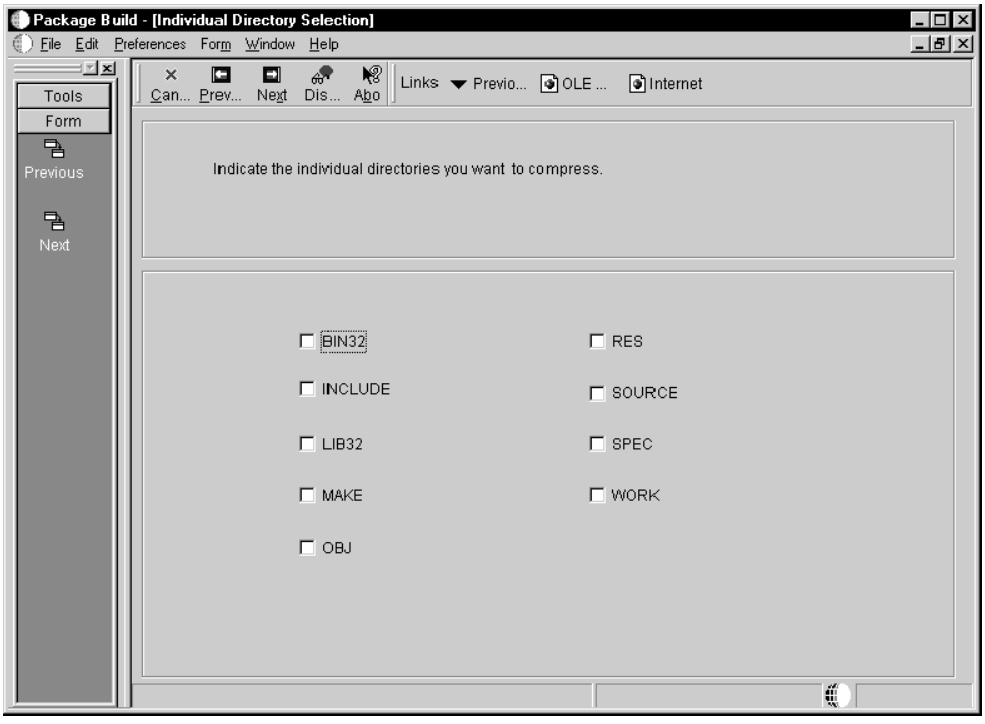
If you turn on the All Directories options, all of the directories listed on the Individual Directory Selection form are compressed. If you are compressing all directories, skip the following step.

Important
Verify that the DoCompression setting is set to 1 to enable compression. If this setting is not set to 1, the system does not create compressed files on the server. For more information, see *JDE.INI Settings for Server Package Builds*.

---

19. Click Next.

    If you chose to compress individual directories, the Individual Directory Selection form appears.

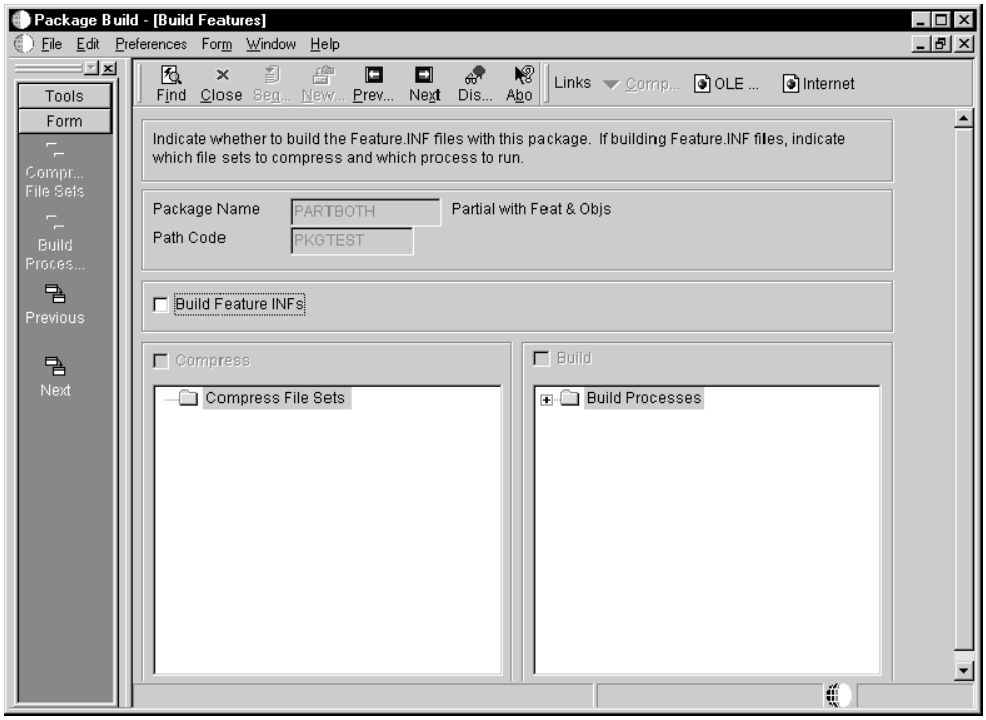


20. On Individual Directory Selection, to indicate that you want to compress a directory, click its option to turn it on.

    You can turn on multiple options.

21. Click Next.

    If the package does not include features, skip to the next task.



22. On Build Features, if you want to build a feature.inf file with the package, click the following option to turn it on:

    - Build Feature INFs

    When you turn on this option, the Compress and Build options become available. For more information about the Compress and Build options, see *Configuring Features During Package Build Definition*.

23. Click Next.

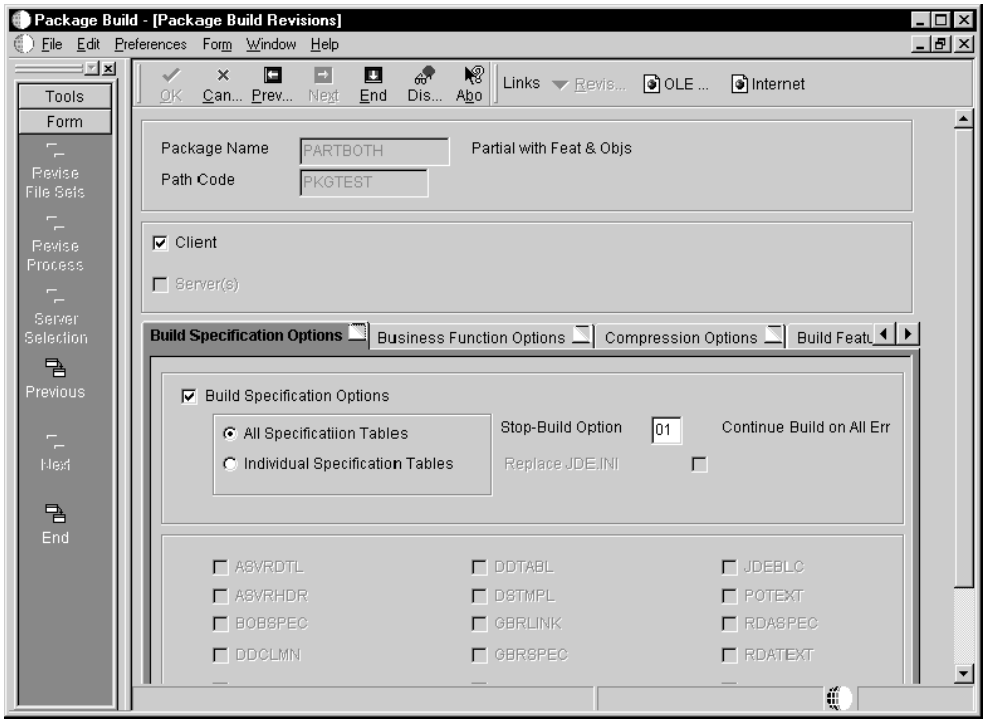
► To review the package build selections

After you complete the compression and build features options, the Package Build Revisions form appears.

This form allows you to see at a glance the current build options, business function options, compression options, and feature options that you specified for the package.

1. To change any of these options, click the tab for the type of option that you want to change.

   Only tabs for options that you selected appear on this form.



2. On Package Build Revisions, when you are finished reviewing or changing your build options, click End to exit the Package Build Definition Director.
3. On Work With Package Build Definition, choose Active/Inactive from the Row menu to activate the package.
4. Choose Submit Build from the Row menu when you are ready to initiate the package build.
5. Click one of the following options and click OK:
   - On Screen
   - To Printer

   The form automatically closes and ERP 9.0 begins building the package. Build time varies, depending on the number and size of the items in your package. A build could take five minutes for a small package, or several hours for a full package that contains all applications. When the build finishes, the report either appears on the screen or prints, depending on the destination that you specified.
6. Review the report to make sure that all components in your package were built successfully. If the report indicates any errors, review the error logs for more detail.

   If the package build finishes successfully, you can schedule the package for deployment as described in *Deployment*.

Processing Options for the Package Build Definition Director (P9621)

The Package Build Definition Director has the following processing options:

- 1. Enter a value to determine how changes will occur.

  Use this processing option to specify whether to allow changes to the build definitions on individual servers.

  When you enter 1 for this processing option, ERP 9.0 allows you to change the build definition for individual servers. If you leave this processing option blank, any revisions that you make to the build options and any information that you enter will be applied to all of the servers for which you are building the package.

  For example, if you enter 1 in this processing option, on the Work With Package Build Definition form, you can change the properties, the build options, the business function options, and the compression options for an individual server. If you leave this processing option blank, you cannot change an individual server, but you can change the package. The changes apply to all of the servers for that build.

- 2. Mark this processing option with a 1 if this process is for Mastering purposes.

  Leave this field blank if the process is for all users, or type 1 if the process is for mastering.

- 3. Mark this processing option with a 1 if the Build Verification UBE is to be run prior to building all packages.

  Type 1 in this field if you want the Build Verification program to run automatically before you build a package. Selecting this option forces the system to verify the Package Build configuration every time before a package is built. If the build verification fails, the package UBE will not run. For more information about the Build Verification program, see *Verifying a Package* in the *Package Management Guide*.

Verifying a Package

After you assemble a package, you can verify whether the package can be built successfully. You can use this verification to test the package before you submit the build, or troubleshoot problems with the build process if the package build fails.

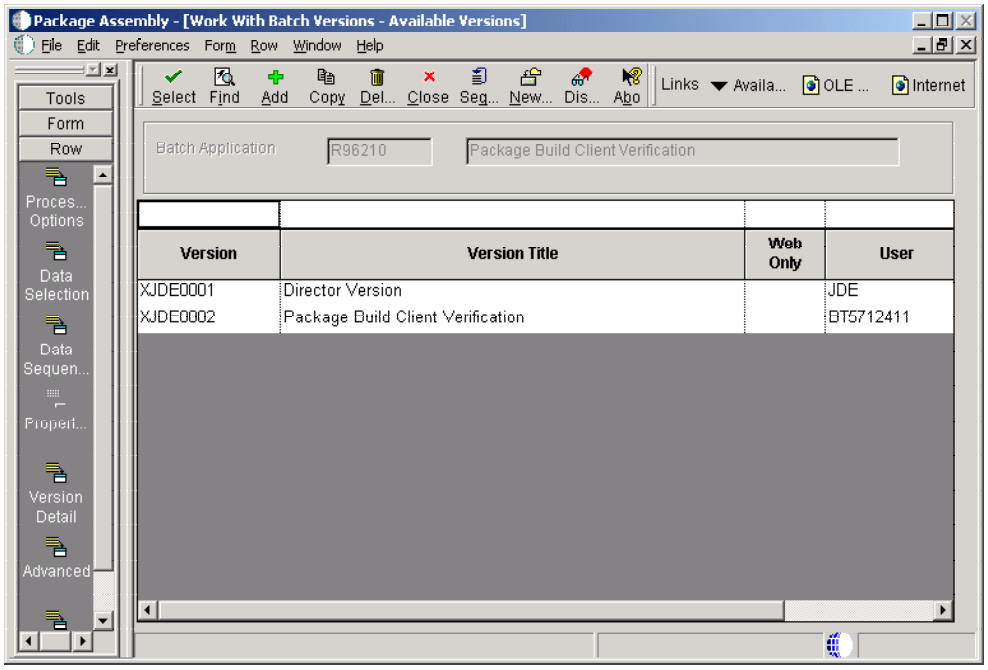
During the verification process, the program verifies the following:

- Disk space is adequate
- Central objects and package build tables are accessible
- User has permissions to create directories on the deployment server and enterprise server
- Required service pack is installed
- Required MDAC is installed
- Machine tables are set up
- Required compiler version is installed
- Enterprise Server port is accessible
- Debug levels of the jde.ini files are adequate for the client and enterprise server

► To verify a package

Choose Package Assembly (P9601) or Package Build (P9621) from the Package and Deployment Tools menu (GH9083).

1. From the Row Exit menu, choose Build Verification.



2. On Printer Selection, choose the desired printer, and then click OK.

   The program verifies the configuration of the environment and package.

Revising Build Options for a Package

After you enter the build options for a package, you can easily revise any of those options using the Package Build Revision form. You do not need to go through all of the forms in the Package Build Definition Director to revise build options.

► To revise a package's build options

From the Package and Deployment Tools menu (GH9083), choose Package Build.

1. On Work With Package Build Definition, find the package you want to revise, either by using the Find function or by clicking on the plus sign (+) to expand the tree structure.
2. If the package is active (its package icon is closed), choose the package and then choose Active/Inactive from the Row menu.

   The package must be inactive before you can make revisions.

3. Choose the package and then choose Build Revisions from the Row menu.

The Package Build Revisions form displays all of the build information for the selected package.

4. Click the appropriate tab to change the build options, business function options, or compression options for the package. When you are finished, click OK.
5. On Work With Package Build Definition, activate the package by choosing Active/Inactive from the Row menu.
6. Choose Submit Build from the Row menu when you are ready to begin the build process.

Compressing the Parent Package (After Building the Update)

ERP 9.0 backs up the parent package cab files when you build an update package. You must recompress the parent package before you deploy the parent package.

Copying a Built Package

After you build a package, you can copy the package definition, which includes the package record, header, and detail files. The copy function is useful in cases where you want to create a package that is similar to an existing package. In this situation, you can copy the package definition and then modify the package record, header, or detail files as needed.

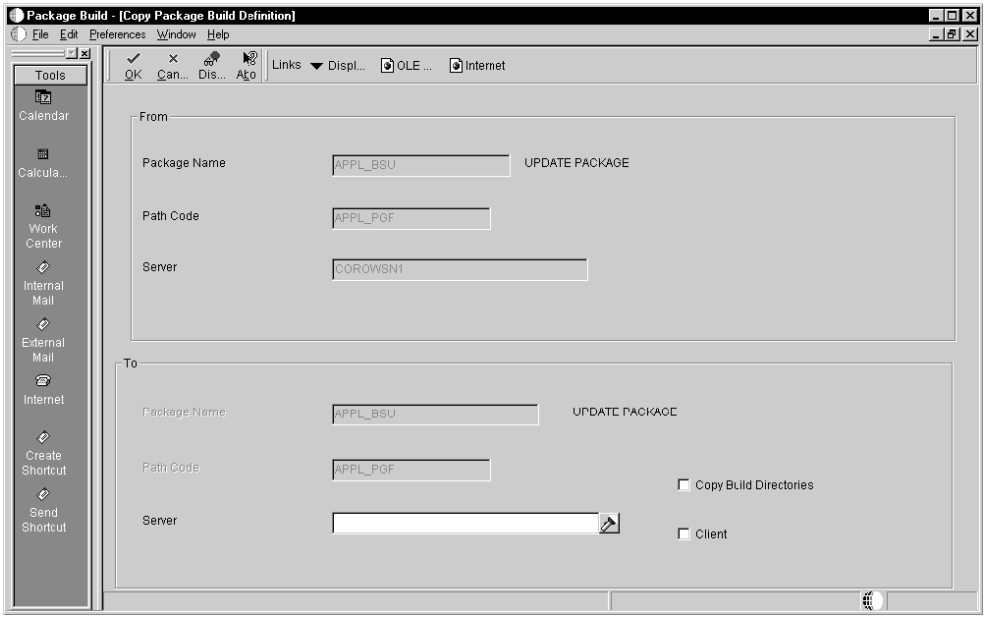
You can choose from the following two ways to copy a package:

- Copy the build definition for a specific server. You might use this method when you want to copy the same package from one server to another, or to a workstation.
- Copy the build definition for the entire package. You might use this method when you want to create a new package based on an existing package.

► To copy a package build definition from a server

From the Package and Deployment Tools menu (GH9083), choose Package Build.

1. On Work With Package Build Definition, find the package that you want to copy. Subordinate to that package, choose the server for which you want to copy the definition.
2. Click Copy.



3. On Copy Package Build Definition, enter the name of the server to which you want to copy the package.

   The system displays the package name, path code, and server name from which you are copying, as well as the destination package name and path code, which are the same.

4. Complete the following optional fields:

   - Copy Build Directories

     Turn on this option to copy the package build directories.

   - Client

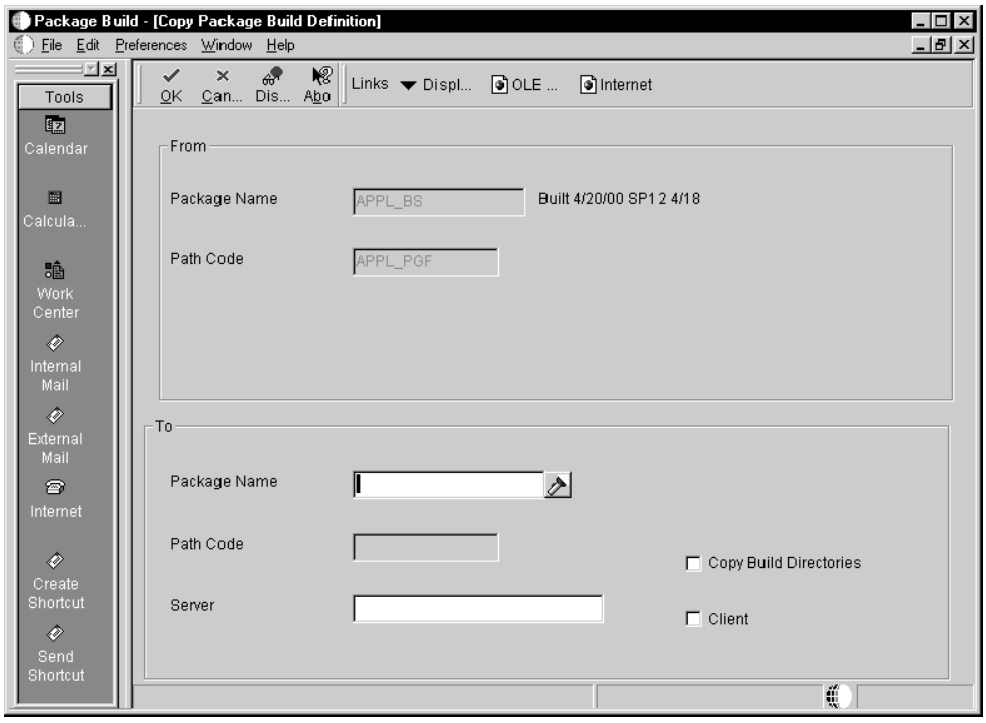
     Turn on this option to copy to the client workstation instead of to the server.

5. Click OK to return to the Work With Package Build Definition form.

► To copy a package to a new package

From the Package and Deployment Tools menu (GH9083), choose Package Build.

1. On Work With Package Build Definition, find and choose the package that you want to copy.
2. Click Copy.



3. On Copy Package Build Definition, enter the new package name and server name.

   The system displays the package name, path code, and server name from which you are copying.

4. Complete the following optional fields:

   - Copy Build Directories

     Click this option to copy the Package Build directories.

   - Client

     Click this option to copy to the client workstation instead of to the server.

5. Click OK to return to the Work With Package Build Definition form.

Viewing Package Build History and Resubmitting Builds

After you submit the package for building, you can track the build status using the Package Build History (P9622) application. This application also enables you to view logs that are associated with the build process to determine if any errors occurred during the build process.

If the build did not complete successfully, you can resubmit the package and resume building from the point where the build stopped. Alternatively, you can reset the status of the specifications and objects and then build the package again.

See Also

- *Viewing Package Build History and Logs* in the *Package Management Guide* for information about reviewing the package build history, viewing logs, resubmitting builds, and changing the package build status

Incorporating Features into Packages

A feature is a set of files or configuration options, such as registry settings, that is copied to a workstation or server to support a B9 application or other B9 functions. Like B9 objects, features are built into a package and deployed to the workstations and servers that require the feature components.

Here are some examples of features that you might want to include when you build a package:

- ActiveX controls. The B9 Application Design Aid tool allows you to include ActiveX controls in B9 applications. If ActiveX controls are delivered with B9, you need a way to copy these controls to the workstation.
- Open Data Access (ODA) data sources. ODA requires that additional ODBC data sources be created on any workstation or server that uses ODA.
- Sales Force Automation databases. The Sales Force Automation feature requires that you install a separate J.D. Edwards Supported Local Database on the workstation so that it can be disconnected from the network during offline operation. You must also write a registry setting that indicates that the machine is used offline.
- BMC Patrol, GenCorba, GenCom, and other third-party interfaces or products. Each of these products and interfaces requires additional components on the workstation and server in order to function. As B9 functionality expands to support additional third-party products and interfaces, these products will each have their own set of supporting files.

For software releases prior to ERP 9.0, custom programming was required to add feature components to the workstation and server. In EP 9.0, you can now use familiar tools such as the Package Assembly Director and the Package Build Definition Director to create a package that contains your feature, and then you can deploy it using the Package Deployment Director or multitier deployment.

Because feature components are not ERP 9.0 objects, the process for incorporating feature components into a package is slightly different from the normal package build process. Specifically, you must first define the feature before you can add it to a package.

Feature Build and Deployment Process Overview

The following overview summarizes the process for defining and adding a feature to a package:

Define the feature

Before adding the feature to a package, you must first define it using the Feature Based Deployment Director. During feature definition, you specify the feature name and type, enter a brief description, and specify installation parameters.

The forms in the Feature Based Deployment Director enable you to do the following:

- Create a file set
- Define registry settings
- Define a Windows shortcut
- Enter initialization file information
- Add ODBC data sources
- Specify the feature build sequence
- Enter information for third party products

For details about defining features, see *Understanding Features*.

Select the feature during package assembly

After you have defined the feature, it is ready to be included in a package. Use the Package Assembly Director to assemble the package as you would any other package. When you assemble the package, feature-specific forms enable you to specify the features that you want to include.

For information about assembling a package that contains features, see *Selecting Features During Package Assembly*.

Configure the feature during package build definition

After you have assembled the package that contains the features, you can use the Package Build Definition Director to define the build for the package. Forms in this director enable you to choose the file sets that will be compressed within the package, and to specify the processes that will be run before and after the feature is built.

For information about building a package that contains features, see *Configuring Features During Package Build Definition*.

Deploy the package

After you have built the package, you are ready to schedule it for deployment by using the Package Deployment Director. The procedure is the same as the procedure that you use to schedule packages that do not include features.

For information about scheduling packages for deployment, see *Deployment*.

Run Workstation Installation and Deployment Server Installation

After you have deployed the package to workstations and deployment servers, use the Workstation Installation and Deployment Server Installation applications to install the package.

For information about installing a package that contains features, see *Installing Packages Containing Features*.

Understanding the Feature Based Deployment Director

The Feature Based Deployment Director enables you to define your feature so that it can be included in a package and then deployed to workstations and servers. The forms in the director enable you to specify the name and type of the feature, as well as the different feature components.

For this release, the Platform value must always be 80 for CLIENT. Future releases will enable you to choose alternative platforms.

Throughout the feature definition process, you can always proceed to the next or previous form by clicking Next or Previous. Also, regardless of where you are in the process, you can always cancel the feature definition by clicking Cancel.

The following table summarizes each of the forms in the Feature Based Deployment Director and the function of each form:

The Welcome Form	**View this form for an introduction to the Feature Base Deployment Director**
The Feature Information form	Use this form to enter the feature name, type, and a brief description of it. You also indicate whether the feature is required when it is ready to be installed. If so, the feature will be installed automatically when the installation program runs. If the feature is optional, you can specify whether the feature is initially selected (that is, Default On) when the installation program runs. If the optional feature is not initially selected (that is, Default Off), you must manually select the feature before you can install it. You also need to select any of the following feature components that apply: • File set • Registry • Shortcut • ODBC data sources • Additional package build processes • Additional install processes • Initialization files (INI) Your selections on this form determine which forms subsequently appear. You must choose at least one feature component in order to continue defining the feature.
The File Set Definition form	Use this form to choose the file sets that you want to include in the package. A file set is a collection of files that must be installed on the workstation or deployment server in order for the feature to function correctly.
The Registry Definition form	Use this form when the feature that you are defining requires an addition, modification, or deletion of an entry in the Windows registry. Registry information that you enter on this form will be delivered in the package that contains the feature.
The Shortcut Definition form	Use this form when you want to create shortcuts on the Windows desktop as part of the installation process. When you enter shortcut information, the system automatically creates a shortcut on the desktop after the feature is installed.
The Additional Package Build Processes form	Use this form to specify whether to run a batch application or executable program either before or after the package that contains the feature is installed.
The Additional Install Processes form	Use this form when you need to install a third-party product as part of the feature installation.

The Initialization File (INI) Definition form	Use this form when you need to make changes to an initialization file, such as the JDE.ini file, as part of the installation. For example, ERP 9.0 developers often manually turn off the debug log; therefore, if you plan to deliver a package to developers, you can automatically turn off the debug log.
The ODBC Data Source Definition form	Use this form when the feature requires that additional ODBC data sources be added when the package that contains the feature is installed.
The Features Summary form	Use this form to review the information that you entered for the feature on the previous form. Primary information for the feature appears in a tree structure on the left side of the form. Options on the right side of the form enable you to access any of the previous forms so that you can make changes, if necessary.

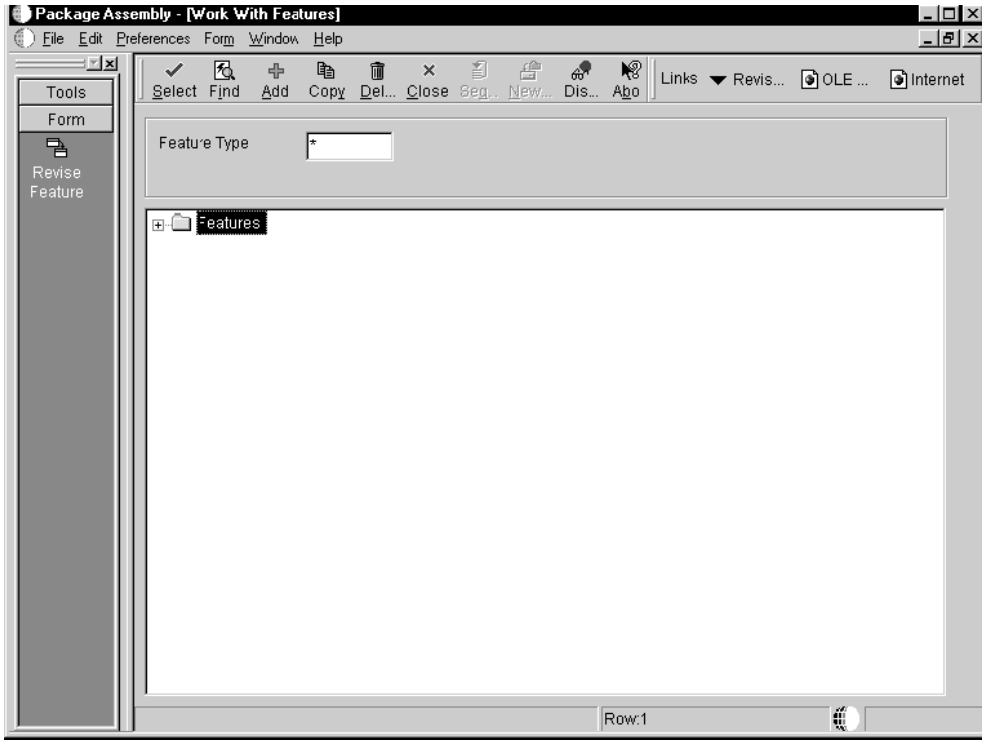
Creating a Feature

Complete the following task to define a feature and add one or more components to the feature.

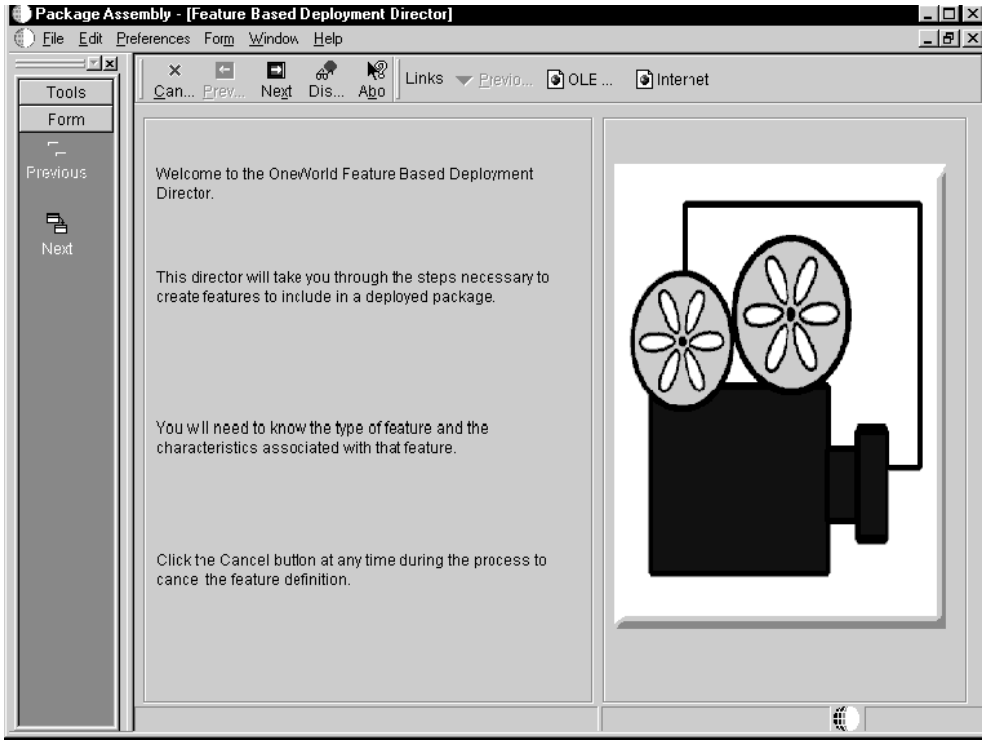
▶ To create a feature

From the Package and Deployment Tools menu (GH9083), choose Package Assembly.

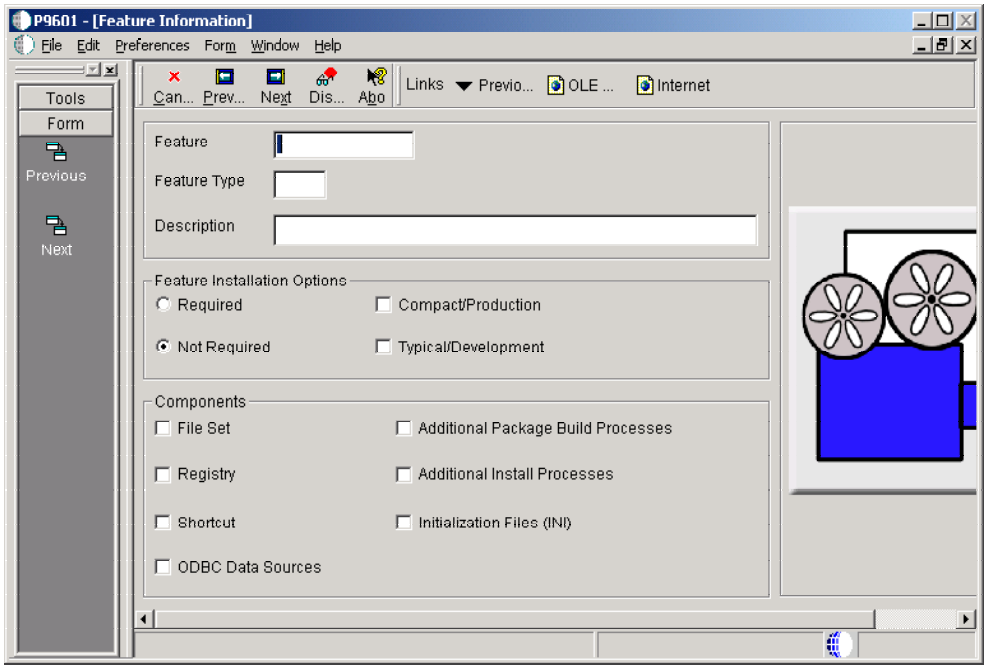
1. On Work with Packages, choose Features from the Form menu.



2. On Work With Features, click Add.



3. On Feature Based Deployment Director, click Next.



4. On Feature Information, complete the following fields:

   - Feature
   - Feature Type
   - Description

   Turn on one of the following:

   - Required

     The installation of this feature is mandatory for both Compact/Production and Typical/Development installs. Inclusion of this feature cannot be overridden when the package is installed.

   - Not Required

     The installation of this feature is optional. Whether the feature is installed depends on the options that you select below (Compact/Production and Typical/Development). Inclusion of the feature can be overridden when the package is installed.

   Turn on one or both of the options below. (If you chose “Required,” both of the following options are automatically selected.)

   - Compact/Production

     When turned on, this feature is included in a Compact/Production install by default. This option can be overridden when the package is installed if “Not Required” is also selected.

   - Typical/Development

     When turned on, this feature is included in a Typical/Development install by default. This option can be overridden when the package is installed if “Not Required” is also selected.

   Turn on any of the following options and feature components that apply to your package. The forms that subsequently appear depend on the feature components that you select.

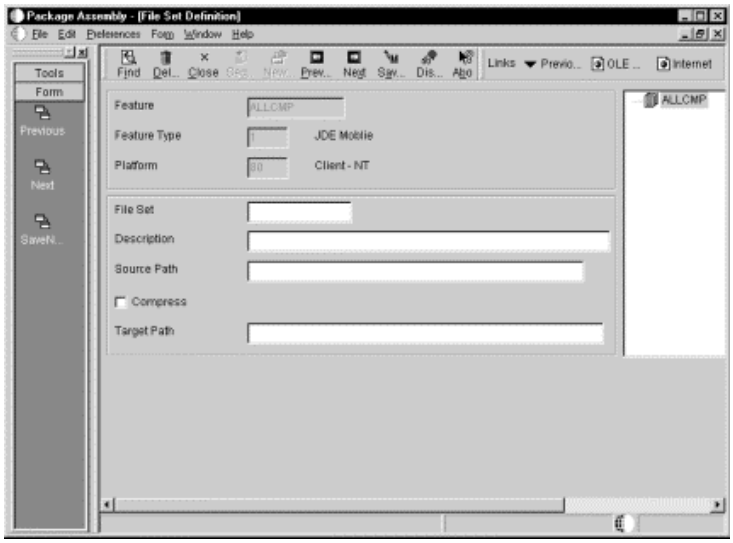
   - File Set
   - Registry
   - Shortcut
   - ODBC Data Sources
   - Additional Package Build Processes
   - Additional Install Processes
   - Initialization Files (INI)

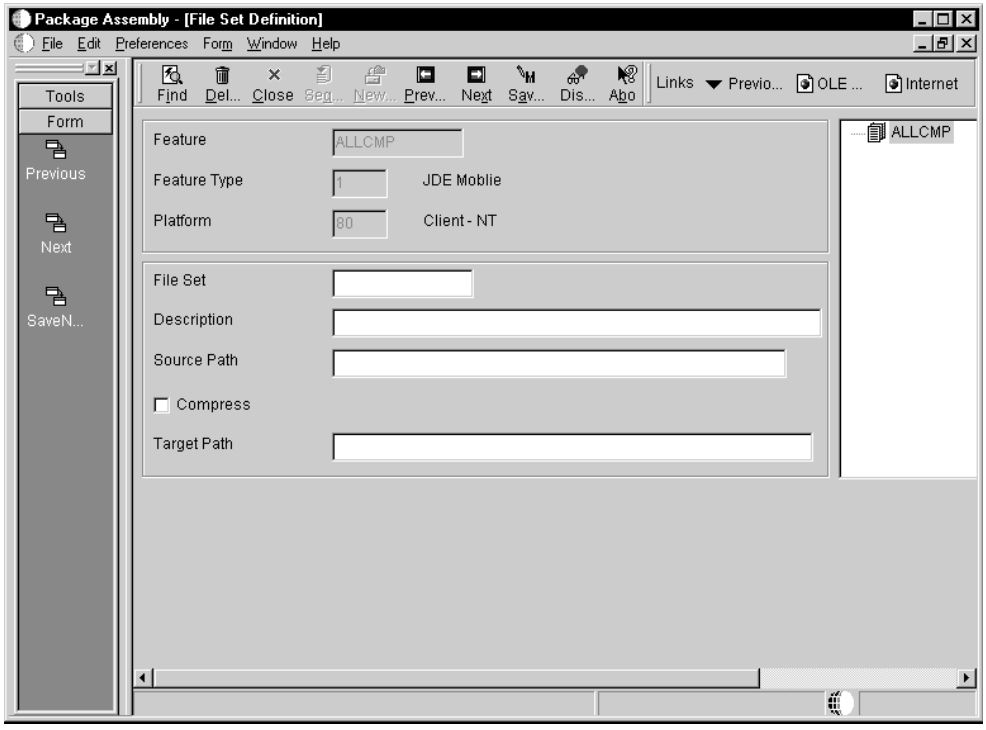
5. Click Next.

Defining a File Set Component

If you selected the File Set component, the File Set Definition form appears. Use this form to enter information about any file sets that must be installed on the workstation or server in order for the feature to function properly.

► To define a file set component





1. On File Set Definition, complete the following fields:

   - File Set

     A free-form text field for comments or memoranda.

   - File Set Description

     A description of a group of files.

   - Source Path

     A path that identifies the source location of a file set.

   - Compress

     An option to compress the file.

   - Target Path

     A path that identifies the target location of a file set.

   The source path tells ERP 9.0 where to find the file set to be copied into the package, and the target path indicates the location to which the file set should be copied when the package is installed. Although a feature can have an unlimited number of file sets, each file set can have only one target path.

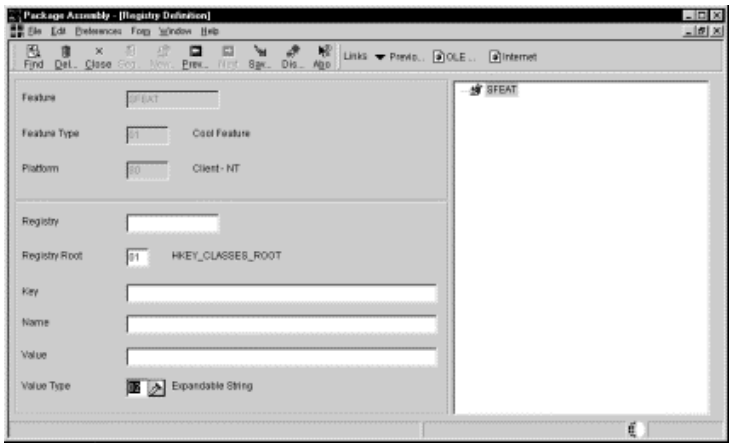
   You can also use this form to modify or delete any previously defined file sets. Existing file sets appear in the tree structure on the right side of the form. To modify a file set, select the file set on the tree structure and modify any of the fields for the file set. To delete a file set, select the file set and click Delete.

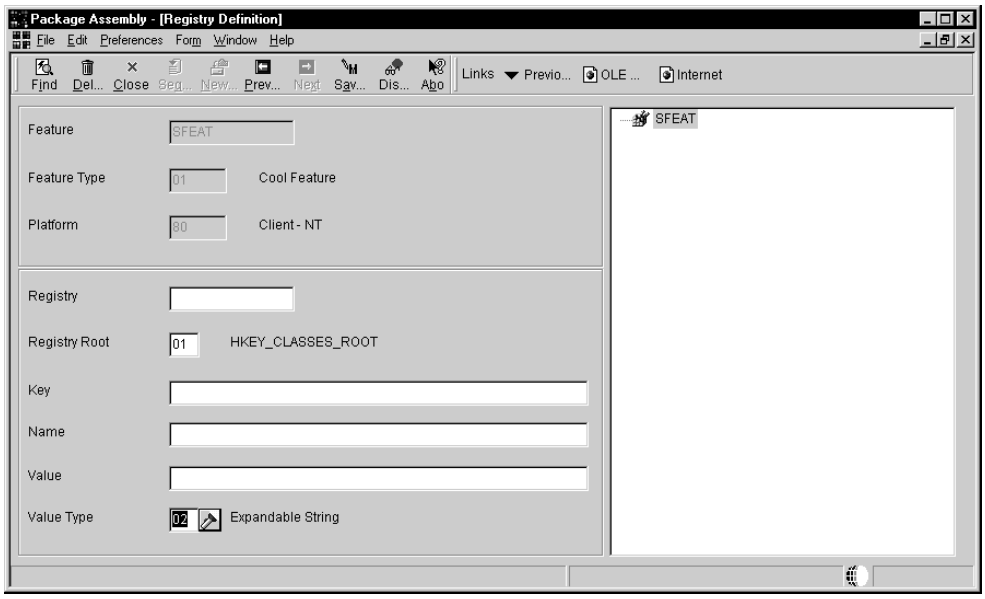
2. When you are finished adding file set information, choose Save Node from the Form menu.
3. Click Next.

Defining a Registry Component

If you selected the Registry component, the Registry Definition form appears. Use this form to enter information that should be added to the Windows registry as part of the feature installation.

► **To define a registry component**





1. On Registry Definition, complete the following fields:
   - Registry

     The identifier of a registry modification.
   - Registry Root

     The root key in the registry.
   - Key

     The key for a registry value.
   - Name

     The registry value name.
   - Value

The registry value.

- Value Type

  In the registry, the data type in which the value is stored.

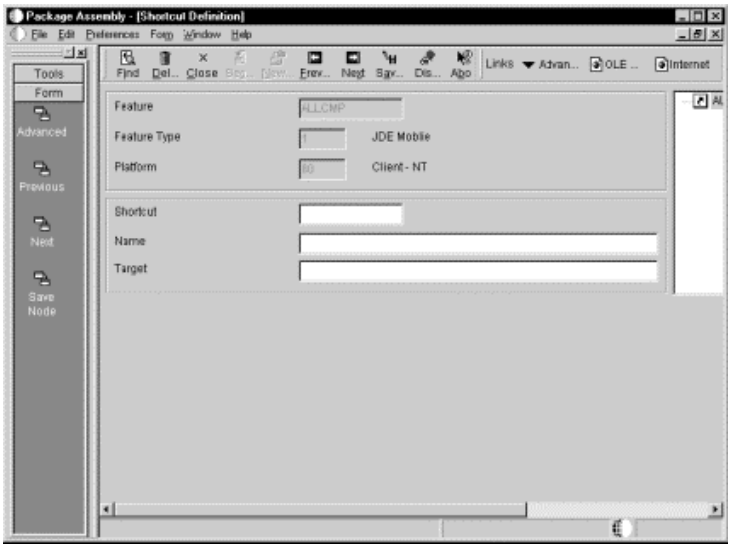
You can also use this form to modify or delete any previous registry definitions. Existing registry definitions appear in the tree structure on the right side of the form. To modify a registry definition, select the item on the tree structure and modify any of the fields for the registry definition. To delete a registry definition, select the item and click Delete.

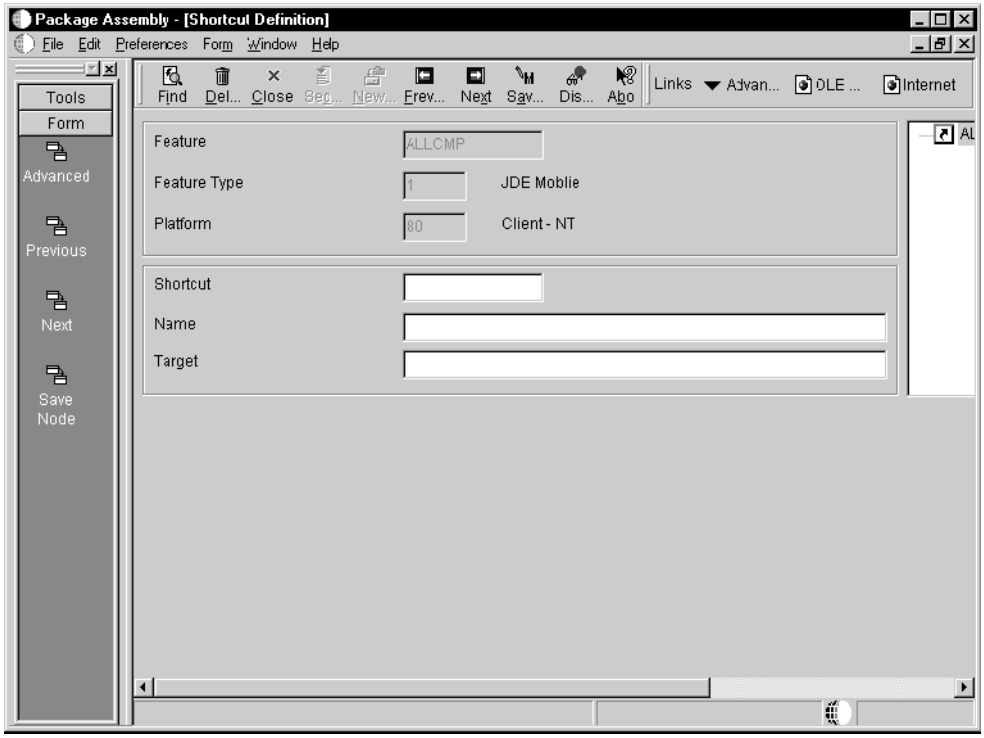
2. When you are finished adding registry information, choose Save Node from the Form menu.
3. Click Next.

Defining a Shortcut Component

If you selected the Shortcut component, the Shortcut Definition form appears. Use this form if you want to add a shortcut for your feature to the Windows desktop.

► To define a shortcut component





1. On Shortcut definition, complete the following fields:

   - Shortcut

     A name that identifies a unique shortcut to a user's computer.

   - Name

     The name of the shortcut.

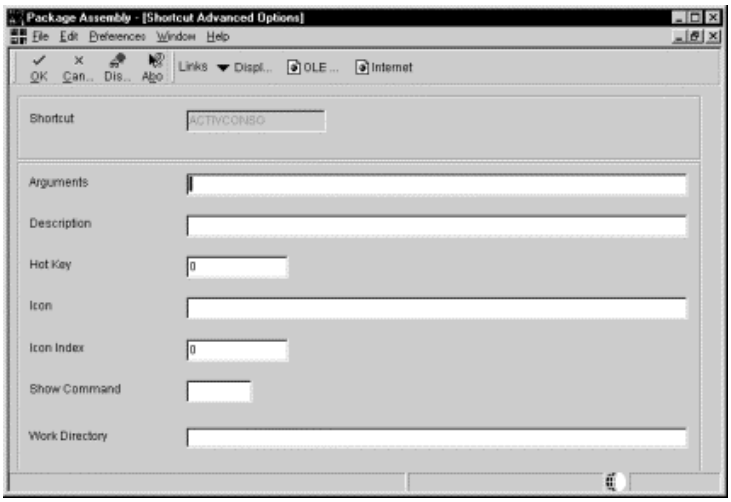
   - Target

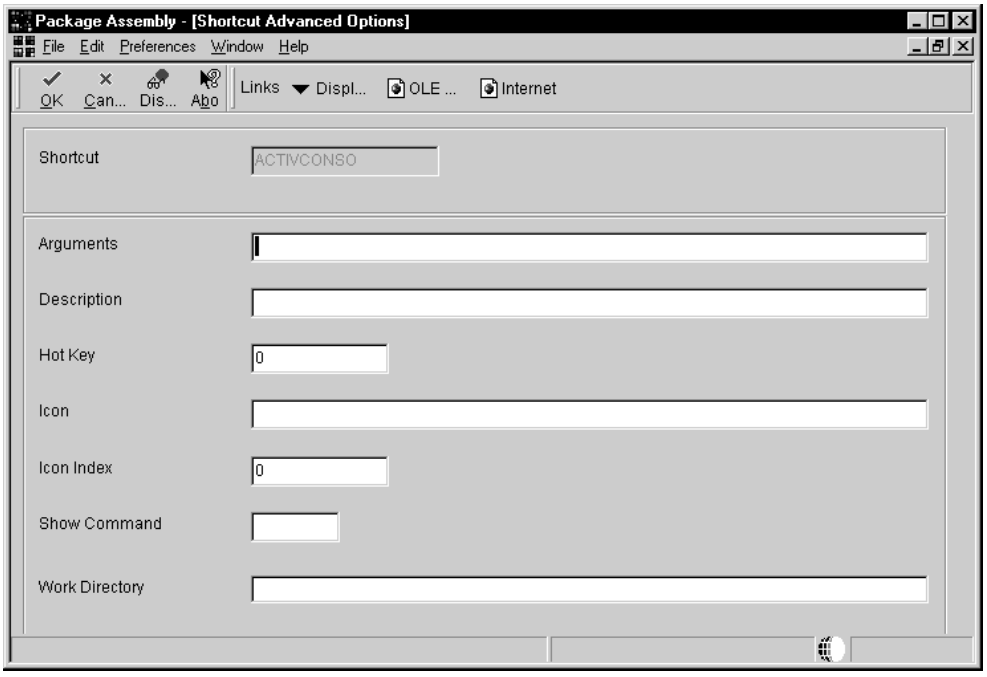
     The path and file name of a target file.

     If you enter the shortcut using a standard Internet protocol (such as http://, FTP//, or FILE//), ERP 9.0 automatically creates an Internet shortcut.

   You can also use this form to modify or delete any previous shortcut definitions. Existing definitions appear in the tree structure on the right side of the form. To modify a shortcut definition, select the item on the tree structure and modify any of the fields for the shortcut definition. To delete a shortcut definition, select the item and click Delete.

2. To enter advanced shortcut options, choose Advanced from the Form menu.





3. On Shortcut Advanced Options, complete any of the following fields:
   - Arguments

     Parameters that are entered at the command line for the shortcut.
   - Description

     The description of the shortcut.
   - Hot Key

     A key sequence that, when pressed, automatically launches the shortcut.

- Icon

  The path and name of an icon file, based on a relative target path.

- Icon Index

  The icon index for a shortcut.

- Show Command

  The size of the window after the shortcut is launched. For example, the window might be minimized or maximized.

- Work Directory

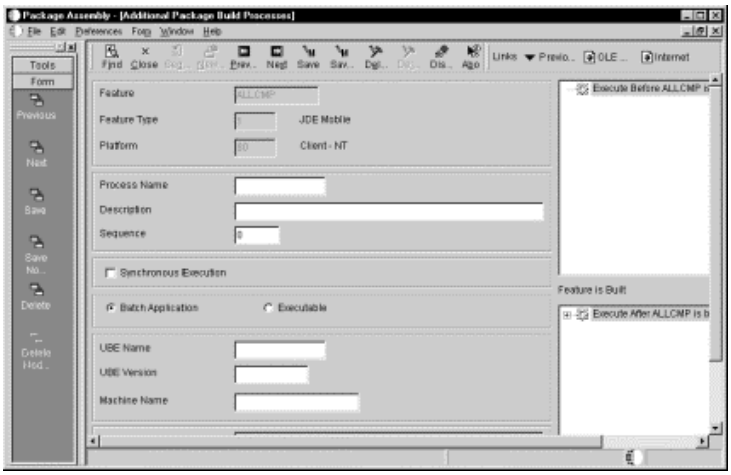
  The identifier of the directory path or the working directory of a shortcut.

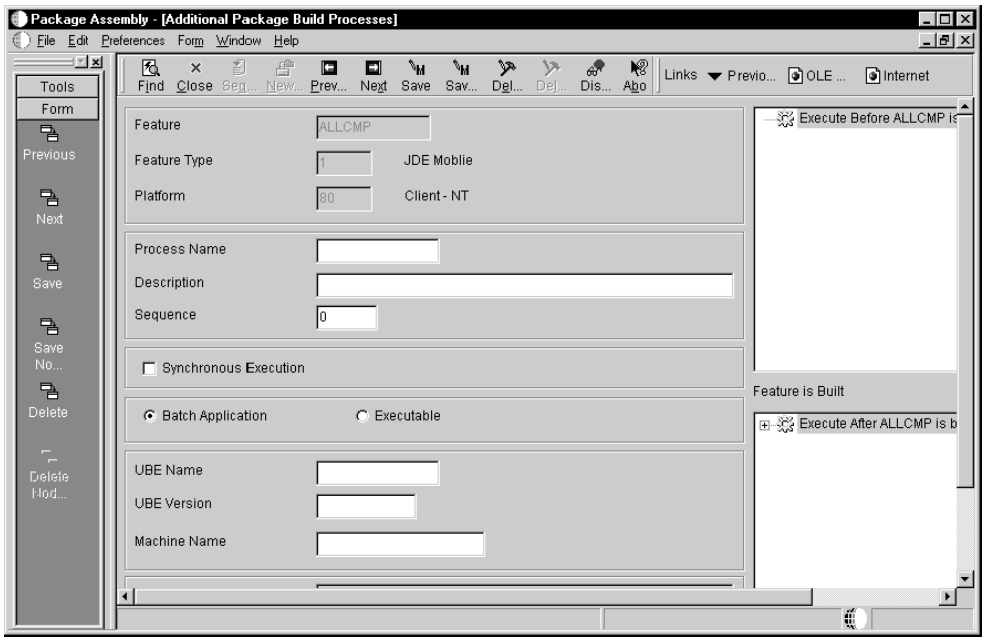
4. Click OK to save your entries and return to the Shortcut Definition form.
5. Complete the shortcut information and choose Save Node from the Form menu.
6. Add all of your shortcuts.
7. Click Next.

Defining Additional Package Build Processes

If you selected additional package build processes, the Additional Package Build Processes form appears. Complete this form to specify UBEs or executables that should run when the package is built.

► To define additional package build processes





1. On Additional Package Build Processes, complete the following fields:

   - Process Name

     Name of the build process.

   - Description

     Description of the build process.

   - Sequence

     A number that identifies the order in which the process will be run relative to the other processes that run during the package build.

   - Synchronous Execution

     An option that indicates whether the package build job waits for the process to finish before it continues.

   - Batch Application or Executable

     Specify whether the process is an application or an executable

2. If you selected Batch Application, complete the following fields:

   - UBE Name

     The name of the ERP 9.0 UBE.

   - UBE Version

     The version number of the UBE.

   - Machine Name

     The name of the server or workstation on which the UBE will run.

3. If you selected an executable program, complete the following fields:

   - Executable Name

     The name of the executable program that the system launches to install the third-party software.

   - Target Path

     The path and file name of a target file.

   - Parameters

     The executable parameters that the setup program uses to install the third-party software.

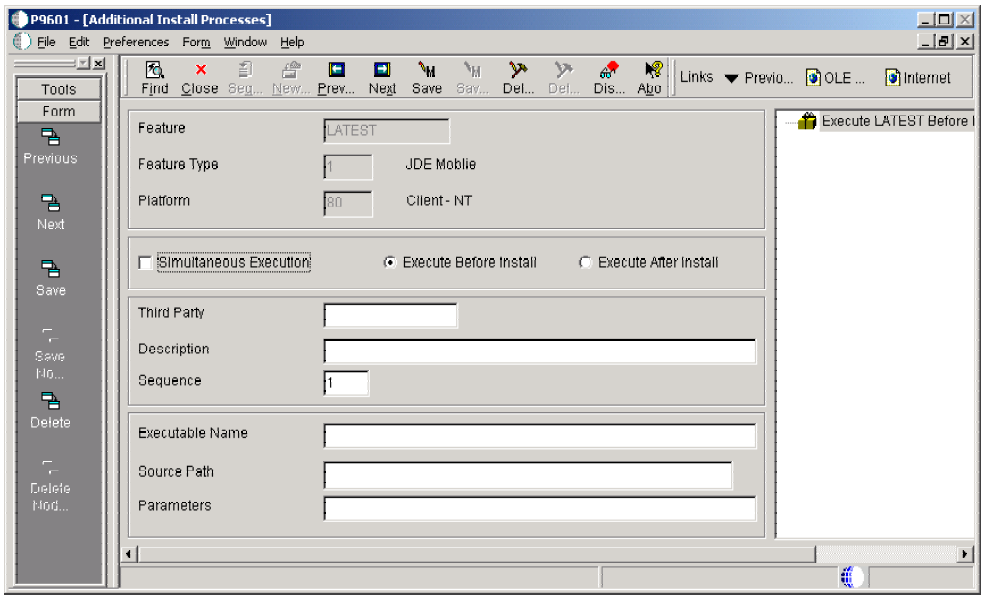
   You can also use this form to modify or delete any previously defined processes. Existing processes appear in the tree structure on the right side of the form. To modify a process definition, select the item on the tree structure and modify any of the fields for the definition. To delete a process definition, select the item and then choose Delete or Delete Node After from the Form menu, depending on whether you want to delete a process that is executed before or after the feature is installed. You can run the process either before or after the feature is built.

4. When you are finished adding process information, choose either Save or Save Node After from the Form menu, depending on when you want the process to run.
5. Click Next.

Defining Additional Install Processes

If you chose the option to add additional installation processes, the Additional Install Processes form appears. Use this form to enter information about third-party applications that should be run when the package is installed.

► To define additional install processes



1. On Additional Install Processes, complete the following fields:
   - Third Party

     The name of the third-party component.

   - Description

     A description of third-party software.

   - Sequence

     A number that identifies the order in which this process will run relative to the other additional install processes.

2. The following options require you to choose two fields that interact with each other. Select the combination for the desired outcome:
   - Synchronous and Execute After Install

     Simultaneous Execution turned off and Execute After Install option turned on. The third-party process waits for the J.D. Edwards client install to finish before running.

   - Synchronous and Execute Before Install

     Simultaneous Execution turned off and Execute Before Install option turned on. The J.D. Edwards client install will run the third-party process and wait until it finishes before installing the client.

   - Asynchronous and Execute After Install

     Simultaneous Execution turned on and Execute After Install option turned on. The J.D. Edwards client install finishes, then starts the third-party process. Neither process waits for the other to finish before proceeding.

   - Asynchronous and Execute Before Install

     Simultaneous Execution turned on and Execute Before Install option turned on. The J.D. Edwards client install begins, then immediately starts the third party process and resumes the client install without waiting for the third-party process to finish.

3. Complete the remaining fields:
   - Executable Name

     The name of the program that launches the third-party software.

   - Target Path

     The path to the executable file. Do no include the name of the file.

   - Parameters

     The executable parameters that the system passes to the third-party program.

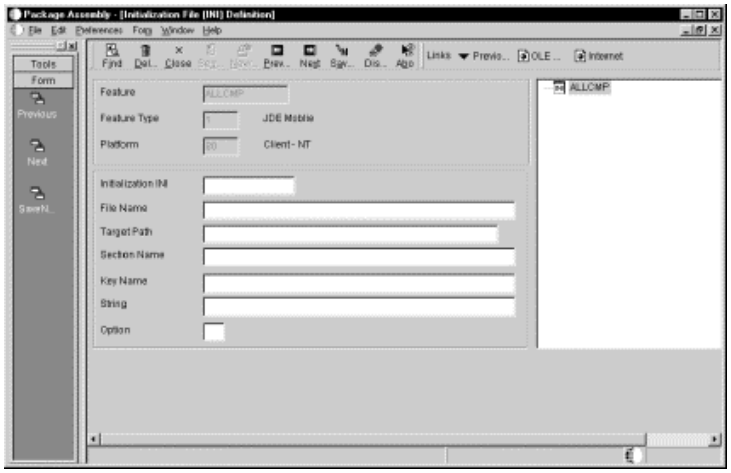
4. When you finish adding third-party product information, choose Save from the Form menu.
5. To delete a definition for a third-party product, select the item on the tree structure and then choose Delete from the Form menu.

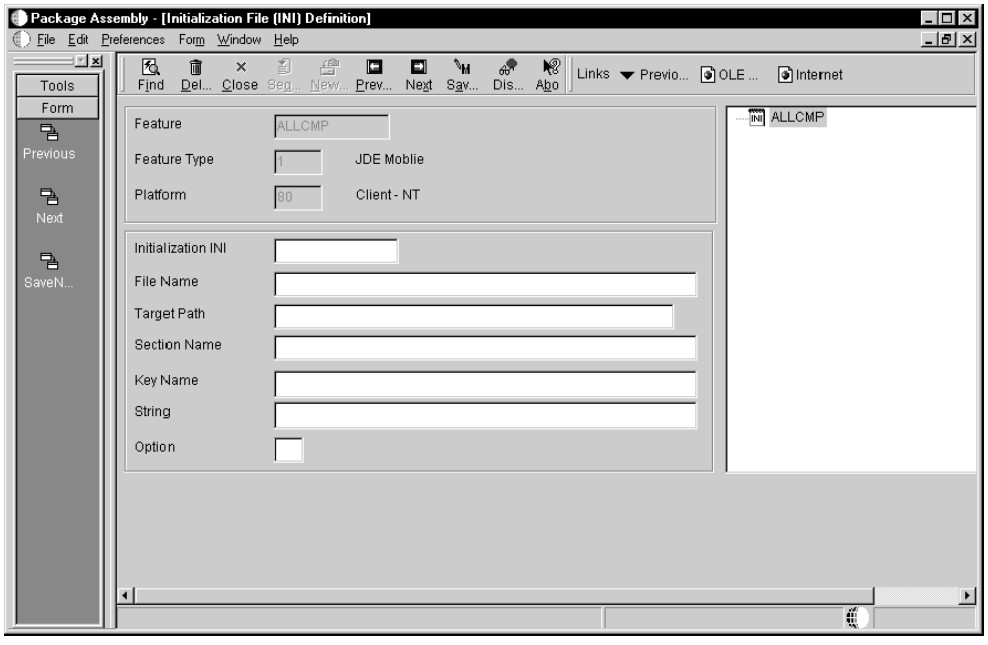
6. Click Next.

Defining an Initialization File Component

If you selected the initialization file component, the Initialization File (INI) Definition form appears. Use this form to enter any information that should be written to an initialization file (such as JDE.ini) as part of the feature installation.

► To define an initialization file component





1. On Initialization Files (INI), complete the following fields:
   - Initialization INI

The identifier of an initialization file component.

- File Name

  The name of the initialization file.

- Target Path

  The path of the INI file.

- Section Name

  The name of the application section in an initialization file.

- Key Name

  A key in the initialization file that is to be added, modified, or removed.

- String

  The value of the key in an initialization file.

- Option

  The option that identifies the action associated with the key in the initialization file.

You can use this form to modify or delete any previous initialization file definitions. Existing definitions appear in the tree structure on the right side of the form. To modify an initialization file definition, select the item in the tree structure and modify any of the fields for the definition. To delete an initialization file definition, select the item and click Delete.

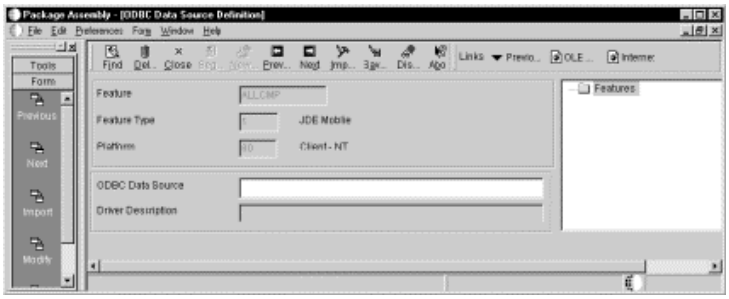
2. When you finish adding initialization information, choose Save Node from the Form menu.
3. Click Next.

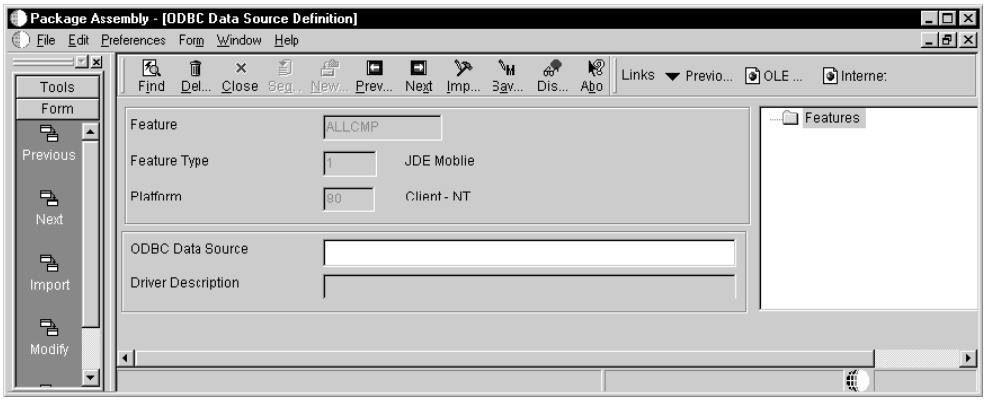
Defining an ODBC Data Source Component

If you selected the ODBC Data Sources component, the ODBC Data Sources Definition form appears. Use this form to enter information for any ODBC data sources that must be added to support the feature.

You can either create new ODBC data sources or import existing data sources:

► To create new ODBC data sources





1. On ODBC Data Sources Definition, complete the following field:

   - ODBC Data Source

     The name of the ODBC Data Source

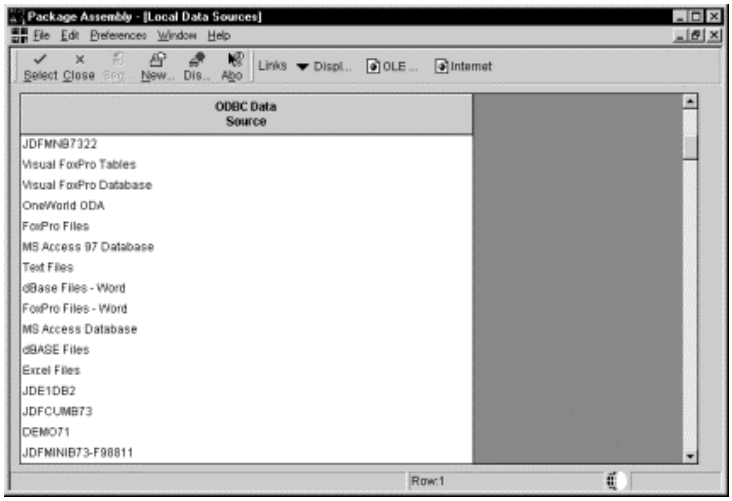
2. Choose Save Node from the Form menu.

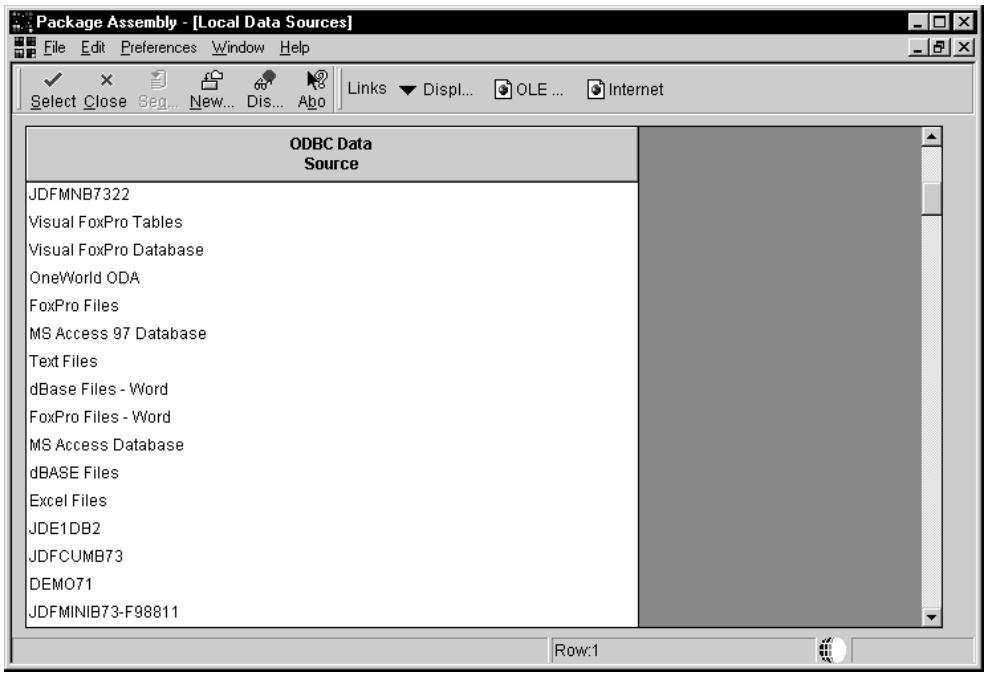
   ERP 9.0 activates the Windows control panel applet that displays the ODBC Data Source forms.

3. Enter the data source information on the Windows-based forms.
4. Repeat this process to create additional data sources.

► To import existing ODBC data sources

1. On ODBC Data Sources Definition, choose Import from the Form menu to display the Local Data Sources form. This form lists all previously created data sources that reside locally on your machine.





2. On Local Data Sources, use the Control or Shift key to choose one or several data sources, and click Select to add the data sources to the feature.

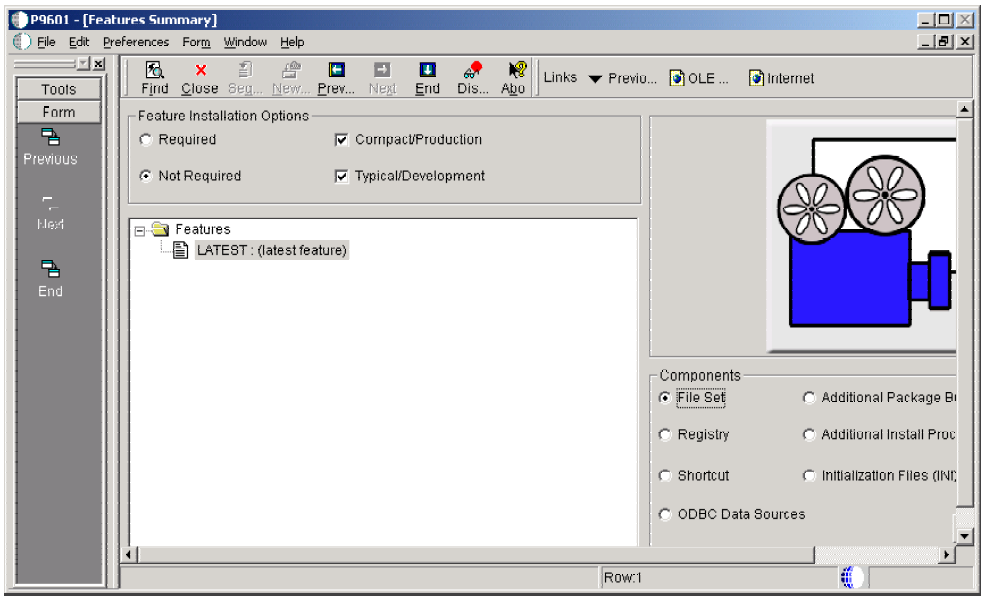
   The ODBC Data Source Definition form reappears.

3. When you are finished adding data source information, choose Save Node from the Form menu.
4. Click Next.
5. To modify existing data sources, enter the data source name and then choose Modify from the Form menu. The ODBC Data Source Revisions form appears. Use this form to make changes to the data source.
6. When you are finished, click OK to return to the ODBC Data Source Definition form.

Reviewing the Feature Components

After defining all the selected components, you can review and modify information that you entered on any of the Feature Based Deployments forms. For more information about revising feature components, see *Revising an Existing Package*.

▶ To review the feature components



1. On the Features Summary form, choose a component in the right pane and click the Revise button to review the information for that component.
2. If needed, change the field values for the selected component and click the Save button.
3. Repeat the previous steps to modify other components.
4. When you are finished defining the feature, click End on the Feature Summary form.

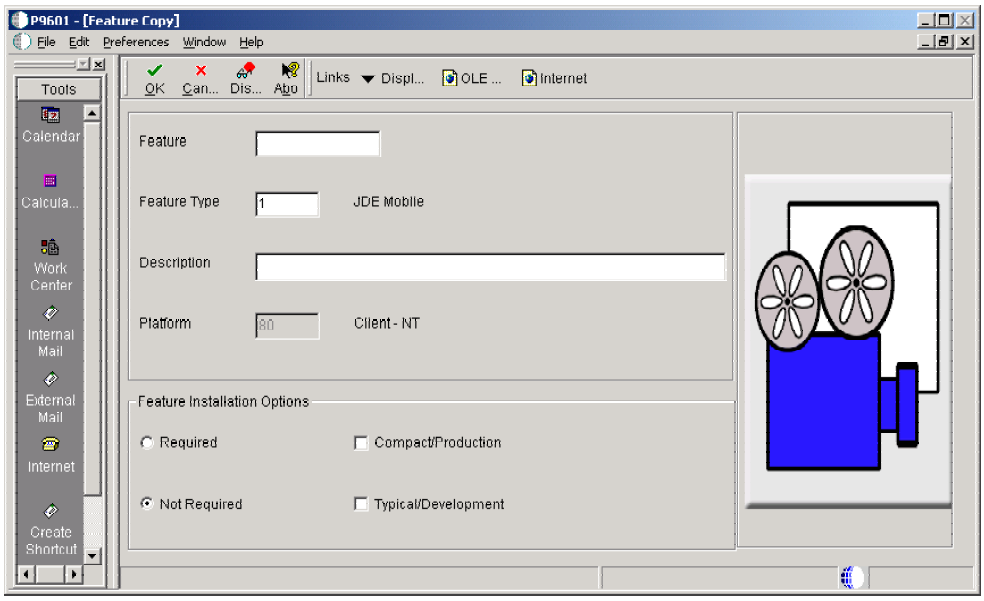
Copying Features

The Feature Based Deployment Director includes a copy function that enables you to copy an existing feature and rename it as a new feature. This feature is especially useful if you want to create a feature definition that closely matches an existing feature definition.

▶ To copy a feature

From the Package and Deployment Tools menu (GH9083), choose Package Assembly (P9601).

1. On Work with Packages, choose Features from the Form menu.
2. On Work With Features, select the feature for which you want to copy the definition, and then click Copy.



3. On Feature Copy, complete the following fields:
   - Feature
   - Feature Type
   - Description
4. Turn on one of the following:
   - Required

     The installation of this feature is mandatory for both Compact/Production and Typical/Development installs. Inclusion of this feature cannot be overridden when the package is installed.

   - Not Required

     The installation of this feature is optional. Whether the feature is installed depends on the options that you select below (Compact/Production and Typical/Development). Inclusion of the feature can be overridden when the package is installed.

5. Turn on one or both of the options below. (If you chose “Required,” both of the following options are automatically turned on.
   - Compact/Production

     When turned on, this feature is included in a Compact/Production install by default. This option can be overridden when the package is installed if “Not Required” is also turned on.

   - Typical/Development

     When turned on, this feature is included in a Typical/Development install by default. This option can be overridden when the package is installed if “Not Required” is also turned on.

6. Click OK to return to the Work With Features form.

7. To revise the new feature definition, choose the feature and choose Revise Feature from the Form menu.

Adding Features During Package Assembly

The Package Assembly Director includes a form called Feature Component that enables you to add defined features to a package. You can add a feature to either a new package or an existing package that is open for revision.

The following tasks describe only the process for adding features to a new or existing package. For more detailed information about assembling or modifying an assembled package, see *Assembling Packages*.

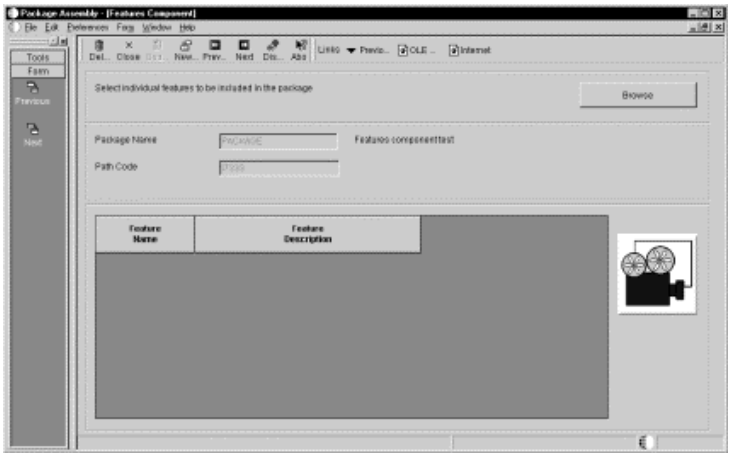
Note

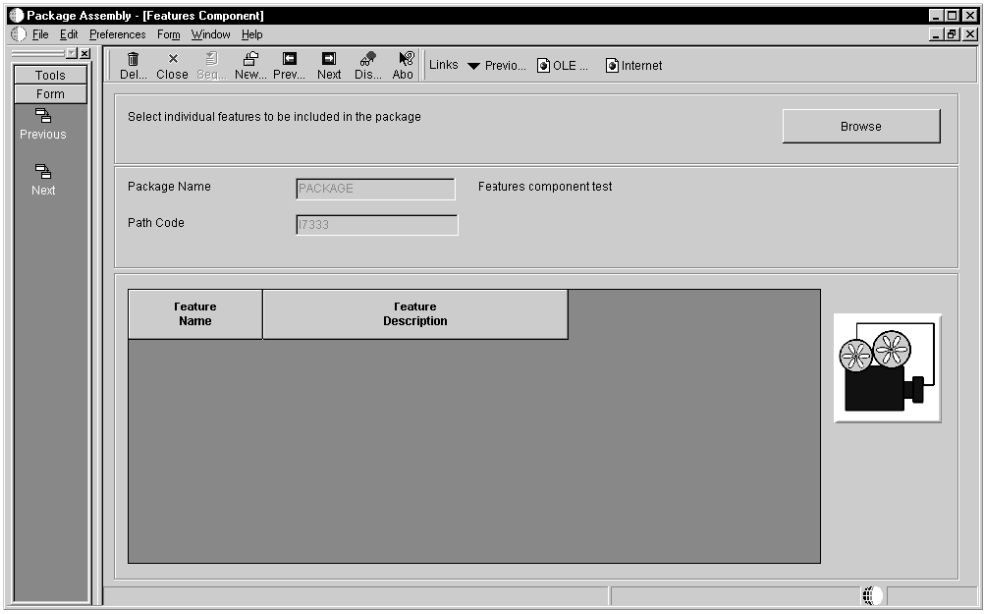
Before the feature is available for inclusion in the package, you must first define the feature as described in *Understanding Features*.

▶ To add existing features to a new package during package assembly

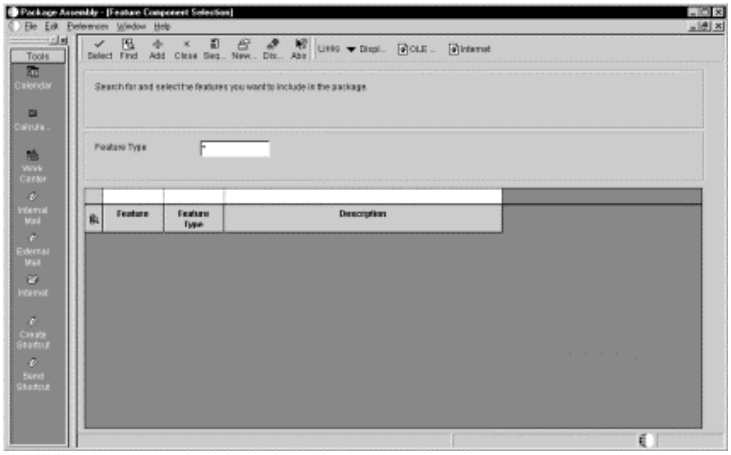
From the Package and Deployment Tools menu (GH9083), choose Package Assembly (P9601).

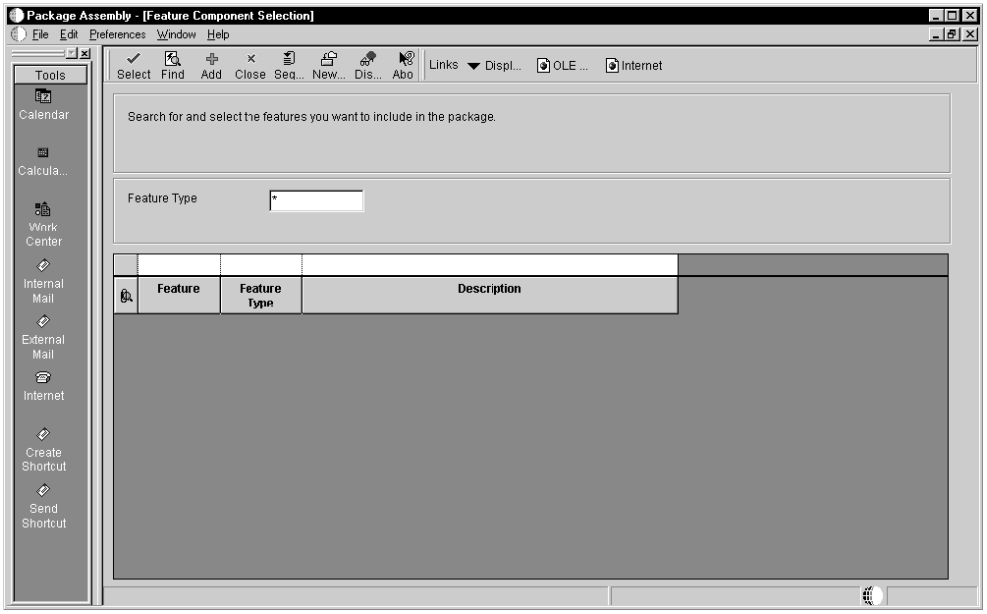
1. On Work with Packages, click Add to create a new package.
2. Enter the forms in the Package Assembly Directory and follow the steps in *Assembling a Package* until the Feature Component form appears.





3. To add a feature, click Browse.





On Feature Component Selection, click Find to display the list of available features.

4. Use one of the following methods to select one or more features to include in your package:

- Choose a feature and click the Select button. (Use the Control or Shift key to select multiple features.)
- Double-click each feature.

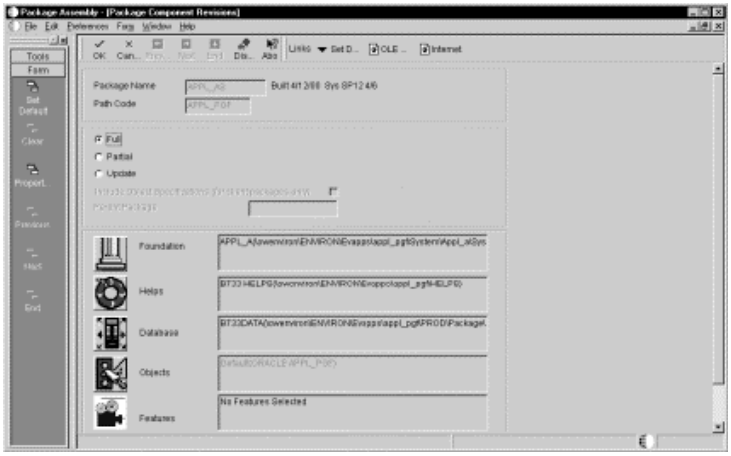
5. When you are finished adding features, click Close to return to the Features Component form. The selected features will appear.
6. Click Next and complete the remaining forms to finish assembling the package.

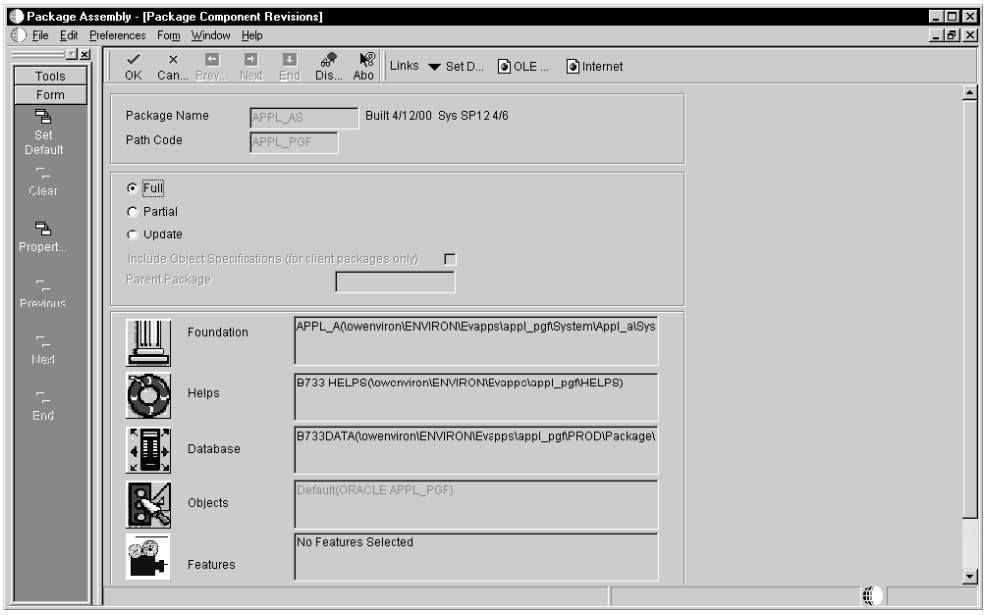
► To add features to an existing assembled package

From the Package and Deployment Tools menu (GH9083), choose Package Assembly (P9601).

1. On Work with Packages, to add a feature to an existing package that is available for revision, choose the package and then choose Package Revisions from the Row menu.

   The Package Component Revisions form appears.





2. Click the Features button.
3. To add a feature, click Browse.
4. On Feature Component Selection, click Find to display defined features.
5. Use one of the following methods to select one or more features that you want to add:
   - Choose a feature and click the Select button. (Use the Control or Shift key to select multiple features)
   - Double-click each feature.

     A check mark appears in the left column of each feature that you selected.
6. When you are finished adding features, click Close to return to the Features Component form.

7. Click Close to return to the Package Component Revisions form.
8. Click OK to return to the Work With Packages form.

Note about Deleting Features

To delete a feature that was previously included in the package, on Features Component, select the feature and then click Delete.

Configuring Features During Package Build Definition

The Package Build Definition Director includes the Build Features form, which enables you to specify whether the system builds feature INF files for the features in the package. If you defined a fileset component in your feature, you can choose to compress it. If any additional package build processes are included in the feature, you must click Build Processes and select them before they will run during package build.

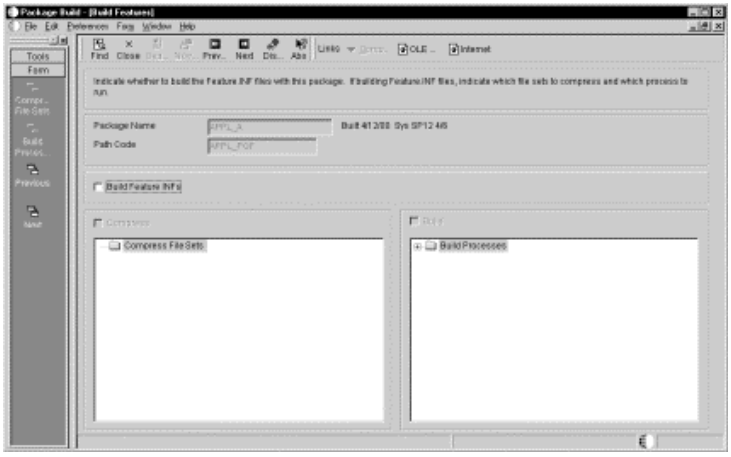
Note

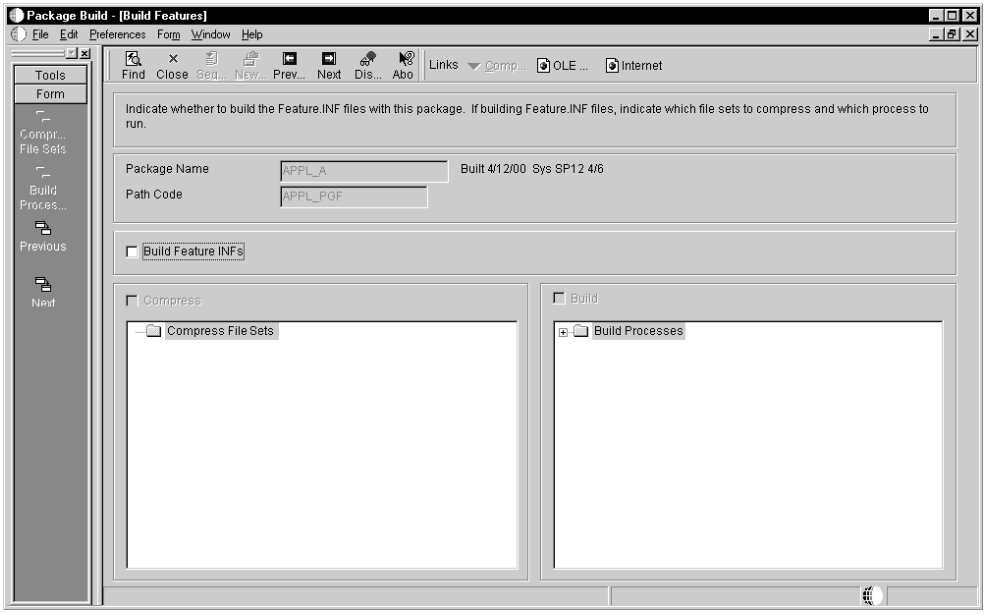
Before you can enter feature information for the build definition, the features must have already been added to the package as described in *Adding Features During Package Assembly*.

► To configure features for a new package build definition

From the Package and Deployment Tools menu (GH9083), choose Package Build.

1. On Work with Package Build Definition, click Add to launch the Package Build Definition Director.
2. Click Next and complete the task To define a package build until you come to the Build Features form.





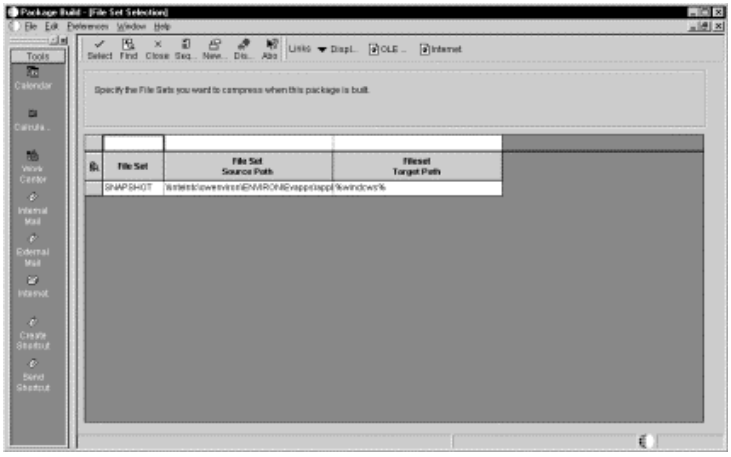
3. If you want to build a feature.inf file with the package, choose Build Feature INFs.

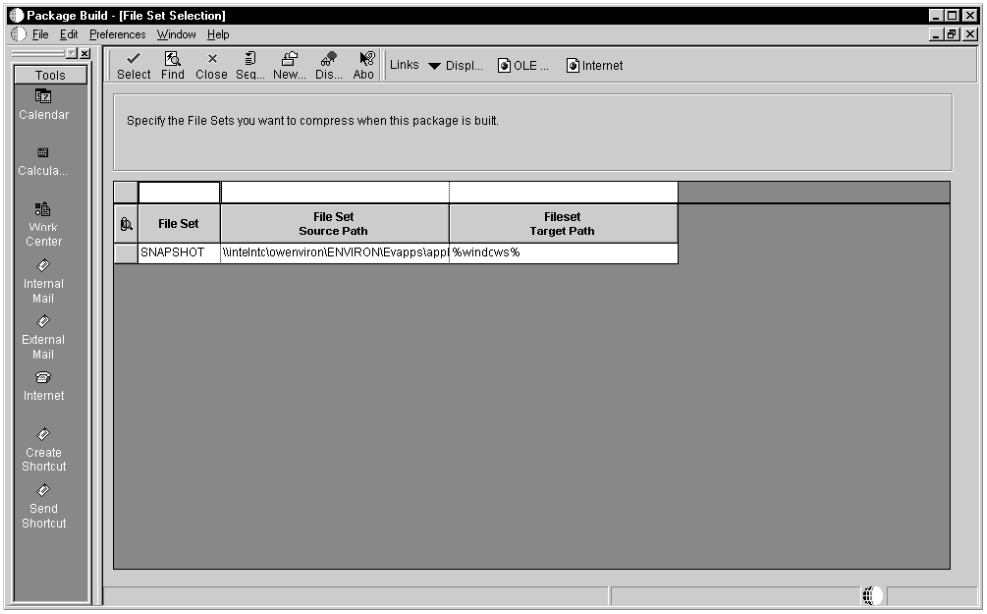
   When you choose this option, the Compress and Build fields become available if file sets or additional package build process components are included in the package.

4. Continue with one or both of the following tasks:
   - To compress file sets
   - To build ERP 9.0 processes

► To compress file sets

1. Choose Compress, and then choose Compress File Sets from the Form menu.

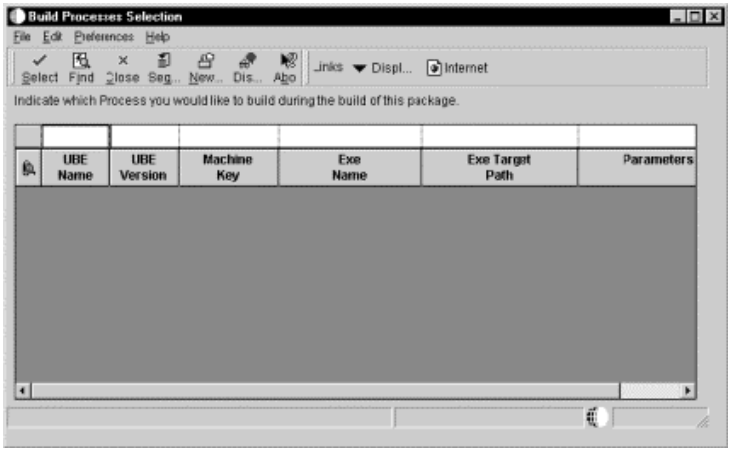


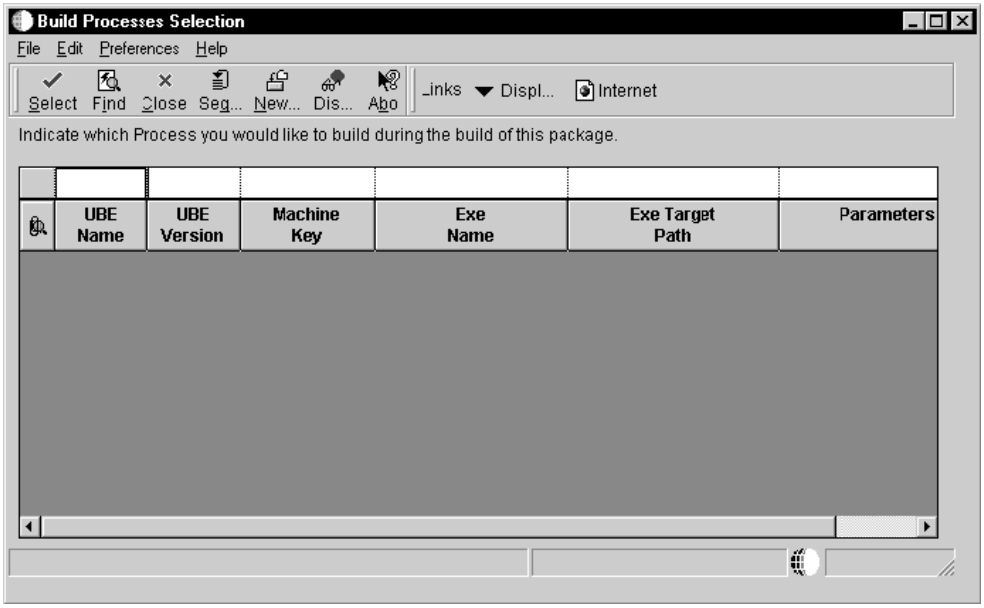


2. On File Set Selection, choose each feature that you want to include by choosing a file set and clicking Select.
3. When you are finished selecting file sets, click Close.
4. Continue either by completing the steps To build ERP 9.0 processes, or by clicking Next and completing the remaining forms to finish defining the package build.

► To build ERP 9.0 processes

1. To build processes, choose Build, and then click Select Build Processes.





2. On Build Processes Selection, select each process you want to build by choosing a process and clicking Select.
3. When you are finished selecting processes to build, click Close.
4. From the Form menu, choose Build Processes and manually select each process to run during the package build.

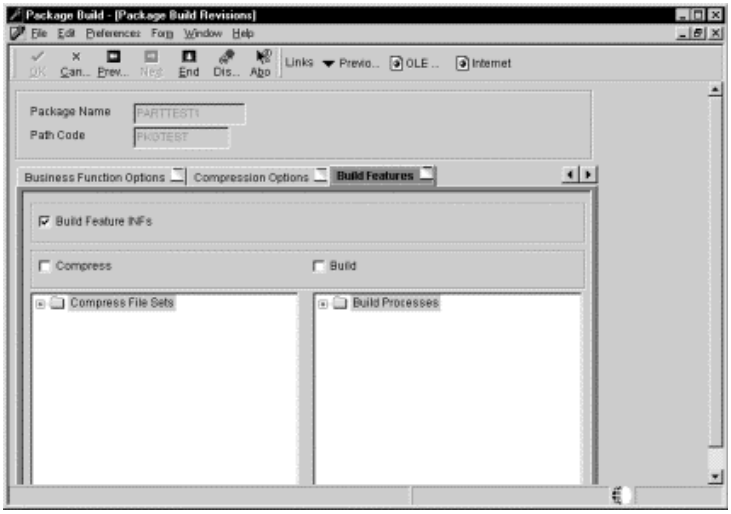
   You must complete this step or none of the processes will run, even though they are included in the feature.

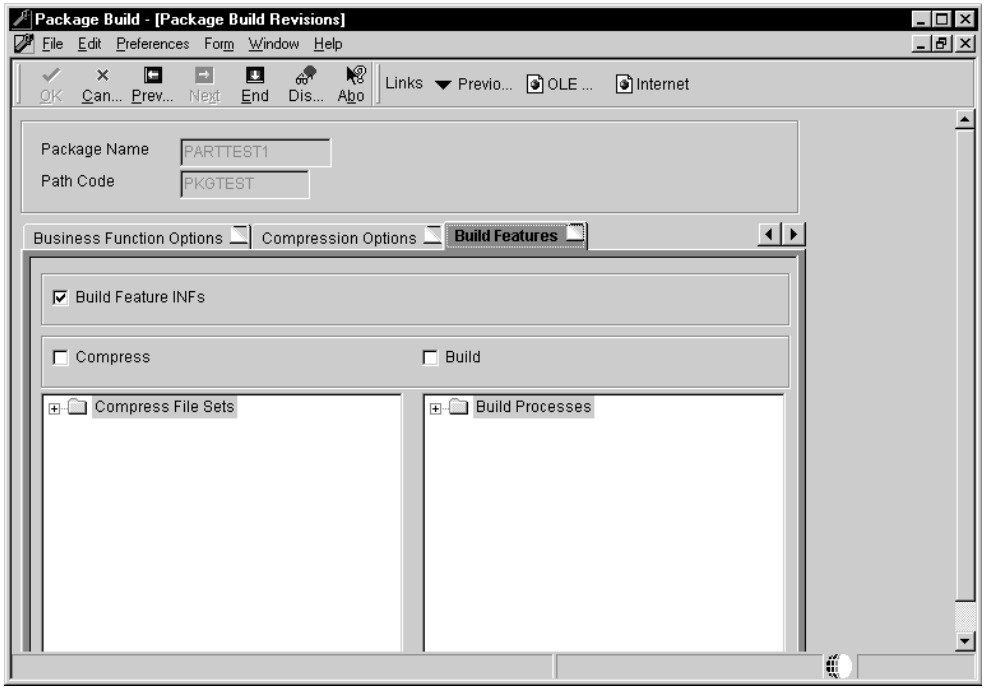
5. Click Next and complete the remaining forms to finish defining the package build.

► To configure features for an existing package build definition

From the Package and Deployment Tools menu (GH9083), choose Package Build (P9621).

1. Find and select the package that contains features.
2. Choose Build Revisions from the Row menu.
3. On Package Build Revisions, click the Build Features tab.





4. Modify or add to any of the following existing build feature settings:
   - Build Feature INFs
   - Compress
   - Build
5. If you choose Compress, choose Revise File Sets from the Form menu to modify file sets.

   The File Set Selection form appears.

6. When you are finished modifying file sets, click Close to return to the Package Build Revisions form.
7. If you chose Build, click Revise Processes to modify processes.

   The Build Processes Selection form appears.
8. When you are finished modifying processes, click Close to return to the Package Build Revisions form.
9. If you selected Build, from the Form menu, choose Build Processes and manually select each process to run during package build.

   You must complete this step or none of the process will run, even though they are included in the feature.
10. Click OK to complete the package build definition.

Installing Packages Containing Features

You install packages containing features on workstations and servers in the same way in which you install any other package: through the Workstation Installation and Deployment Server Installation applications.

When you launch either of these installations, you can turn on the Custom option to select the features that you want to install.

See Also

- The *ERP 9.0 Installation Guide* for more information about installing Workstations and Deployment Servers

Understanding Feature Entries in the Package.INF File

When a package contains a feature, the Package.INF file [Features] section provides the feature name and the location of the feature.INF file that the system creates for each feature. The feature.INF file contains information pertaining to the feature, such as shortcut information, registry settings, initialization file settings, and environment information.

See Also

- *Understanding Package INF Files* in the *Package Management Guide*
- *Understanding Feature INF Files* in the *Package Management Guide*

Viewing Package Build History and Logs

The Package Build History program (P9622) allows you to view information pertaining to the build process, including the options and objects that you specified when you created the build definition. This program provides the following build information:

- Package name
- Path code
- Date and time built
- Name of the server for which the package was built

- Current build status and status description
- Current status of selected specification tables
- Number of specifications written
- Package records written and read

The View Logs option on the Form menu allows you to view four logs that contain additional information about the build process. Refer to these logs in the event that the build does not finish successfully and you need to review the errors that occurred during the build.

If a build does not finish successfully, you can use the Resubmit Build option to resume the build from the point at which the process stopped. Only the business functions and objects that did not build successfully will be built; the entire package will not be rebuilt.

In some cases, if a build is interrupted or otherwise unable to finish, you might need to reset the build status from Build Started to Build Definition Complete. Unlike the Resume Build feature, which continues the build from the point at which it failed, resetting the status enables you to start the build process from the beginning.

Viewing the Package Build History

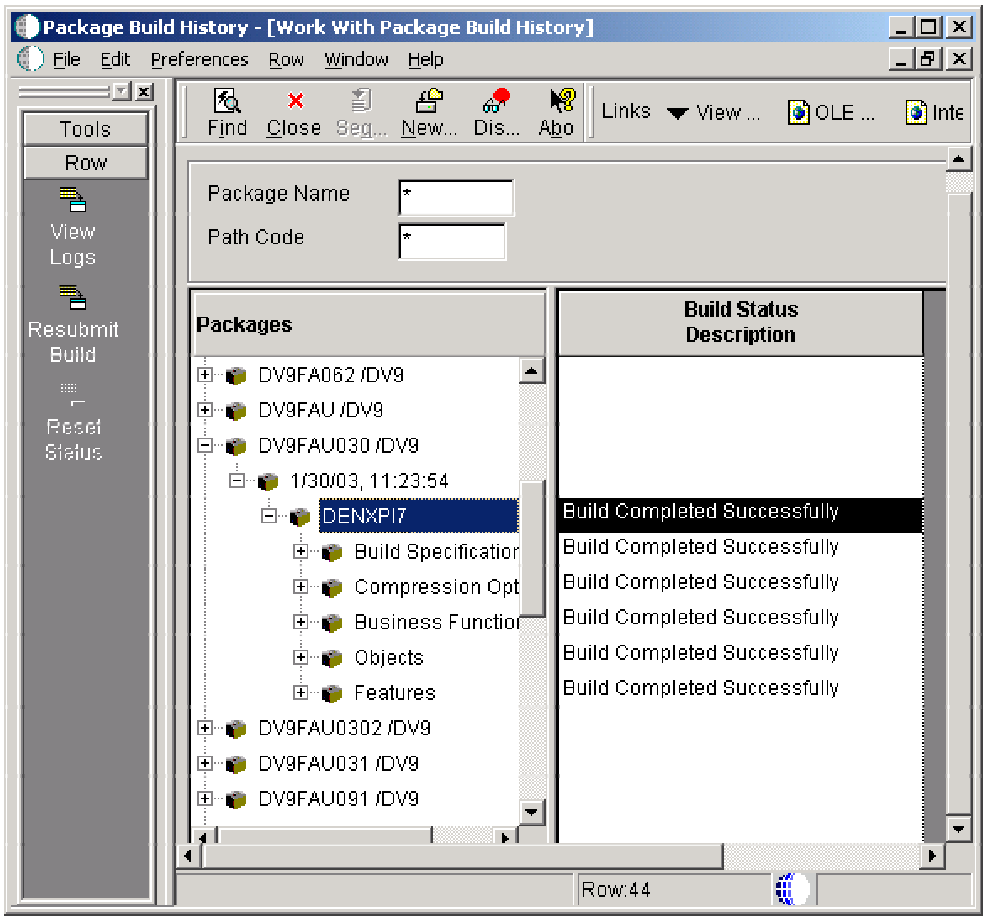
ERP 9.0 maintains a history of the package build in the Software Package Build Detail - History table (F96225). This table contains details about any package components that failed to build successfully.

If you encountered errors during the build process and your package failed to build successfully, you can resubmit the package and continue building at the point at which the build failed. In this situation, ERP 9.0 reviews the F96225 table and rebuilds only the business functions or other package components that have a status of Not Built or Error. ERP 9.0 builds only the package components that failed to build successfully, not the entire package. This feature can save you a tremendous amount of time, especially if only a few package components failed to build successfully.

If you originally specified package compression, when you resubmit the package to resume building, the system automatically compresses the directories after it successfully builds the package.

► To view package build history

From the Package and Deployment Tools menu (GH9083), choose Package Build History (P9622).



1. On Work with Package Build History, choose CLIENT or the server name to display information about the current build status for those computers. You can also expand the tree to view the following information:

   - Build specification options
   - Compression options
   - Business function options
   - Objects

   These options and objects are those that you specified when you created the build definition for the package. For example, if you chose to build only selected specifications, you can determine the status for each specification, as well as other pertinent information.

2. When you are finished viewing build history information, click Close.

Viewing Logs

After you build your package, you can view logs that list any errors that occurred during the build process. In particular, you can view the following logs:

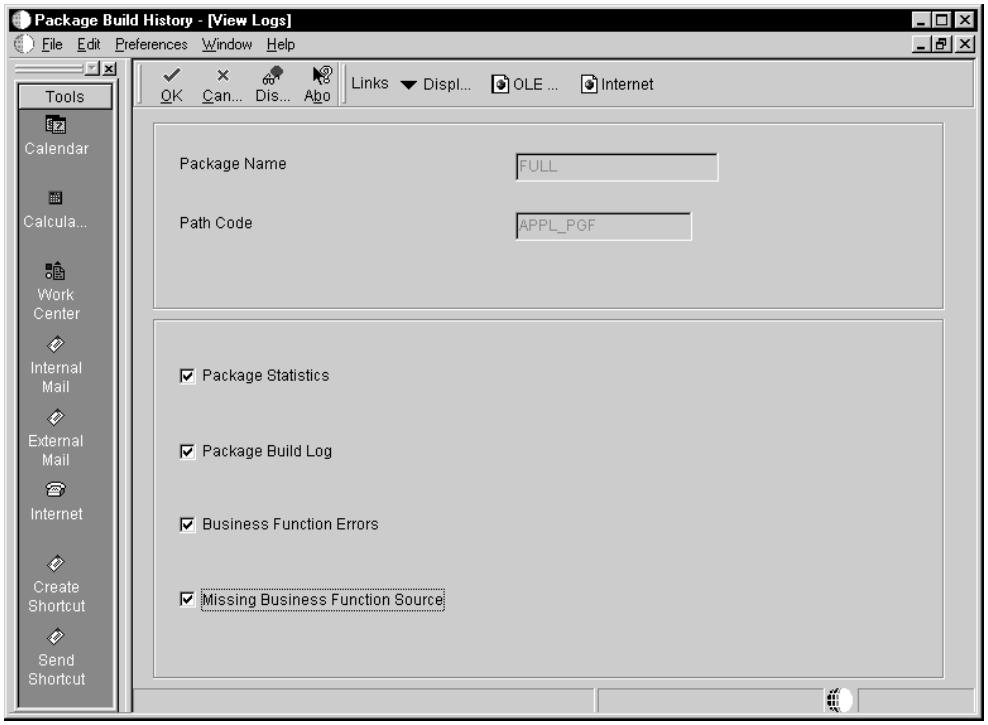
- Package statistics log
- Package build log
- Business function errors log
- Missing business function source errors log

Each log contains a header, which includes the package name, date, build machine, and path code.

► To view a log

From the Package and Deployment Tools menu (GH9083), choose Package Build History (P9622).

1. On Work With Package Build History, choose View Logs from the Form menu.



2. On View Logs, turn on any of the following log options and click OK:
   - Package Statistics
   - Package Build Log
   - Business Function Errors
   - Missing Business Function Source

Each log that you select appears in its own window.

3. When you are finished viewing logs, close each log window.
4. On View Logs, click Cancel to return to the Work with Package Build History form.

Where to Find the Error Logs

To review error logs without using the Package Build History program (P9622), locate the desired log in the correct directory. Error logs are stored on the deployment server in directories that are subordinate to the directory for the package itself. The package build log is stored in the package directory. The package statistics log, business function source errors log, and missing business function source errors log are stored in the work directory for the package.

You can view the error logs by accessing the appropriate directory and opening the log with Microsoft Notepad or a similar application that allows you to display text files.

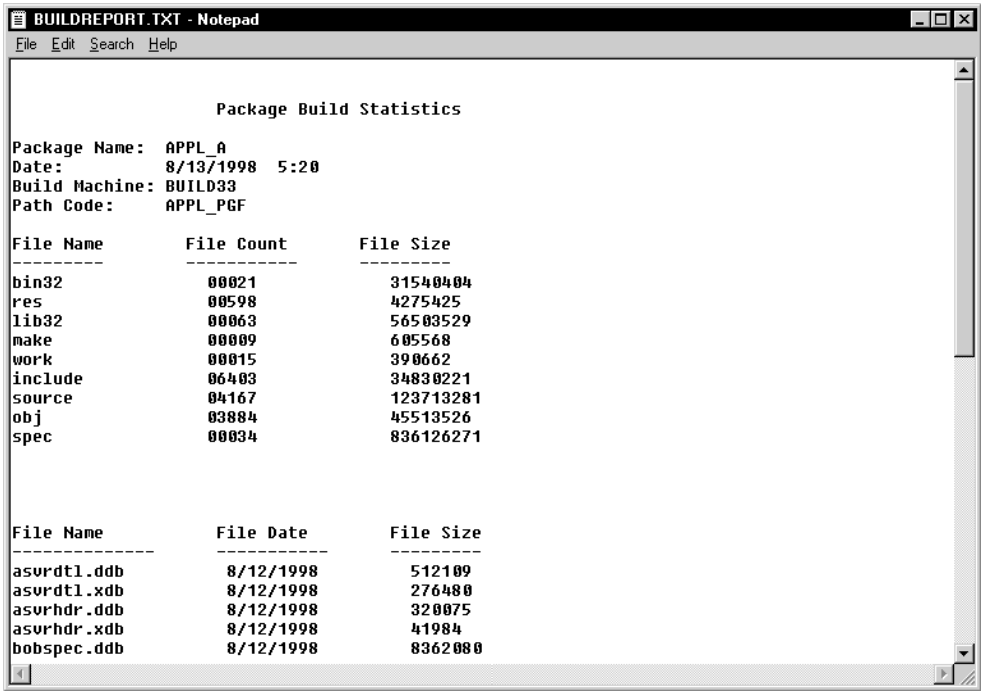
In the following examples, PD9FA is used as the package name. To determine your actual directory, substitute your package name for PD9FA.

- Package statistics: \PD9FA\work\buildreport.log
- Package build log: \PD9FA\builderror.log
- Business function errors log: \PD9FA\work\buildlog.txt
- Missing business function source log: \PD9FA\work\NoSource.txt

The Package Statistics Log

The package statistics log summarizes the outcome of the package build, showing statistics for the directories in the package, including the size and file count of each directory. This log displays a complete build that you can use to review your build directories. The report shows a breakdown of the files in the spec directory and the size of each spec file, as well as the total count and size. You can use this log to verify that the package built successfully.

The package statistics log looks similar to the following:

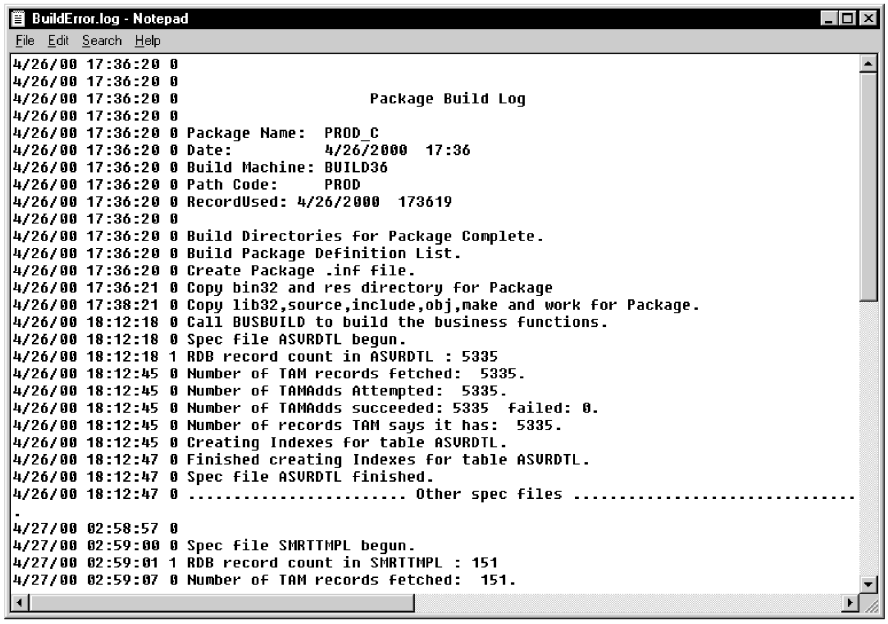


The Package Build Log

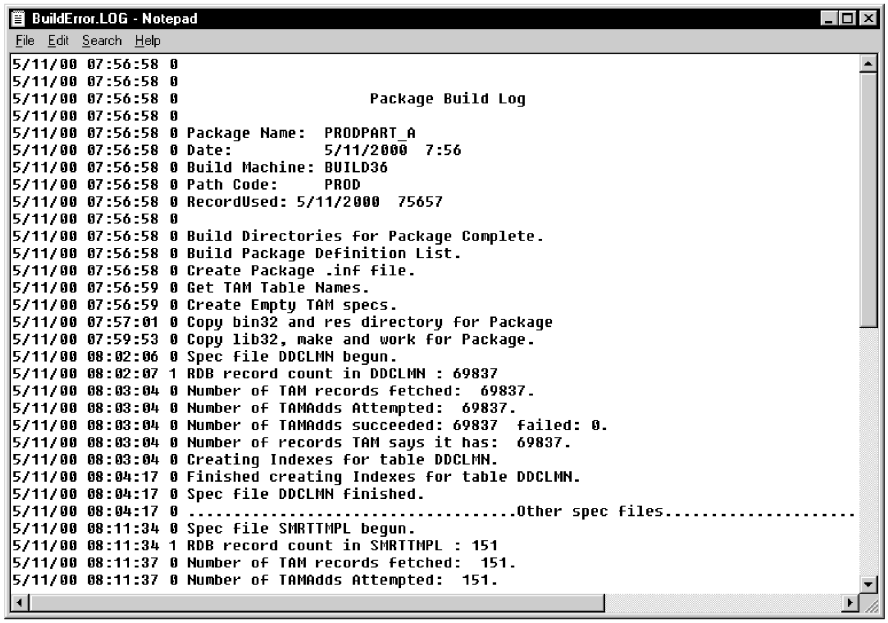
The package build log appears in the package name directory. This log lists the steps completed in building the package, as well as any errors that occurred during the package build process. It also describes the steps involved in building the package. The final page of the log tells you whether the package was successfully built.

The package build logs for each of the different package build processes (full, partial, and update) look similar to the following:

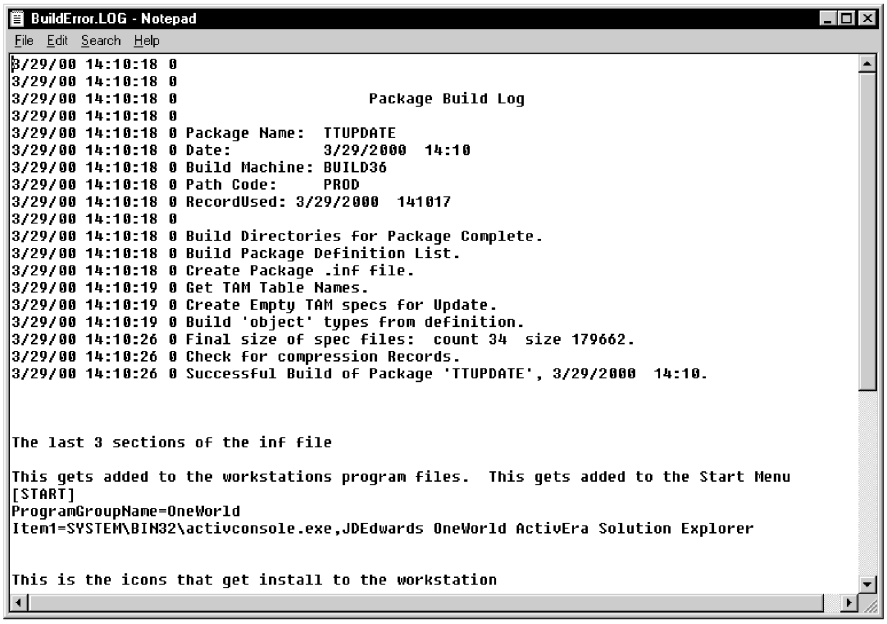
Full Package



Partial Package



Update Package



Error and warning messages are preceded with the words ERROR and WARNING, respectively. All other messages are for information only, and they describe the steps in the build process. If the build process fails, you can use this log as a troubleshooting tool to determine the last step that was completed before the failure.

For example, following are some steps from the log and some troubleshooting suggestions that you can follow if the build fails during or after the step:

- Build directories for package complete

  This step creates the directories for the package. If the build could not create the directories, verify that the server is working. Also, verify that you have the authority to create directories that are subordinate to the package directory.

- Build package definition list

  This step builds a list of all items defined in the package. If the build fails during this step, verify that the object is defined in the Software Package Detail table (F9631).

- Create empty TAM specs

  This step creates the local workstation specifications (TAM tables) in the package directory that is subordinate to the specs directory. If the build fails during this step, verify that the server is working. Also, verify that you have permission to create directories that are subordinate to the spec directory.

- Copy lib32, source, include, obj, make, and work for [Full] package

  This step copies the lib32, source, include, obj, make, and work directories from the check-in location to the package directory. If the build fails during this step, verify that each of these directories exiss in the check-in location and is subordinate to the package directory.

- Copy lib32, make, and work for partial package

  This step copies the lib32, make, and work directories from the check-in location to the package directory. If the build fails during this step, verify that each of these directories exists in the check-in location and is subordinate to the package directory.

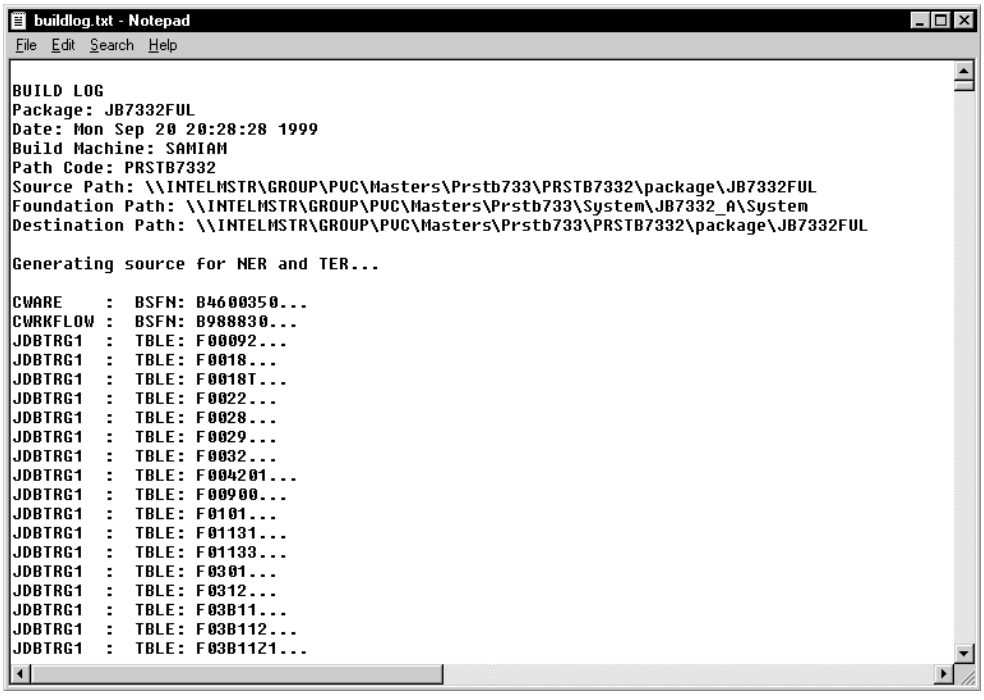
- Spec file DDCLMN begun
- Spec file DDCLMN finished

  The preceding two messages are related; they indicate that the system created the TAM spec file. If the *begun* message appears, but not the *finished* message, the system did not build the spec file. After the package is finished, you need to build only the one TAM spec file that failed. The log probably contains an entry that indicates that the system encountered an error while building that TAM spec file.

The Business Functions Errors Log

The business functions errors log allows you to view any errors that occur while business functions are being built. The final page of the log describes whether the business functions were successfully built or were built with errors. Business functions that appear on this report might be business functions that are still in development and have not yet been checked in. Business functions that have never been checked in do not have source, and therefore, are listed in the missing business function source errors log.

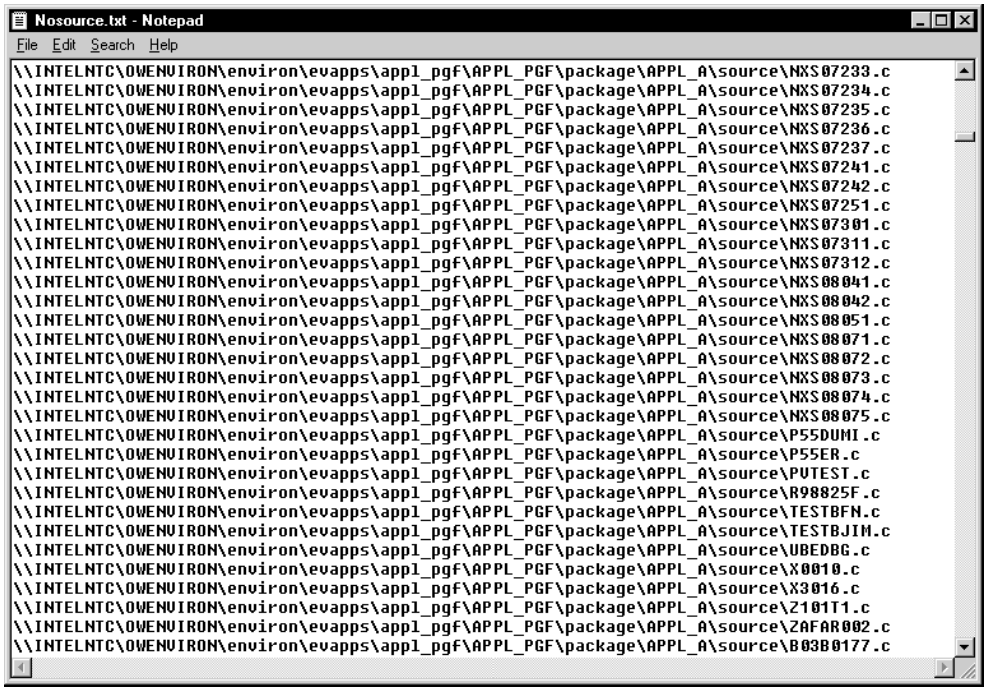
The business functions errors log looks similar to the following:



The Missing Business Function Source Errors Log

The missing business function source errors log describes any business functions in the package that are defined in the Object Librarian and have a record, but could not be built because no source existed.

The missing business function source errors log looks similar to the following:



Server Logs

All compile logs for the enterprise server are located on the server itself in the source directory of the DLL in which the object belongs. For example, suppose that you want to see the log for the Sales Order Entry Master Business Function (B4200310) in the package PACKAGE1 on an HP 9000 for which the BuildArea is /u02/PeopleSoft/packages. The system creates a file called /u02/PeopleSoft/packages/PACKAGE1/source/CDIST/b4200310.log because B4200310 is in the CDIST.DLL.

If the system could not link the CDIST.DLL (shared library) on the HP 9000, it would create a file called /u02/PeopleSoft/packages/PACKAGE1/obj/CDIST/CDIST.log.

On the AS/400, logs for business functions that failed to compile are members in a file called FAILED in the package library. Using the previous example, you would review member B4200310 of the FAILED file in library PACKAGE1.

Resubmitting a Package Build

You can resubmit a package build in the event that all objects or business functions did not build successfully.

► To resubmit a package build

From the Package and Deployment Tools menu (GH9083), choose Package Build History (P9622).

1. On Work with Package Build History, use one of the following options to find and choose the package that you want to resubmit:
   - Select a specific server to resubmit only the builds for that server
   - Select the CLIENT heading to resubmit only the workstation builds
2. Choose Resubmit Build from the Row menu.

   If you generated NERs when you initially submitted the build, ERP 9.0 displays a window that asks whether you want to regenerate the NERs.
3. Click OK to regenerate NERs, or click Cancel to skip this process.

   Note
   If you do not want to regenerate NERs, you can prevent this window from appearing by entering 2 in the Generate NER processing option for the Package Build History program.
4. Choose one of the following destinations for the build report, and click OK:
   - On Screen
   - To Printer

   The form automatically closes, and ERP 9.0 begins to build the package. Build time varies, depending on the number and size of the items in your package. When the build is finished, the report either appears on the screen or prints, depending on the destination you specified.
5. Review the report to verify that the system successfully generated all components in your package. If the report indicates any errors, review the error logs for more detail.

See Also

- *Deployment* in the *Package Management Guide* for information about scheduling the package for deployment

Changing the Build Status

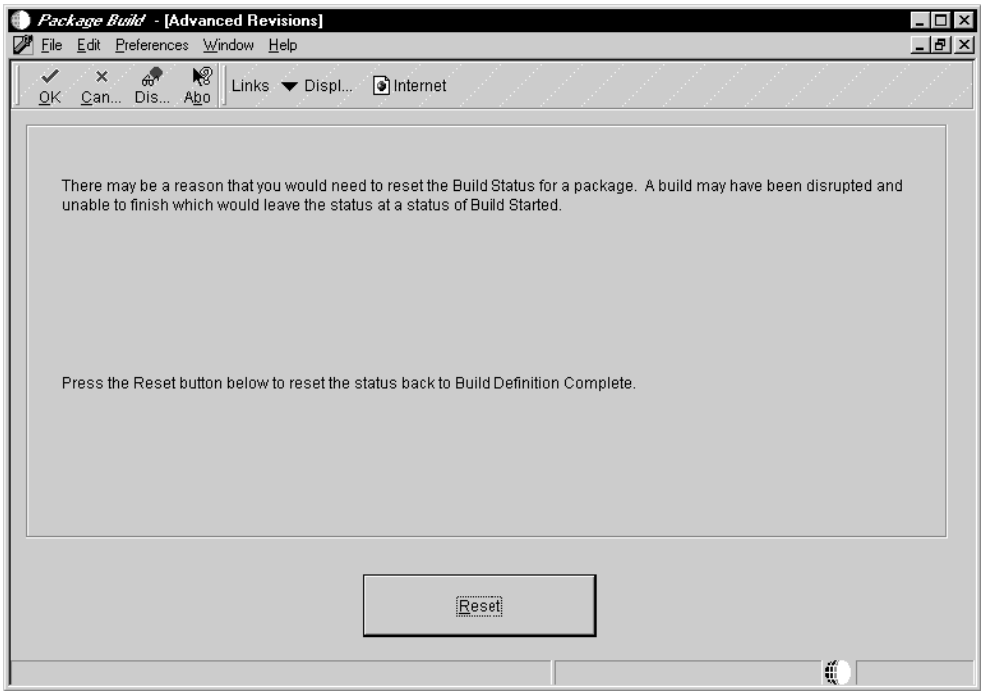
In some cases, you might need to attempt to rebuild the package rather than resume the build from the point at which the build failed. Before you can do so, you must first change the status of the package build from Build Started to Build Definition Complete.

When you reset the status of the package build, you can reset the status for the server only or for all servers and client workstations for which you want to build the package.

► To change the build status

From the Package and Deployment Tools menu (GH9083), choose Package Build (P9621).

1. On Work with Package Build Definition, find the package for which you want to reset the status. Underneath the package name, choose the server or servers and client workstation for which you want to build the package.
2. From the Row menu, choose Advanced.



3. On Advanced Revisions, click Reset to change the status of the package build from Build Started to Build Definition Complete.
4. Click OK to return to the Work with Package Build Definition form.
5. If desired, choose the package name and choose Submit Build from the Row menu.
6. The program asks whether you want to delete the current build or to continue without deleting it.

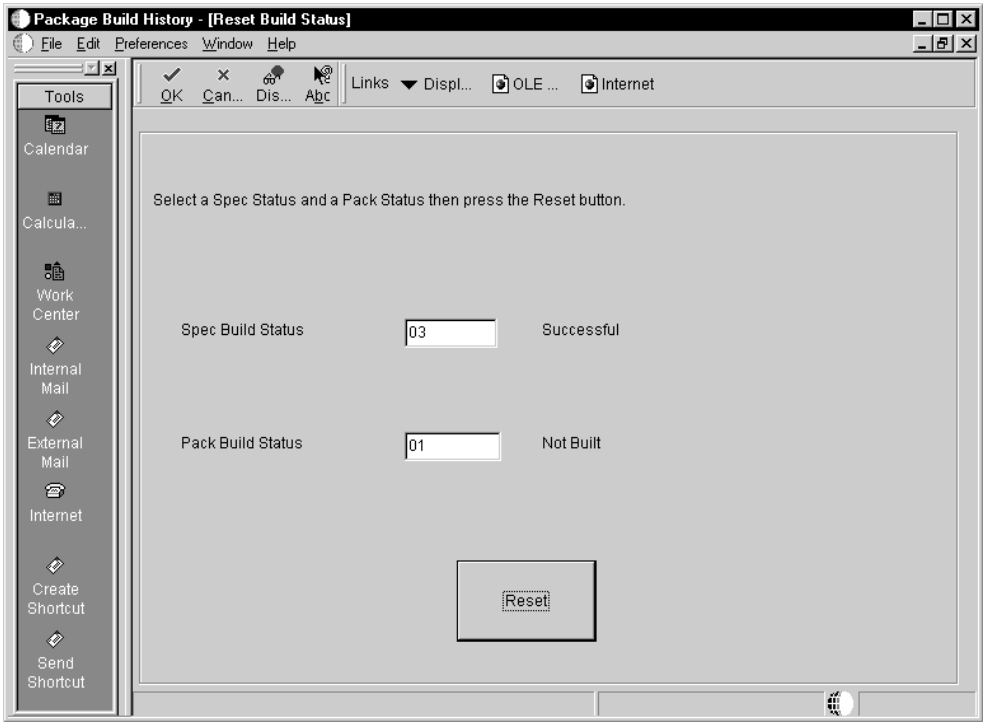
Changing the Spec Build and Pack Build Statuses

You can also reset the spec status and pack status for a package to the statuses that you specify. This option is available through the Package Build History (P9622) program.

► To change the spec and pack statuses

From the Package and Deployment Tools menu (GH9083), choose Package Build History (P9622).

1. On Work with Package Build History, find the package for which you want to reset the statuses, expand the package, and choose an individual item.
2. Choose Reset Status from the Row menu.



3. On Reset Build Status, enter the desired statuses in the Spec Build Status and Pack Build Status fields.

   Both of these fields have a visual assist feature to help you determine the available statuses.

4. Click Reset.
5. Click OK to return to the Work with Package Build History form.

Understanding Package INF Files

The Package INF file is essentially the interface between the package build and the Workstation Installation program. The INF file defines the components included in the package, the source and destination paths for those components, and the attributes needed to install the package.

The INF file is created during the package build process and is stored in its own package_inf directory, based on the release master directory. Workstation Installation reads the INF file for the package it is installing to determine which components are loaded to a workstation, as well as their locations.

Sample INF File

The following is a typical INF file for package PD9FA, which is full package A for the PD9 path code.

[Attributes]

This section contains information about the current ERP 9.0 release, global tables, specification and help files, and the JDE.INI file.

Item	Purpose
PackageName=STEST	Indicates the name of the package.
PathCode=	Indicates the path code for which the package is being built.
Built=	Indicates the package status. A status of 50 or 70 means that the package is ready for installation.
PackageType=Full	Indicates the package type: full, partial, or update.
Release=B9	Indicates the current ERP 9.0 release, which determines which setup.inf file to use in building the jde.ini for the workstation. The release is also used to determine paths for system and helps.
SystemBuildType	Indicates the type of build: DEBUG or RELEASE. This option is retrieved from owver.dll
MFCVersion	Indicates the version of the MFC compiler.
SpecFilesAvailable=Y	Indicates that specification files are available to merge or copy. This option is always set to Y for full or partial packages. For update packages, this option is set to Y only if objects are included in the package.
DelBlblTbl=Y	Indicates whether to delete global tables when installing. This option is set to Y for full or partial packages. For update packages, this option is set to Y only if the objects include a table object.
ReplaceIni=Y	Indicates whether to delete the existing jde.ini file when installing, and then create a new one. This option is set to Y for full and partial packages. For update packages, the user must specify during package build definition whether to replace the jde.ini file.
AppBuildDate=Mon Jul 20 11:22:22 2005	Indicates the date and time when the package was built. Full and partial packages always have this date. This option is blank when no objects are included in the package.
FoundationBuildDate=Wed Jun 03 15:08:34 2005	Indicates the date and time when the foundation was built. For ERP 9.0 and later releases, this date is retrieved from owver.dll. Full and partial packages always have this date. This option is blank when no foundation location is specified in the package.

Item	Purpose
DataBuildDate=Wed Jun 03 15:08:34 2005	Indicates the date and time when the database file (jdeb7.mdb) was built. Full and partial packages always have this date. This option is blank when no database location is specified in the package.
HelpBuildDate=Wed Jun 03 15:08:34 2005	Indicates the date and time when the help (help.jz, the compressed help file,) was built. Full and partial packages always have this date. This option is blank when no help location is specified in the package.
DeploymentServerName	Indicates the computer name of the deployment server.
Location	Indicates the location of the computer (the deployment server).
DeploymentStatus=	Indicates the package deployment status.
PackageDescription=	Indicates the package description.
Icon Description	Describes the desktop icon.
Default Environment	Indicates the default environment.
Spec	Indicates the name of the spec file that is specified in the File set.
ServerHelpPath	Indicates the path to the help files.

[SrcDirs]

The Workstation Installation program uses these settings to determine the source path from which information is copied. A package build compresses these directories. Workstation Installation copies the compressed directories to workstations.

Item	Purpose
SPathcode=\\MachineName\ ONEWORLD\B9\PACKAGE\ PD9FA	Indicates the location of the package that the package build compresses for Workstation Installation. The default value for this path is the path code directory over which you built the package. You can change this setting if you want to use a different package.
SSYS=\\MachineName\ ONEWORLD\B9\SYSTEM	Indicates the location of the system directory that the package build compresses for Workstation Installation. The default value for this path is the system directory that is associated with the path code over which you built the package. Normally, this directory is subordinate to the release share name (B9). This item appears only when included in the package.

Item	Purpose
SPathcodeDATA=\\MachineName\ ONEWORLD\B9\ PD9\PACKAGE\DATA	Indicates the location of the J.D. Edwards Supported Local Database that the package build compresses for Workstation Installation. The default value for this path is the data directory that is subordinate to the path code over which you built the package. This item appears only when included in the package.
SHELP=\\MachineName\ ONEWORLD\B9\HELPS	Indicates the location of the help files that the package build compresses for Workstation Installation. Users might not want to allocate disk space on their workstations to help files. When you build a package, the default value for this setting is the help directory under the release share path (B9). You can change this setting if you want to change the location in which helps reside. This item appears only when included in the package.

[DestDirs]

The Workstation Installation program uses these settings to determine the destination paths on the workstation. The process replaces "%INSTALL" with the user's computer configuration that is set up in the User Display Preferences table (F00921) and the User Defined Codes Language Status table (F00051).

Item	Purpose
DPathcode=%INSTALL\path code	Indicates the destination directory for the package.
DSYS=%INSTALL\system	Indicates the destination directory for system files. This item appears only when included in the package.
DPathcodeDATA=%INSTALL\path code\data	Indicates the destination directory for the database. This item appears only when included in the package.
DHELP=%INSTALL\helps	Indicates the destination directory for help files. This item appears only when included in the package.

[Filesets]

These settings list the various source and destination directories that are subordinate to the path code for the package. Y=compressed, and N=not compressed. The source and destination directory names are preceded by an S and D, respectively.

Item	Purpose
Pathcode1=Y, $Spathcode\bin32, $Dpathcode\bin32	Indicates the bin32 directory that is subordinate to the path code.
Pathcode2=Y, $Spathcode\res, $Dpathcode\res	Indicates the res directory that is subordinate to the path code.

Item	Purpose
Pathcode3=Y, $Spathcode\spec, $Dpathcode\spec	Indicates the spec directory that is subordinate to the path code.
Pathcode4=Y, $Spathcode\include, $Dpathcode\include	Indicates the include directory that is subordinate to the path code.
Pathcode5=Y, $Spathcode\lib, $Dpathcode\lib	Indicates the lib directory that is subordinate to the path code.
Pathcode6=Y, $Spathcode\obj, $Dpathcode\obj	Indicates the obj directory that is subordinate to the path code.
Pathcode7=Y, $Spathcode\source, $Dpathcode\source	Indicates the source directory that is subordinate to the path code.
Pathcode8=Y, $Spathcode\work, $Dpathcode\work	Indicates the work directory that is subordinate to the path code.
Pathcode9=Y, $Spathcode\make, $Dpathcode\make	Indicates the make directory that is subordinate to the path code.
PathcodeDATA=Y, $SpathcodeDATA\data.CAB, $DpathcodeDATA	Indicates the compressed database that the package build generates.
HELP=Y, $SHELP\Helps.CAB, $DHELP	Indicates the compressed help file.

[Components]

These settings indicate the location of the foundation, production, and development objects; the database; and the help files.

Item	Purpose
ProdObj=APPL_PKG1, APPL_PKG2, APPL_PKG3	Indicates the location of production objects.
DevObj=APPL_PKG4, APPL_PKG5, APPL_PKG6, APPL_PKG7, APPL_PKG8, APPL_PKG9	Indicates the location of development objects.
Foundation=SYS	Indicates the foundation location.
Data=<pathcode> DATA	Indicates the database location.
Help=HELP	Indicates the help file location.

[Typical]

This section describes the setting for a typical development user.

Item	Purpose
Name=Development	Indicates that the package is for a development user.
Description=Install the development objects	Indicates the package description.
Components=ProdObj, DevObj, Foundation, Data	Indicates that the package should contain both production and development objects, as well as the database and foundation. See the *Components* settings to determine the location of these components.

[Compact]

This section describes settings for a typical production user.

Item	Purpose
Name=Production	Indicates that the package is for a production user.
Description=Install the production objects	Indicates the package description.
Components=ProdObj, Foundation, Data	Indicates that the package should contain only production objects, as well as the database and foundation. See the *Components* settings to determine the location of these components.

[MSDE]

This section contains information about the local data source. The name of this section corresponds to the local data source name in the ODBC data sources section.

Item	Purpose
Driver=jdboledb.dll	Indicates the name of the driver.
DefaultDir=C:\ACCESS	Indicates the default directory.
UID=sa	Indicates a required attribute setting for the J.D. Edwards Supported Local Database.
Driver32=jdboledb.dll	Indicates the ODBC driver for the J.D. Edwards Supported Local Database.
DBQ=$DAPPL_PGFDATA\JDELocal.mdf	Indicates the path code for the J.D. Edwards Supported Local Database.
MaxBufferSize=2048	Indicates the maximum size of the buffer

Threads=1028	Specifies the number of threads

[Features]

This section describes information for any features that are included in the package. A feature is a set of files or configuration options that is included in a package for deployment to a workstation or server. For more information about features, see *Incorporating Features Into Packages*.

Item	Purpose
FeatureName=SFEAT	Indicates the name of the feature in the package.
Feature.inf location=	Indicates the location of the feature.inf file for the feature.

Understanding Feature INF Files

When a package contains features, a section called [Features] in the Package INF file includes both the feature name and a pointer to the Feature INF file that is created for each feature in the package. These Feature INF files provide specifications that tell the installation program the actions to perform during the installation.

The Feature INF file can include the following sections:

- [Header]
- [Registry]
- [INI]
- [FileSets]
- [Shortcut]
- [ThirdPartyApps]
- [ODBCDataSources]

Sample Feature INF File

The following is a typical Feature INF file for which the sections contain specifications for each feature component.

[Header]

The header section contains general information about the feature and specifies the installation options for the feature.

Item	Purpose
Feature=	**Feature**. Name of the feature
FeatureType=	**FeatureType**. Type of feature.
Description=	

Required= InitialChoice=	**Description**. Text description of the feature. **Required**. A setting that indicates whether installation of the feature is required. **InitialChoice**. A setting that specifies the default selections for features that the user has the option to install.

The Required and InitialChoice entries are set via the three Feature Installation option settings (Required, Selected, Deselected) on the Feature Information form. When you select one of these three options, ERP 9.0 automatically writes the following values into the Required and InitialChoice entries in the feature inf file.

Feature Installation Option	Required	InitialChoice
Required	Y	Both
Selected	N	Both
Deselected	N	Custom

[Registry]

This section contains information about how the feature affects the Windows registry.

Item	Purpose
Registry[no.]=Root[value],Key,[prefix]Name,[prefix]Value	Indicates the following specifications for the Windows registry settings for the features: **Root.** Describes the root in the registry with the following values: • 0 means root • 1 means current user • 2 means local machine location • 3 means users **Key.** Indicates the key for the registry value. **Name.** The registry value name. Name prefixes are: • + means that the name is created (if it does not already exist) when the feature is installed • - means that the name is deleted with all subkeys when the feature is uninstalled • * means that the name is created (if it does not already exist) when the feature is installed, and it is removed with all subkeys when the feature is uninstalled **Value.** The name of the registry value. Value prefixes are: • #x means that the value is stored as a hexadecimal value. • #% means that the value is stored as an

expandable string.

- # means that the value is stored as an integer.
- #$ means that the value is stored as a string.

[INI]

This section contains information about how the feature affects the JDE.INI file.

Item	Purpose
Ini[no.]=FileName,Directory,Section,Key,Value,Action[value]	Indicates the following information that pertains to the destination INI file: **FileName**. The name of the destination INI file. **Directory**. The location of the destination INI file. **Section**. The name of the section in the destination file. **Key**. The name of the key within the section of the destination file. **Value**. The value to be written to the key of the destination file. **Action**. The action to take regarding the INI entry: • 0 means create the INI entry. • 1 means create the INI entry only if it does not already exist. • 3 means create the INI entry or append to the existing entry.

[FileSets]

This section contains information about additional files that must be installed for the feature to function correctly.

Item	Purpose
Fileset[no.]=Compression, SourceDirectory, FileName, TargetDirectory	Indicates the following specifications for the fileset: • Compression. An option that indicates whether the fileset is compressed. • Source Directory. The source location of the fileset. • FileName. The name of the CAB file for the fileset. • Target Directory. The target location into which the fileset will be placed.

[Shortcut]

This section contains information about shortcuts that appear on the Windows desktop as part of the feature installation.

Item	Purpose
Shortcut[no.]=Directory,Name,Target,Arguments,Description,HotKey,Icon,IconIndex,ShowCmd[value],WkDir	Indicates the following specifications for the shortcut: **Directory**. The directory where the shortcut is created. **Name**. The name of the link file for the shortcut. **Target**. The name of the executable file for the shortcut. **Arguments**. Any command line arguments for the shortcut. **Description**. A description of the shortcut. **HotKey**. A hot key that launches the shortcut. **Icon**. The shortcut icon and location. **IconIndex**. An index of the icon if the icon is inside an image list. **ShowCmd**. A command for the application window, with the following value options: • 0 means show the window normal-sized. • 3 means show the window maximized. • 7 means show the window minimized; not active. **WkDir**. The working directory for the shortcut.

[ThirdPartyApps]

This section contains information about third-party products that are installed with the feature.

Item	Purpose
ThirdPartyApp[no.]=Source Directory, Description, Synchronous/Asynchronous, FileName	Indicates the following specifications for the third party applications: **Source Directory**. Source location of the executable for running the third party application. **Description**. Description of the third party application. **Synchronous/Asynchronous**. An option that indicates whether the third party application can be installed in parallel (synchronous) or must be installed serially (asynchronous). **FileName**. The name of the file that launches the third party application.

[ODBCDataSources]

This section contains information about ODBC data sources that are installed with the feature.

ODBC data sources have 2 sections in the feature INF. One section contains header information and the other contains the detail information. The feature.inf contains one header section listing all data source components that are included in the feature. For each data source listed in the header, a corresponding detail section exists. Only the header section is described in the following table. For information about the detail section, see the documentation for the selected ODBC Driver.

Item	Purpose
DataSourceName=DataSourceDriver	Indicates the following specifications for the ODBC data sources: **DataSource Name**. The name of the ODBC data source. **DataSource Driver**. The driver used for the data source.

Deployment

After you have assembled, defined, and built a package, you can choose from several ways to deploy the package to workstations and servers.

Understanding Package Deployment

After you build a package, you can choose from several methods of deploying the package to workstations and servers throughout your enterprise. For workstations, the method that you select depends on whether ERP 9.0 is already installed on the workstation.

Deployment to Workstations without ERP 9.0

If ERP 9.0 is not currently installed on a workstation, you can deploy the package through the Workstation Installation program. You use Workstation Installation to deploy only full and partial packages; you cannot use Workstation Installation to deploy an update package to a workstation on which ERP 9.0 is not installed.

Workstation Installation retrieves from the central object data source the items specified in the package. A package is like a bill of materials with instructions that describe from where the system retrieves all of the necessary components that the Workstation Installation program deploys to the local workstation. This program can be run interactively (initiated by a person at a workstation) or in silent mode and scheduled through the push installation feature.

If you use the push installation feature, you can use Package Deployment to deploy the package. Push installation enables the ERP 9.0 administrator to initiate the installation of a package from the deployment server to workstations without any user interaction. To use this feature, the push installation *listener* application must be installed on the workstation, and the machine must be defined through the Machine Identification program (P9654A).

See Also

- See the *ERP 9.0 Installation Guide f*or more information about the Workstation Installation application

Deployment to Workstations with ERP 9.0 Already Installed

To reload a new package on workstations on which ERP 9.0 is already installed, use one of two methods:

- Workstation Installation (for full and partial packages)
- Package Deployment (an ERP program (P9631) for full, partial, and update packages).

After you assemble and build a package, use Package Deployment to schedule the package for deployment to individual workstations or to selected groups. On the specified deployment date, when the users who are scheduled to receive the package sign on to ERP 9.0, they are given the opportunity to load the package.

Unless you are using the Push Installation feature, Package Deployment requires that ERP 9.0 be already loaded on the workstation. You can schedule either a new full or partial

package to replace the existing package, or an update package to be merged with the existing package on the workstation.

Both deployment methods have advantages. Workstation Installation is a good method to use when you want to install a package immediately or soon after it is built, without having to schedule the package. Alternatively, Package Deployment is useful if you need to control when the package becomes available, if you want to make the package installation mandatory, or if you want to deploy the package to servers as well as to workstations.

Deployment to Servers

Servers receive the same package that you build for the workstation, but in a different format. When you assemble the package and create the package build definition, you can specify the servers to which you want to deploy the package. To deploy the package, you use the Package Deployment application (P9631), which uses the same scheduling mechanism to deploy packages to workstations. In fact, you can easily schedule deployment to both client workstations and servers on the same form. You cannot use the Push Installation feature to deploy to servers.

Deployment to Tiered Deployment Locations

Multitier deployment allows you to install software on workstations from more than one deployment location and more than one deployment machine. Use this deployment method if your site has more than 50 workstations performing ERP 9.0 software installations per day, or when workstation installations over your WAN are too slow.

Deploying to Workstations from CD

If your system has a CD writer, you can define the CD writer as a deployment location. Essentially, you define the CD writer as a pseudo deployment server from which you can copy a ERP 9.0 package onto a blank CD. You can then use this CD to install ERP 9.0 on workstations by using the Workstation Installation program that is included on the CD.

Defining Machines, Locations, and Deployment Groups

Before you deploy packages, you must identify the workstations, servers, groups, or locations that will receive your package. Identifying these ensures that, when you are ready to schedule packages using the Package Deployment Director, the machines, groups, or locations that you want to receive your package will be available as package recipients.

A deployment group is a group of workstations that are classified by a criterion such as job function, team, or any other grouping that you specify. For example, you might have a software development group, a testing group, a production group, and so on. The Machine Group Identification program (P9652A) enables you to define or revise groups that include several workstations.

A location is a group of workstations and servers that corresponds to a physical location. For example, you might have locations for Corporate and Branch, or for Building 5 and Building 7. Locations are also useful if you use multitier deployment or deploy across a wide area network. In this case, you might define a location for each of your geographic locations. The Machine Identification program (P9654A) enables you to define or revise machines and locations in your enterprise.

Both of these applications simplify the deployment process when you need to deploy a package to several users. Rather than requiring you to schedule deployment to each workstation or server, you can schedule deployment according to location or group.

When you enter a machine definition, you are really defining its usage in the ERP 9.0 configuration. For example, you can use a deployment server as a data server. When you enter machine definitions, consider the following recommendations from J.D. Edwards:

- A Java application server (JAS) can be defined only as a Java application server, not as a data server, enterprise server, and so on.
- A deployment server should not be used as a workstation.
- A deployment server can be used as a data server.
- A deployment server should not be used as an enterprise server for tuning and performance reasons.

Defining Machines

Before you use the Package Deployment Director to deploy a package to individual client workstations, verify that each machine that will receive the package has a record in the Machine Master table (F9650).

Choose one of the following ways to populate this F9650 table:

- Manually. For a machine that no user has ever used to sign on to ERP 9.0, use the Machine Identification application (P9654A) to manually enter a record in the F9650 table.
- Automatically. ERP 9.0 automatically creates a record in the F9650 table when a user on a new machine signs on to ERP 9.0 for the first time. (The system also automatically updates existing records in the F9650 table each time a user signs on to the workstation.)

The simplest way to populate the F9650 table is to have all users on new machines sign on to ERP 9.0. In cases where you need to deploy a package before the users can log on, you must manually enter machine information. The Machine Identification program enables you to perform this task.

In addition to defining workstations, you can also use the Machine Identification program to enter or revise definitions for the following machines:

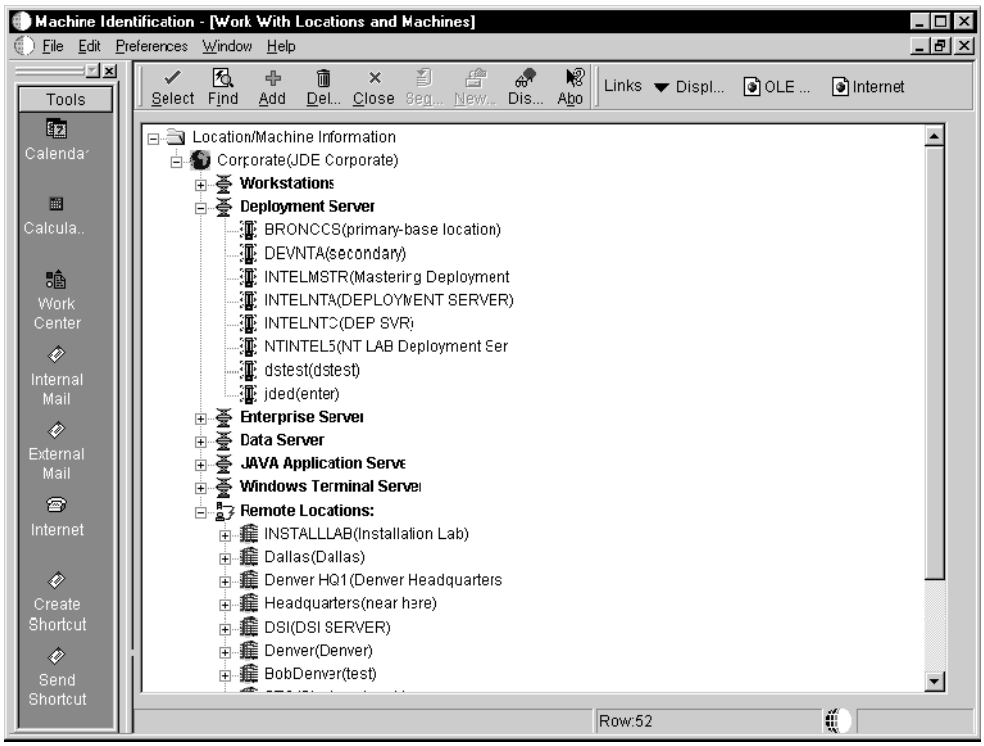
- Deployment servers
- Enterprise servers
- Data servers
- Java application servers
- Windows terminal servers

You can define a machine to be a workstation and a data server or a data server only, but no other combinations are allowed. For example, you cannot define a machine as a workstation and a deployment server.

You can enter or revise definitions for these machines in multiple locations, including remote locations.

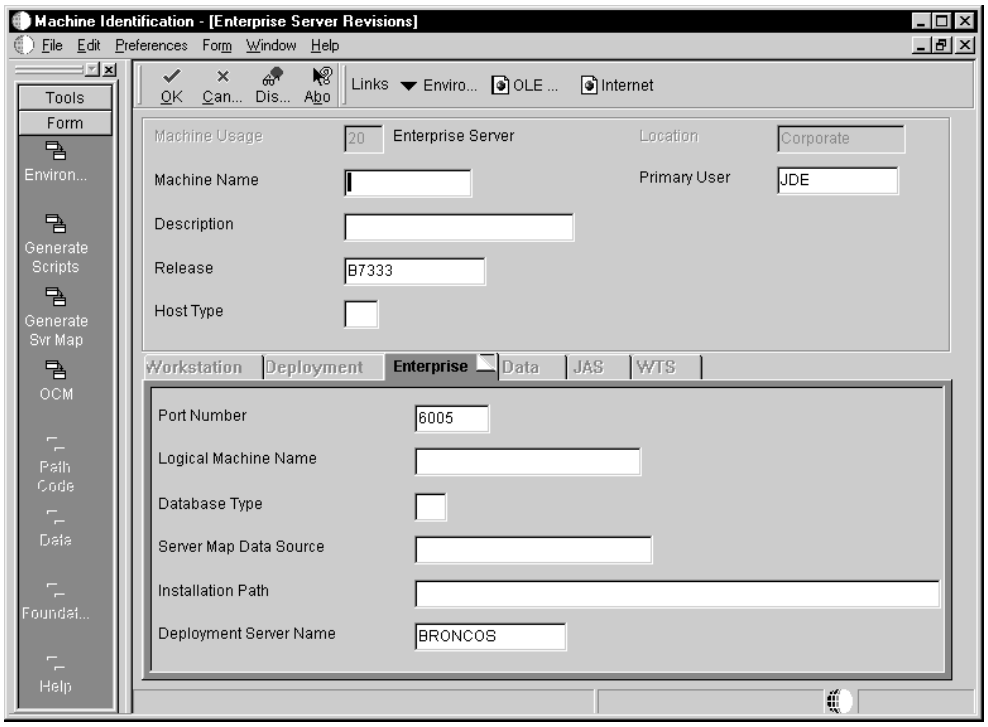
► To define a machine

From the Package and Deployment Tools menu (GH9083), choose Machine Identification.



1. On Work With Locations and Machines, click Find.
2. Underneath the appropriate location, choose the type of machine that you want to add:
   - Workstation
   - Deployment server
   - Enterprise server
   - Data server
   - Java application server
   - Windows terminal server
3. Click Add to add a new machine.

   The Machine Revisions form appears. This form displays different fields, depending on the type of machine that you are adding.



4. On Machine Revisions, complete the following fields:

   - Machine Name

     The machine name is case-sensitive. To avoid problems with inconsistent naming conventions, define the machine name using all uppercase characters.

   - Description

   - Release

   - Host Type

     Examples are AS/400, HP9000, Intel NT, and so on.

   - Primary User

     The name of the primary user of the machine.

   The tab that corresponds to the type of machine that you are defining appears automatically.

5. On the Workstation tab, complete the following field for the machine that you are defining:

   - Deployment Server Name

6. On the Deployment [Server] tab, complete the following field:

   - Primary Deployment Server

If you have set up a primary deployment server, you cannot access the Primary Deployment Server field when you define a new deployment server. You can change the value in this field only when you revise the primary deployment server definition or when you change your primary deployment server to a secondary server. In this case, you can specify a different server as your primary deployment server.

- Server Share Path

When you define a secondary deployment server, options on the Form menu enable you to select path codes, data items, foundation modules, and help items. (These options are not available for the primary deployment server.)

7. On the Enterprise [Server] tab, complete the following fields:
   - Port Number
   - Logical Machine Name
   - Database Type
   - Server Map Data Source
   - Installation Path
   - Deployment Server Name

   When you define an enterprise server, options on the Form menu enable you to generate installation scripts, generate and populate server map tables, and use the Object Configuration manager (OCM) to update the workstation server map. For an enterprise server, you can also display the machine environment and description.

   Some of the fields might already be defined and cannot be changed.
8. On the Data [Server] tab, complete the following field:
   - Data Source Type
9. On the JAS [Server] tab complete the following field:
   - Installation Path
10. On the WTS (Windows Terminal Server) tab, complete the following field:
    - Installation Path
11. From the Form menu, choose Environment.
12. On Machine Environment Revisions, click the Environment column and select one or more environments to add to the new machine.
13. Click OK.
14. On Machine Revisions, click OK again to return to the Work with Locations and Machines form.
15. On Work with Locations and Machines, click Close to exit.

The process for revising existing machine definitions is similar to adding a new definition. Instead of clicking Add on the Work with Locations and Machines form, find and choose the

machine for which you want to revise the definition, and then click Select. The same revision form appears and allows you to change the definition for the machine.

Defining Locations

In some cases, an enterprise might span several buildings, cities, or even countries. In these situations, you might deploy a package to a location rather than to individual workstations and servers. Then, a secondary deployment server at each location can deploy the package to the workstations and servers at that location.

The larger your enterprise, the more you can benefit from creating and deploying to locations. If you use multitier deployment to deploy packages to remote locations, the concept of locations is crucial.

In ERP 9.0, a *location* is essentially a user-defined group of machines, databases, and environments. In some cases, the location is an actual physical location connected by a wide area network (WAN), such as when you have remote offices that are geographically separate from your main office. For example, a location might be a floor in your office building, a separate building on your corporate campus, a branch office across town, or a facility in another city.

After you create a new location, you can add workstations and servers for that location by defining the machine names that are associated with that location.

The topmost location that appears when you launch the Machine Identification program (P9654A) is the base location. You cannot change or remove this base location, but you can create or revise locations that are subordinate to it.

When you create a location that is subordinate to another location, the original location is the parent location, and its subordinate location is the child location. For example, if you have a location called Seattle and then create a location called Redmond that is subordinate to Seattle, Seattle is the parent location and Redmond is the child location.

See Also

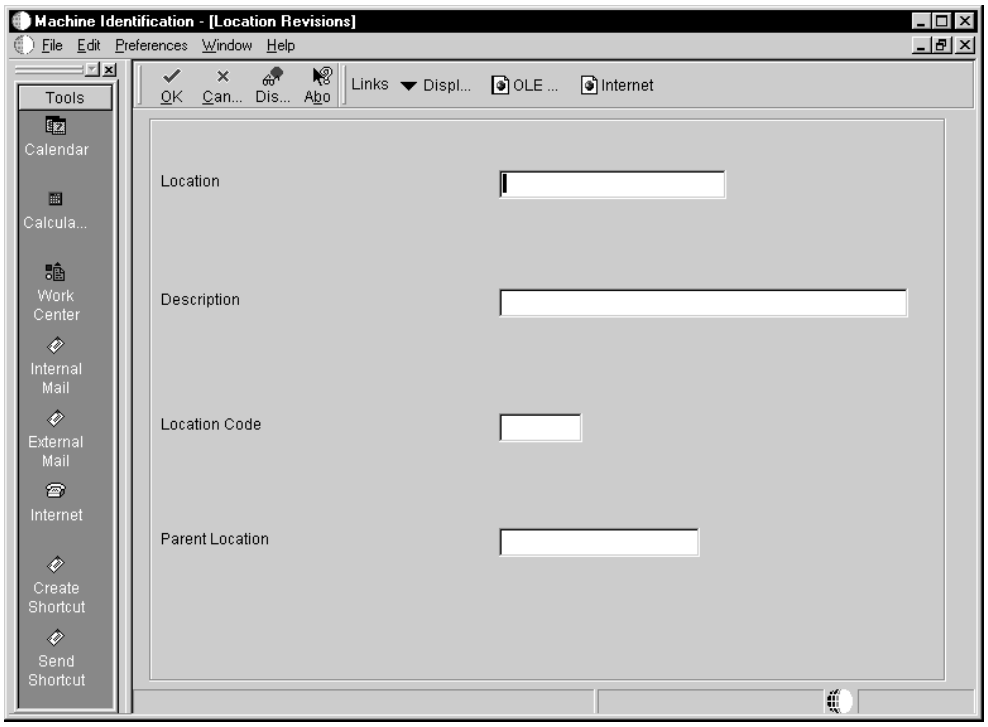
- *Multi-tier Deployment* in the *Package Management Guide* for more information about deploying packages to remote locations

▶ To define a location

From the Package and Deployment Tools menu (GH9083), choose Machine Identification.

On Work With Locations and Machines, select the current location for which you want the new location to be subordinate. For example, if you are adding a new remote location, click the Remote Locations icon.

1. Click Add to add a new location.



2. On Location Revisions, complete the following fields:
   - Location
   - Description
   - Location Code
   - Parent Location
3. After selecting the location, click Close.
4. On Location Revisions, click OK.
5. On Work with Locations and Machines, click Close.

The process for revising existing locations is similar to adding a new location. Instead of clicking Add on the Work with Locations and Machines form, find and choose the location that you want to revise, and click Select. The same revision form appears and allows you to change the location.

Defining Package Deployment Groups

You can create a deployment group based on department, team, or function. For example, you might have an administration group, a testing group, a production group, and so on.

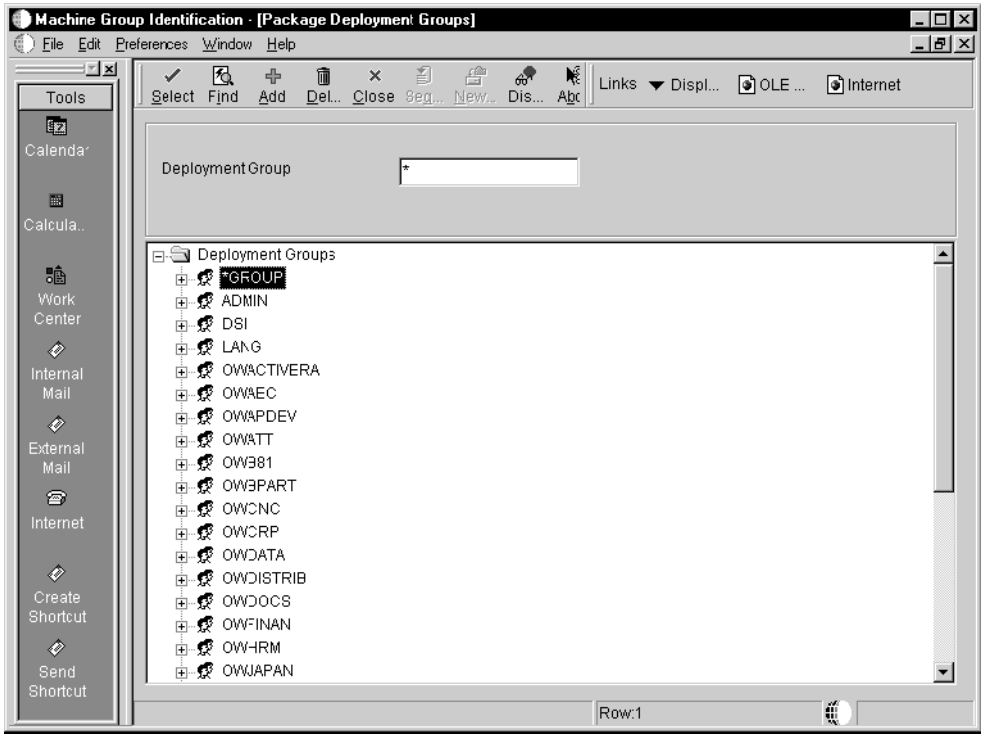
Package deployment groups are particularly useful in large enterprises in which scheduling a package for deployment to several individual workstations is very time consuming. In these environments, you can deploy packages much more quickly when you use deployment groups.

A group can contain a subgroup (a group within a group). For example, you might have a group called Quality Assurance that is a subgroup of the larger Development group.

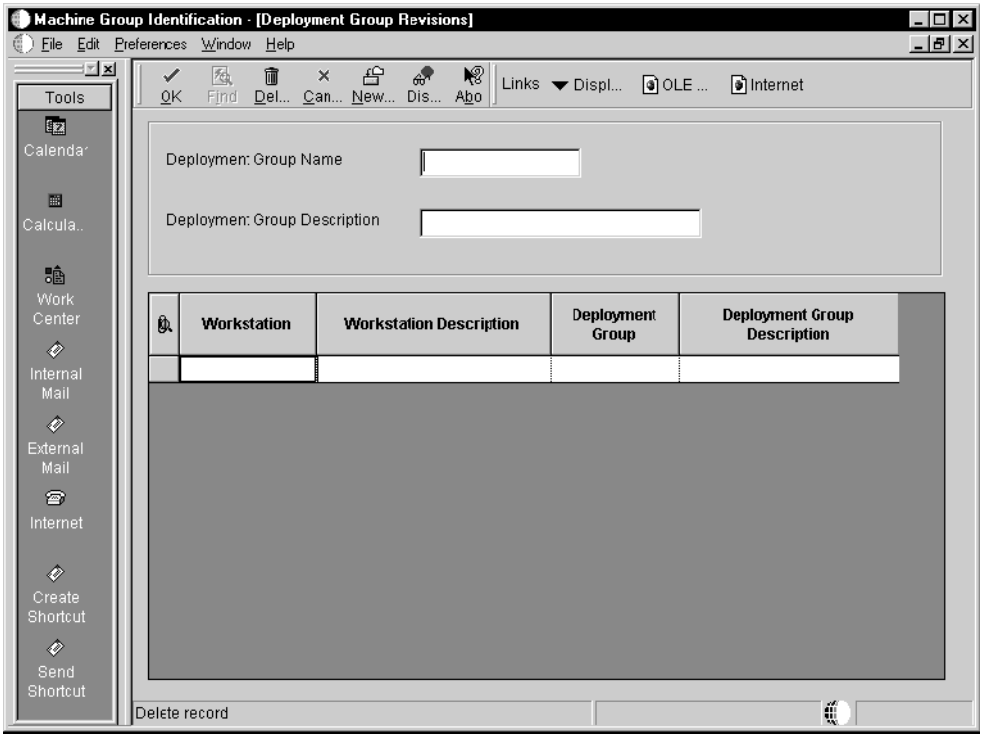
You can help the person who builds and schedules packages by creating easily identifiable names for deployment groups. For example, for a group that includes quality assurance specialists who are responsible for testing, name the group Testing, rather than Green Team.

► To define a package deployment group

From the Package and Deployment Tools menu (GH9083), choose Machine Group Identification.



1. On Package Deployment Groups, click Add to add a new deployment group.



2. Enter the deployment group name and description.
3. Enter the workstations or deployment groups that you want to include in the group.

   You can use the visual assist feature to locate the names of valid workstations and deployment groups.

4. Click OK when you are finished adding workstations or deployment groups.

   The Package Deployment Groups form appears.

5. Click Close to exit the Package Deployment Groups form.

Revising a Deployment Group

The following task describes how to revise an existing package deployment group.

► To revise a package deployment group

From the Package and Deployment Tools menu (GH9083), choose Machine Group Identification.

1. On Package Deployment Groups, choose the desired group and click Select.

   When you revise an existing group, you cannot change the group name, but you can change the description.

2. On Deployment Group Revisions, to add to the group, choose the last row (the empty one) and enter the name of the workstation or deployment group to which you want to add members.

You can do this by typing the name in the Workstation or Deployment Group field or by using the search button for those fields.

When you use the search button for the Workstation field, the Machine Select form appears. When you use the search button for the Deployment Group field, the Deployment Group Search form appears.

3. Click OK when you are finished adding or revising workstations or deployment groups.

   The Package Deployment Groups form reappears.

4. Click Close to exit from the Package Deployment Groups form.

Working with the Package Deployment Director

After you define and build a package, use the Package Deployment program (P9631) to schedule the package for deployment to individual workstations, deployment servers, or enterpriser servers. On the specified deployment date, users who are scheduled to receive the package can load the package when they sign on to ERP 9.0.

Alternatively, you can schedule the package to deployment groups or locations instead of specific machines. Deployment groups are useful in large enterprises that routinely deploy packages to many workstations and servers.

Understanding the Package Deployment Director

The Package Deployment program (P9631) simplifies and expedites the process of scheduling and deploying built packages to workstations and servers. The director displays a series of forms that allow you to specify the package that you want to deploy, the deployment destinations, and the deployment time.

After specifying the package that you want to deploy, you specify any of the following destinations:

- Client workstation
- Enterprise server
- Deployment server or Deployment groups
- Locations

You can deploy a package either to specific workstations and servers, or you can schedule the deployment based on deployment groups or location. You cannot do both; you must choose one of these methods.

You can make the package mandatory, which means that users cannot access ERP 9.0 until they have installed the package. If the package is optional, users will be given the option of installing the package every time that they sign on to ERP 9.0 until they either install or decline the package.

In addition, you can specify a *push installation*, which means that the package can be deployed from the deployment server to the workstations that you specify, without requiring any interaction from the user.

The Package Deployment Director requires that ERP 9.0 already be loaded on the workstation, unless you are using push installation. You can schedule either a new full or partial package to replace the existing package, or an update package to be merged with the existing package on the workstation.

The Package Deployment Director uses the following tables:

- Machine Master (F9650)
- Machine Detail (F9651)
- Deployment Group Header (F9652)
- Deployment Group Detail Definitions (F9653)
- Deployment Locations Definition (F9654)
- Package Deployment Scheduling (F98825)
- Package Deployment on Servers Information (F98826)
- Software Package Header (F9603)

The following table summarizes the function of each form in the Package Deployment Director:

Package Deployment Director form	View this form for a description of the Package Deployment Director.
Package Selection form	Use this form to find and select the package that you want to deploy.
Package Deployment Targets form	Use this form to specify the destination for your package. You can select individual client workstations, deployment servers, and enterprise servers, or you can deploy the package to a deployment group or location.
Package Deployment Attributes form	Use this form to enter the date and time that you want to deploy the package. Also specify whether the package is mandatory (that is, it must be installed by every package recipient) and whether you want to use Push Installation to deploy the package.
Deployment Client Workstation Selection form	Use this form to select each of the client workstations that will receive the package.
Deployment Server Selection form	Use this form to select each of the deployment servers that will receive the package.
Enterprise Server Selection form	Use this form to select each of the enterprise servers that will receive the package.
Deployment Location Selection form	Use this form to specify the deployment location that will receive the package.
Deployment Groups Selection form	Use this form to specify the deployment groups whose members will receive the package.
Build Selection form	For multitier deployment, use this form to specify the server or client package that you want to deploy to the destination deployment server.
Work with Package Deployment form	Use this form to review and revise the locations and package recipients that you entered through the Package Deployment Director.

See Also

- *Using Push Installation* in the *Package Management Guide* for more information about push installation

Using the Package Deployment Director

After you have assembled and built your package, defined all machines, and verified your deployment groups, you are ready to use the Package Deployment Director to specify package recipients and schedule the package for deployment.

Throughout the deployment process, you can choose to either proceed to the next form or return to the previous form. Also, regardless of where you are in the process, you can cancel it.

When you schedule a package for deployment to a machine rather than a deployment group or location, you can schedule to deploy the package to client workstations, deployment servers, enterprise servers, or a combination. The forms that appear vary depending on your selection. For example, if you indicate that you want to schedule a package for deployment to client workstations and a deployment server, the forms for selecting specific workstations and

deployment servers appear. If you schedule a package for deployment only to client workstations, the server selection form does not appear.

When you access the Package Deployment Director, the Work with Package Deployment form enables you to view deployed package information by either machines, deployment groups, locations, or packages.

Depending on your display selection, the tree displays different information when you expand it. The following describes the information that appears as you expand the tree level by level:

- **Machines**

  Level One: Client Workstation, Deployment Server, and Enterprise Server headings

  Level Two: Specific machines under each of the above three headings

  Level Three: Specific packages deployed to the machine, if any

- **Deployment Groups**

  Level One: Specific groups

  Level Two: Members of those groups

  Level Three: Specific packages deployed to the group member

- **Locations**

  Level One: Specific locations

  Level Two: Client Workstation, Deployment Server, Enterprise Server, and Remote Locations headings

  Level Three: Specific machines under the Client Workstation, Deployment Server, and Enterprise Server headings

  Level Three under Remote Locations only: Defined remote locations

  Level Four: Specific packages deployed to each machine, if any

  Level Four under Remote Locations only: Client Workstation, Deployment Server, and Enterprise Server headings

  Level Five under Remote Locations only: Specific machines under the Client Workstation, Deployment Server, and Enterprise Server headings

  Level Six under Remote Locations only: Specific packages deployed to each machine, if any

- **Packages**

  Level One: Package names

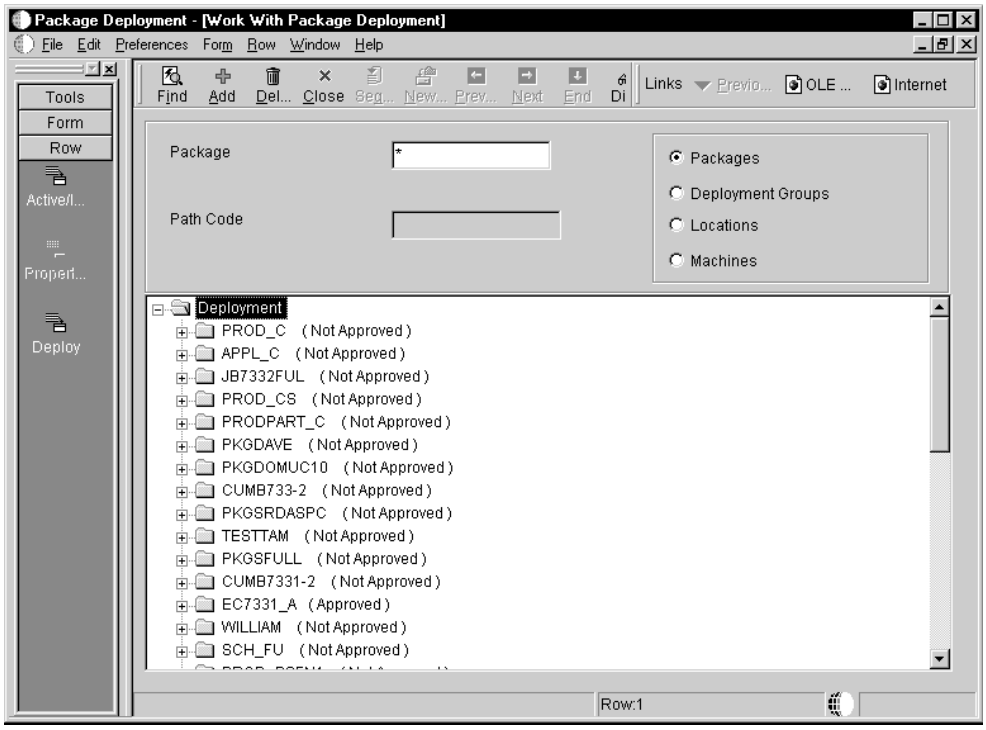
  Level Two: Client Workstation, Deployment Server, and Enterprise Server headings

  Level Three: Package deployment dates and times for each heading

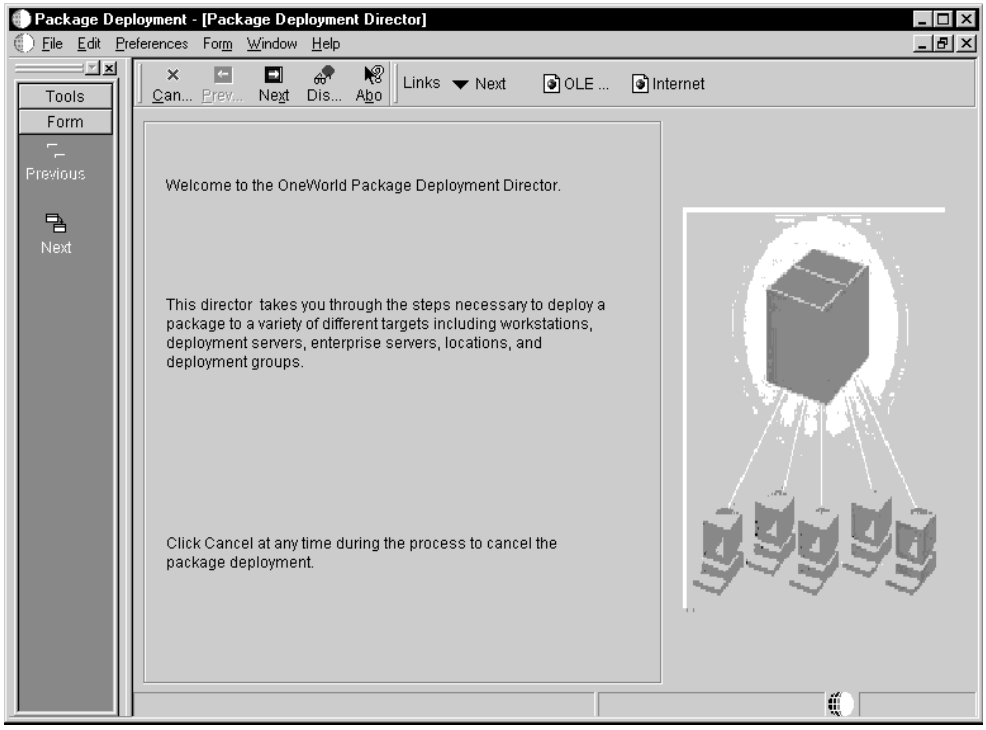
Level Four: Specific machines that have deployed that package for that date and time

► To schedule a package for deployment to a client workstation or server

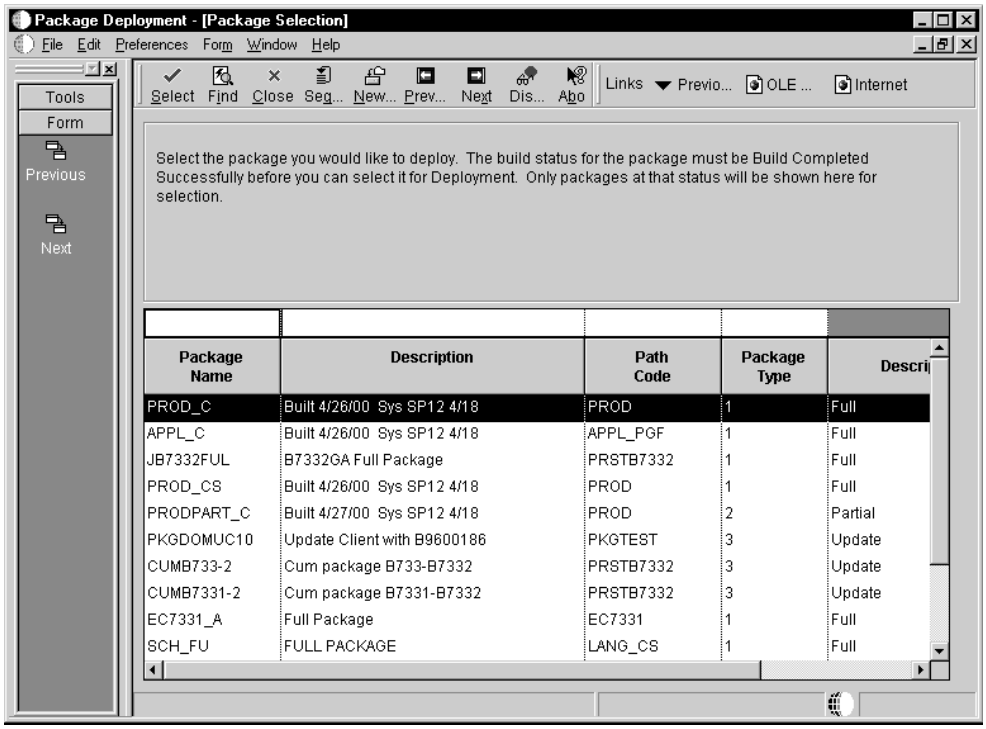
From the Package and Deployment Tools menu (GH9083), choose Package Deployment.



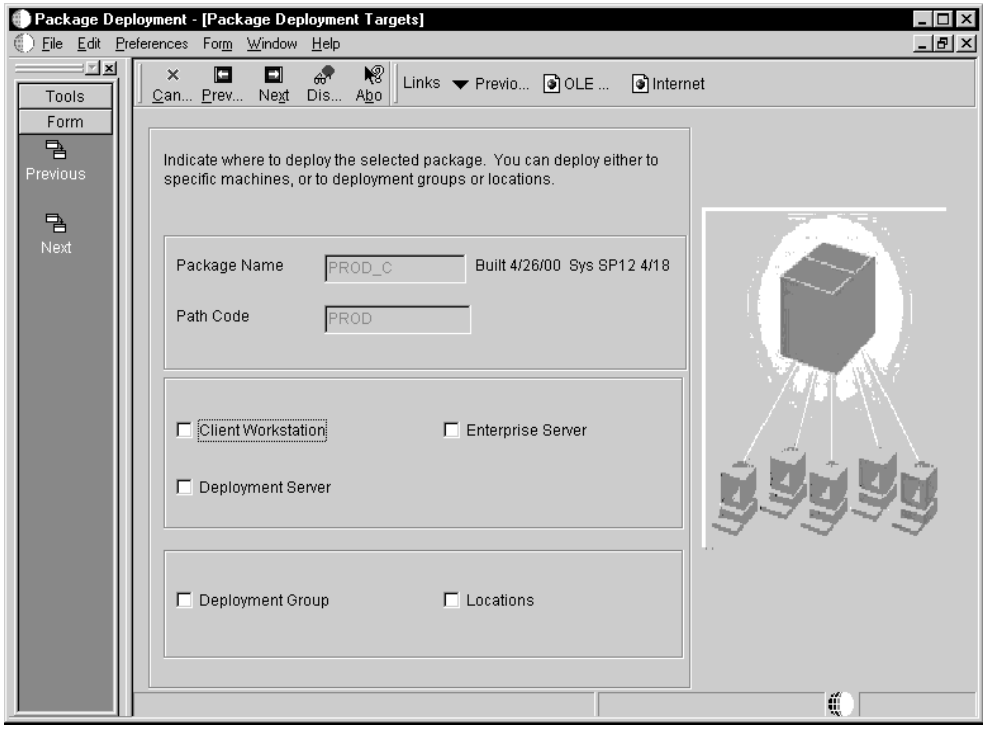
1. On Work with Package Deployment, click Add to launch the Package Deployment Director.



2. On Package Deployment Director, click Next.

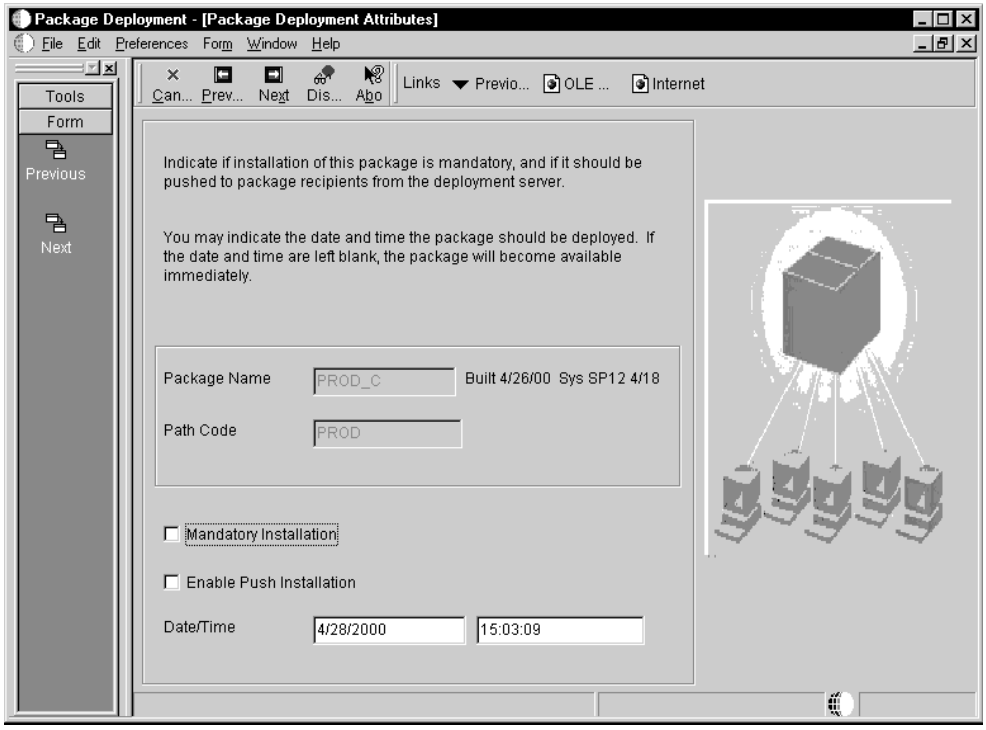


3. On Package Selection, choose the package that you want to deploy, and then click Next.



4. On Package Deployment Targets, click any of the following options to indicate the type of machines to which you want to deploy your package, and then click Next:

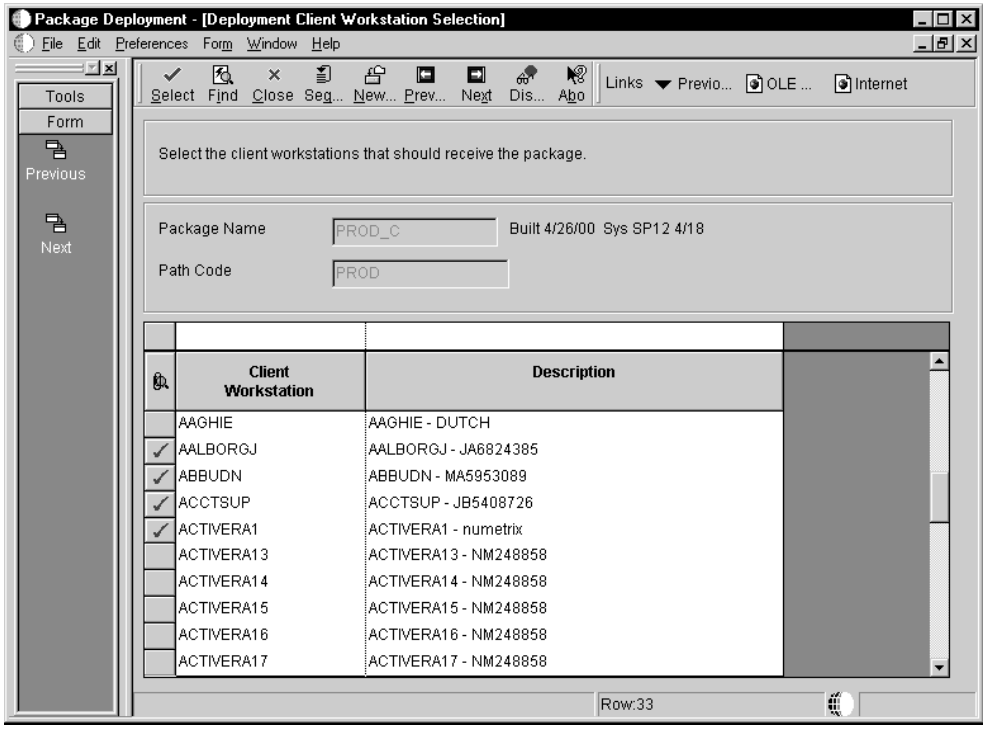
   - Client Workstation
   - Deployment Server
   - Enterprise Server



5. On Package Deployment Attributes, complete the following fields:
   - Mandatory Installation
   - Enable Push Installation
   - Date/Time
6. If you want to deploy the package using push installation, which pushes the package to workstations from the deployment server, click the Enable Push Installation option, and then click Next.

Note
For more information about push installation, see *Using Push Installation*.

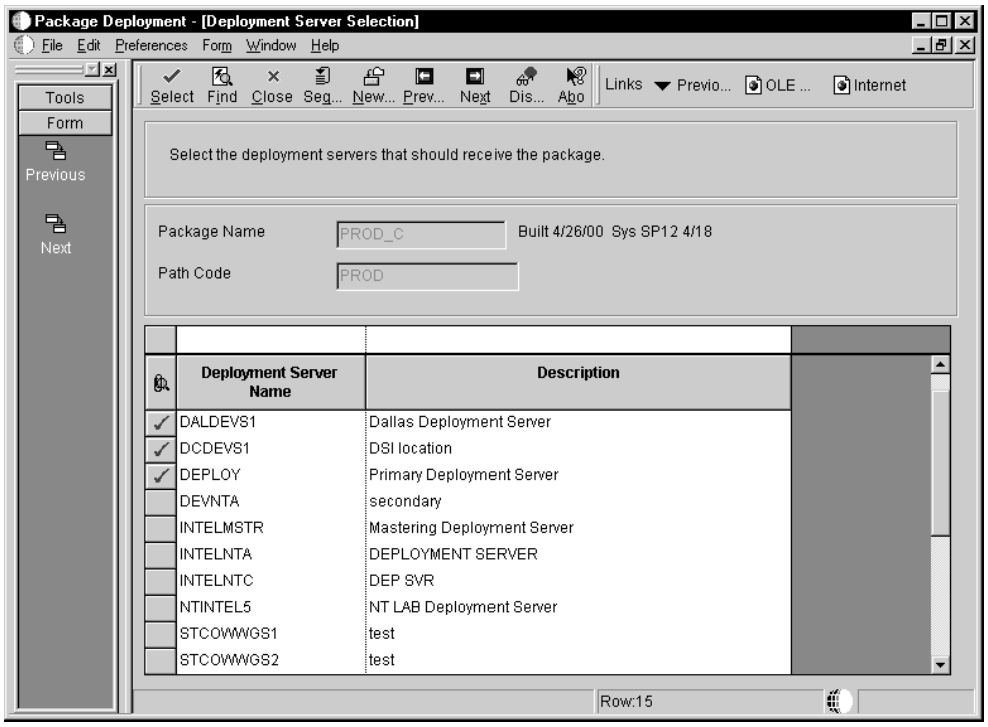
If you are deploying to workstations, the Deployment Client Workstation Selection form appears. If you are not deploying to workstations, skip the following step.



7. Find and choose the workstations to which you want to deploy the package, and then click Next.

   Choose a workstation by double-clicking in its row header. A check mark appears in the row header for each workstation that you choose.

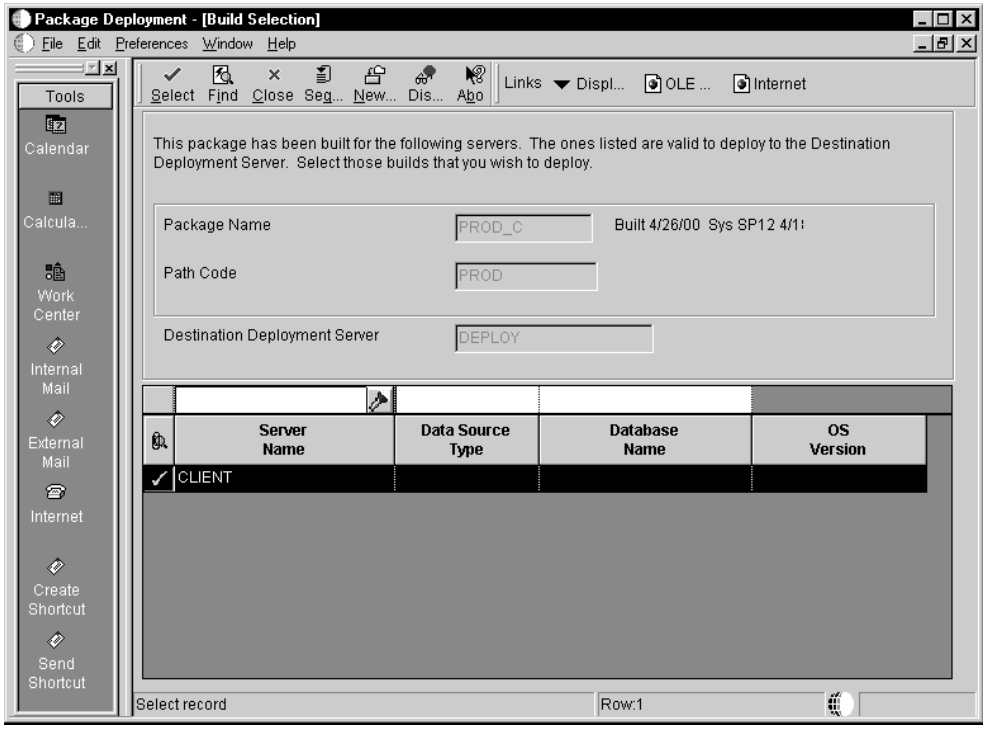
   If you are deploying to a deployment server, the Deployment Server Selection form appears. If you are not deploying to a deployment server, skip the following step.



8. Find the deployment server to which you want to deploy the package, and then click Next.

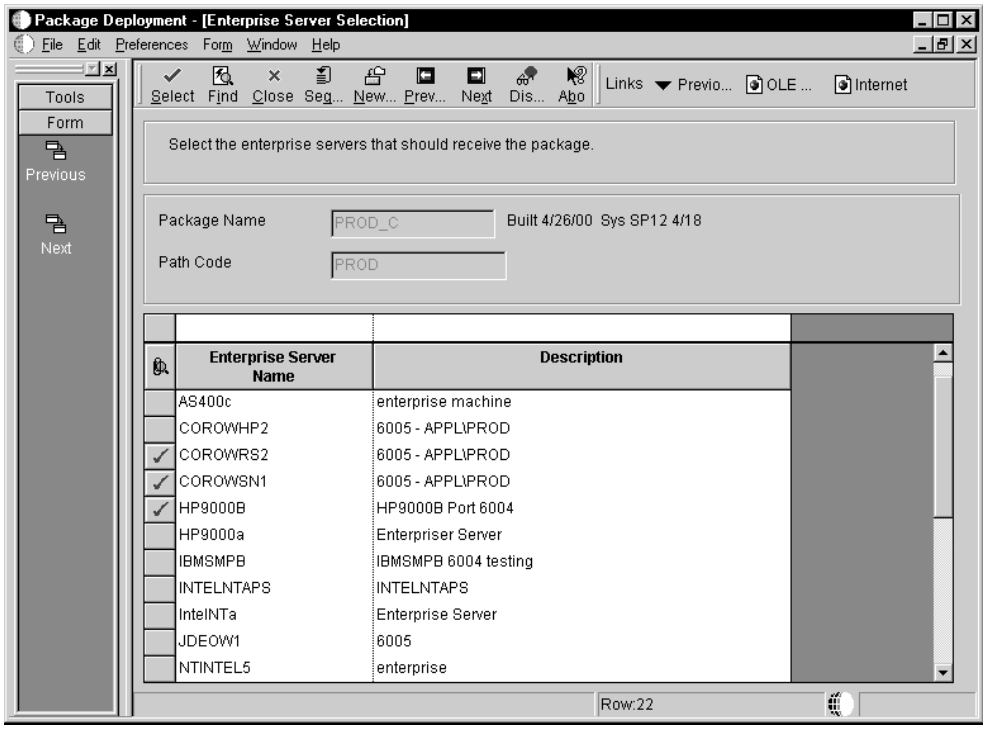
   Choose a server by double-clicking in its row header. A check mark appears next to each server that you choose.

9. On Build Selection, choose the server package build that you want to deploy to the destination deployment server, and then click Close.



10. Click Next.

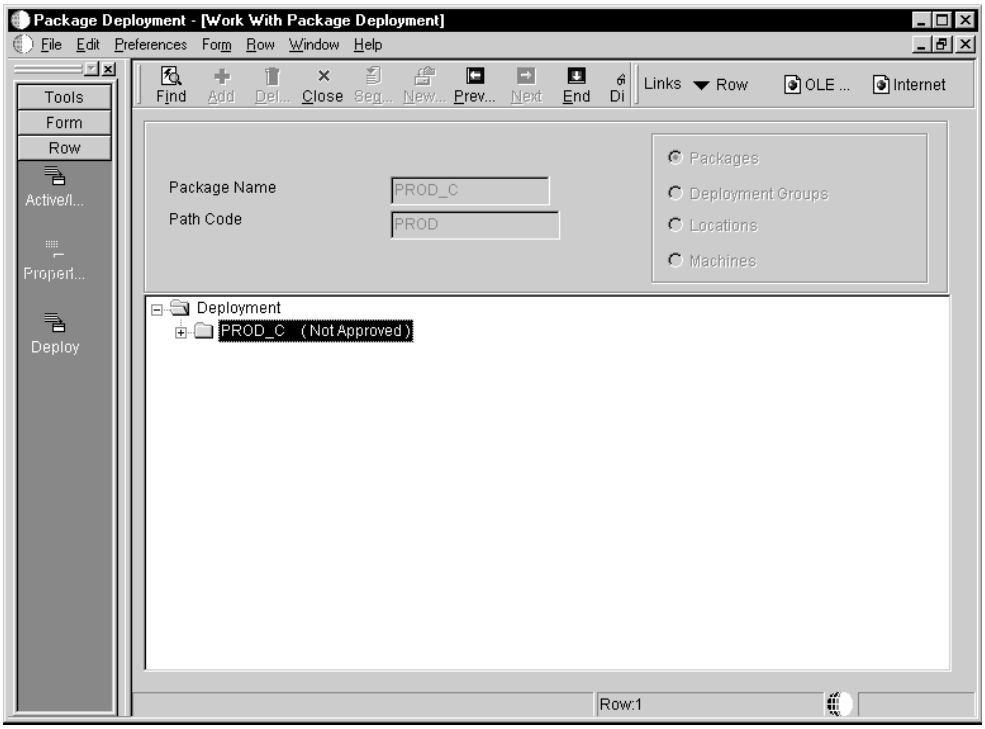
    If you are deploying to an enterprise server, the Enterprise Server Selection form appears. If you are not deploying to an enterprise server, skip the next step.



11. Find and choose the enterprise server to which you want to deploy the package, and then click Next.

    Choose a server by double-clicking in its row header.

12. On Work with Package Deployment, review your deployment selections.
13. To change any of your selections, click Prev to return to the appropriate previous form.



14. When you are finished reviewing and changing your deployment selections, click End.
15. If you are deploying a server package, find and choose the server package on the Work with Package Deployment form, and then choose Deploy from the Row menu.

    Note
    For more information about this step, see *Scheduling a Package for Push Installation*.

After you schedule your package for deployment, at the specified time on the date that you specified, the package deploys to workstations. This package becomes available to the user when the user signs on to ERP 9.0.

If you are using Push Installation, the package automatically installs at the time that you specify in the Schedule Jobs program (P91300).

Note

For more information, see *Using Push Installation*.

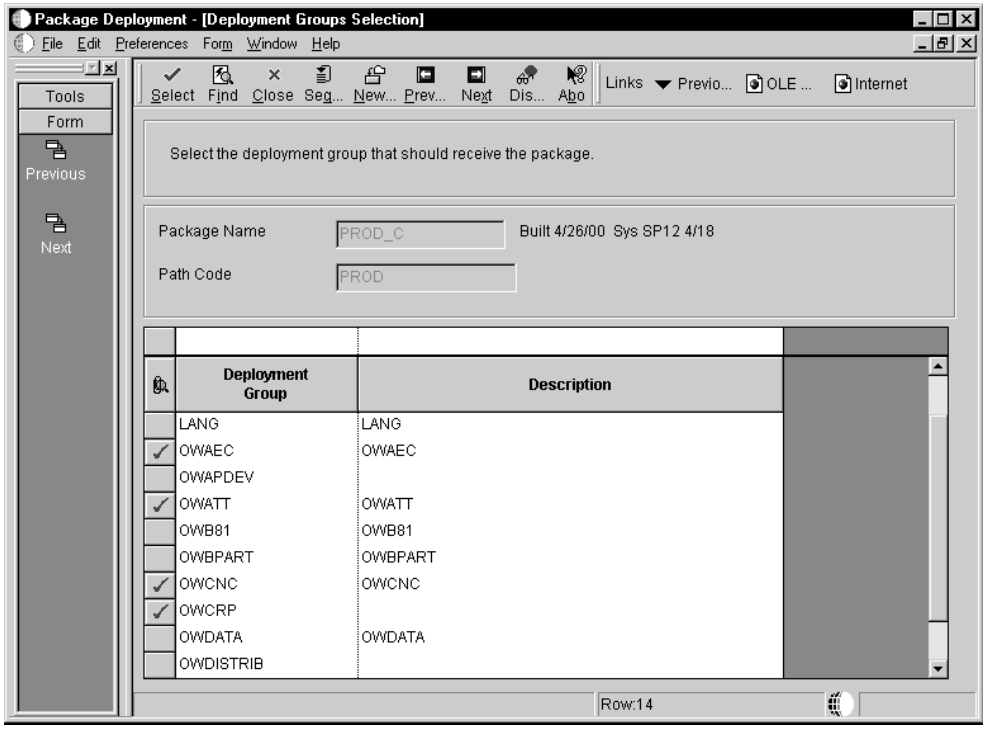
See Also

- *Using Push Installation* for more information about push installation
- *Scheduling a Package for Push Installation* for more information about how packages are deployed using Push Installation

► To schedule a package for deployment to a deployment group or location

From the Package and Deployment Tools menu (GH9083), choose Package Deployment.

1. On Work with Package Deployment, click Add to launch the Package Deployment Director.
2. On Package Deployment Director, click Next.
3. On Package Selections, choose the package that you want to deploy, and then click Next.
4. On Package Deployment Targets, choose either Deployment Group or Locations, and then click Next.
5. On Package Deployment Attributes, complete the following fields:
   - Mandatory Installation
   - Enable Push Installation
   - Date/Time
6. If you want to deploy the package using push installation, which pushes the package to workstations from the deployment server, click the Enable Push Installation option.
7. Click Next.

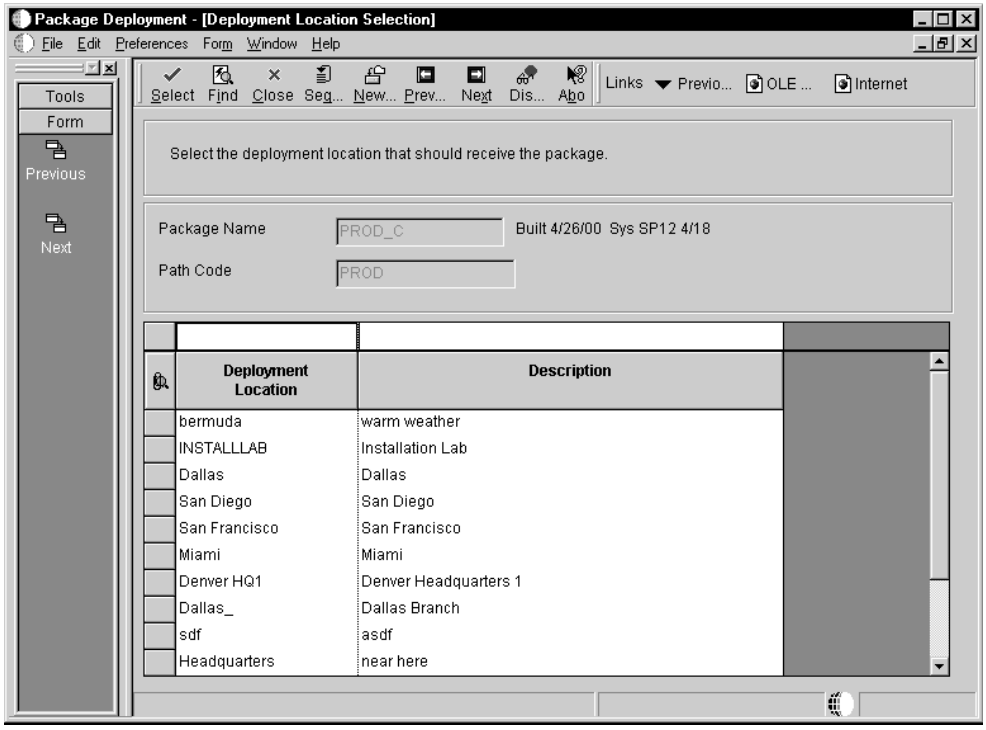


If you are deploying to a deployment group, the Deployment Groups Selection form appears. If you are deploying to a location, skip the next step.

8. Find and choose the deployment group that you want to receive the package, and then click Next.

   Choose a group by double-clicking in its row header.

9. If you are deploying to a location, the Deployment Location Selection form appears. Skip the next step if you are deploying to a deployment group.



10. Find and choose the deployment location that you want to receive the package, and then click Next.

    To choose a location, double-click the row header.

11. On Work with Package Deployment, review your deployment selections.
12. To change any of your selections, click Prev to return to the appropriate previous form.
13. When you are finished reviewing or changing your deployment selections, click End.
14. If you are deploying a server package, find and choose the server package on the Work with Package Deployment form, and then choose Deploy from the Row menu.

After you schedule your package for deployment, at the specified time on the date that you specified, the package deploys to workstations. This package becomes available to the user when the user signs on to ERP 9.0.

If you are using Push Installation, the package automatically installs at the time that you specify in the Schedule Jobs program (P91300).

See Also

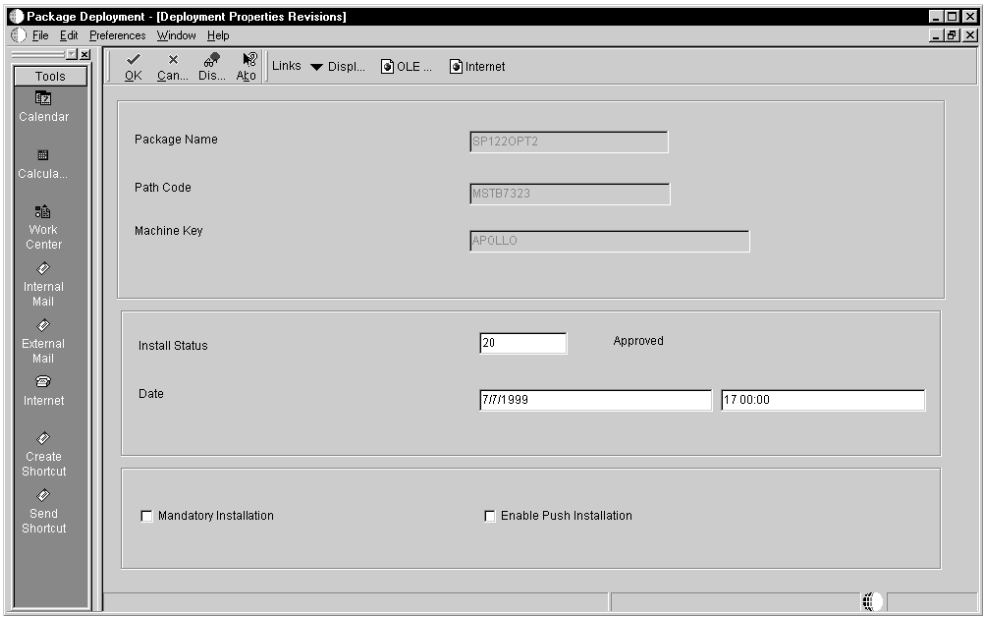
- *Working with the Package Deployment Director* for information about how to deploy a server package

► To revise deployment options

From the Package and Deployment Tools menu (GH9083), choose Package Deployment.

1. Choose Machines and click Find to display information according to machine name.

2. Find and choose the deployed package for which you want to modify the options, and then choose Properties from the Row menu. The Deployment Properties Revisions form appears.



3. Revise the information in any of the following fields:
   - Install Status
   - Date
   - Mandatory Installation
   - Enable Push Installation
4. When you are finished making revisions, click OK to return to the Work with Package Deployment form.

Activating the Scheduled Package

After you successfully define a package deployment, you must activate the package so that it is available for installation using the Workstation Installation program. If you do not activate the package, it will not be included in the list of available packages when users launch the Workstation Installation program.

In some situations, you might need to control which packages are available for installation. If, for example, you have a package that is for the testing group only, you would want to make that package inactive so that it is not available for installation through the Workstation Installation program. Instead, you can use Package Deployment program (P9631) to schedule this package for deployment to the members of the testing group.

► **To activate a package**

From the Package and Deployment Tools menu (GH9083), choose Package Deployment (P9631).

1. On Work with Package Deployment, click the Packages button and click Find.
2. Choose from the list the packages that you want to activate or inactivate.

   Alternatively, you can enter the package name in the Package field.
3. Choose Active/Inactive from the Row menu.

Installing a Scheduled Package

When users receive a package, they can choose to install it when they sign on to ERP 9.0 on or after the scheduled deployment date.

If the package is mandatory, users cannot access ERP 9.0 until they load the package.

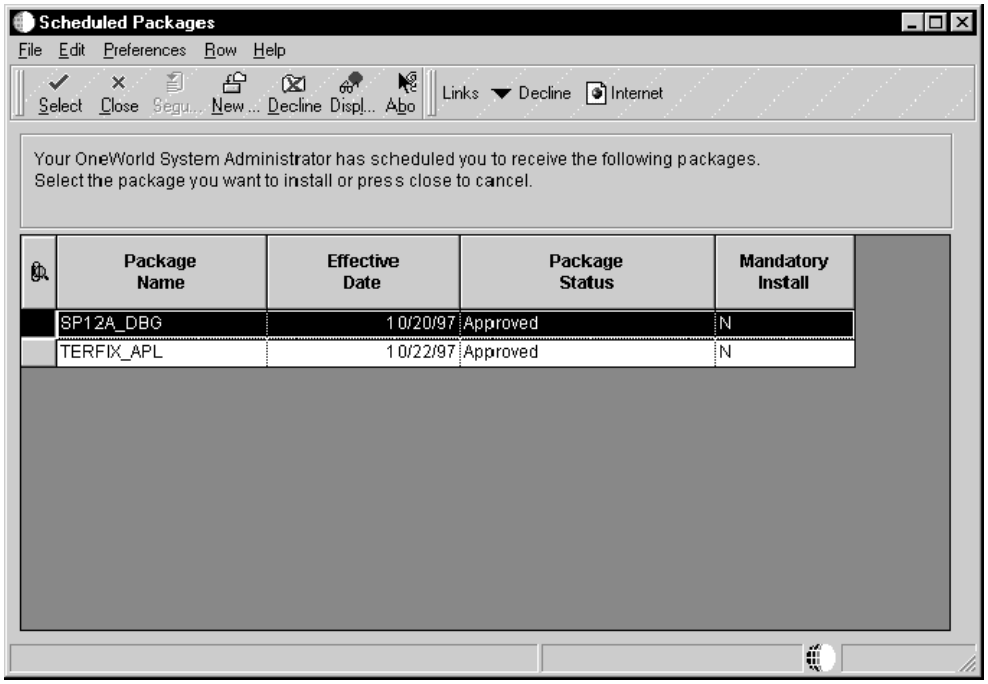
If the package is optional, users can decline the package or postpone the installation until later. If they decide to postpone the installation, ERP 9.0 Explorer launches, and they will be given the opportunity to install the package the next time that they sign on.

If a package that is scheduled for push installation fails to load for some reason (such as if the power to the workstation was turned off during the time that the package was scheduled to deploy), that package will be included in the list of available packages when the user signs on to ERP 9.0.

► To install a scheduled package

1. Sign on to ERP 9.0.

   When you are scheduled to receive a package, the Just-In-Time Installation program launches and the Scheduled Packages form appears.



2. Perform one of the following steps:

   - To load the package immediately, skip to the next step.
   - To decline the package permanently, choose Decline from the Row menu.
   - To list all items in the package, choose Package Detail from the Row menu.
   - To load the package at another time, click Close. If the package is mandatory, you will be unable to access ERP 9.0 Explorer until you load the package.

3. To load the package, choose one or more packages you want to install and click Select.

   The Workstation Installation program loads the package. If you selected more than one package, the program installs them sequentially. When the installation is complete, ERP 9.0 Explorer starts.

Deploying Server Packages

The process for deploying a server package is nearly identical to deploying a package to a workstation. In both cases, you need to assemble, define, build, and schedule the package for deployment by using the Package Assembly (P9601), Package Build (P9621), and Package Deployment (P9631) programs.

After you schedule a server package for deployment, you must complete an additional step to launch the batch program that enables you to deploy to servers. You must perform this task whenever you deploy a package to an enterprise server or deployment server.

Caution

Deploy server packages only when necessary, because the enterprise server is not available to process business applications and batch processes during the installation process. The enterprise server does not actually shut down during package installation. Instead, ERP 9.0 queues any jobs that are submitted to the enterprise server and runs them as soon as the installation finishes. For this reason, J.D. Edwards recommends that you schedule enterprise server packages to be deployed after hours to minimize impact on users. (Before you deploy a package to an enterprise server, verify that the ERP 9.0 services will be running during the deployment.)

To further minimize impact on the network and users, if your development environment is on the same enterprise server as your production environment, consider preventing developers from moving their own objects through server packages. Instead, require that an administrator perform this function.

To deploy a server package, choose the Deploy function from the Row menu on Work with Package Deployment. This is the same function that you use to deploy packages to deployment servers during multitier deployment.

ERP 9.0 determines which of the following batch programs to call, based on what is currently selected on the Work with Package Deployment form when you choose the Deploy function from the Row menu:

- If a specific deployment server is selected, the system launches the Multi Tier Deployment batch program (R98825C).
- If the Deployment Server folder is selected, the system launches the Multitier Deployment batch program for every deployment server that has a package scheduled.
- If a specific enterprise server is selected, the system launches the Enterprise Server Deployment batch program (R98825D).
- If the Enterprise Server folder is selected, the system launches the Enterprise Server Deployment batch program for every enterprise server that has a package scheduled.
- If a specific package is selected, the system launches the Multi Tier Deployment batch program, and then the Enterprise Server Deployment batch program for the selected package.
- If you sort by packages and the Deployment folder is selected, the system launches both the Multi Tier Deployment batch program and Enterprise Server Deployment batch program for all packages.

If a specific workstation or the Workstations folder is selected, the Deploy option is unavailable.

When the system launches a batch program for all servers or all packages, deployment does not occur unless the package has been previously scheduled for a specific server.

Before You Begin

- ❑ Assemble the server package.

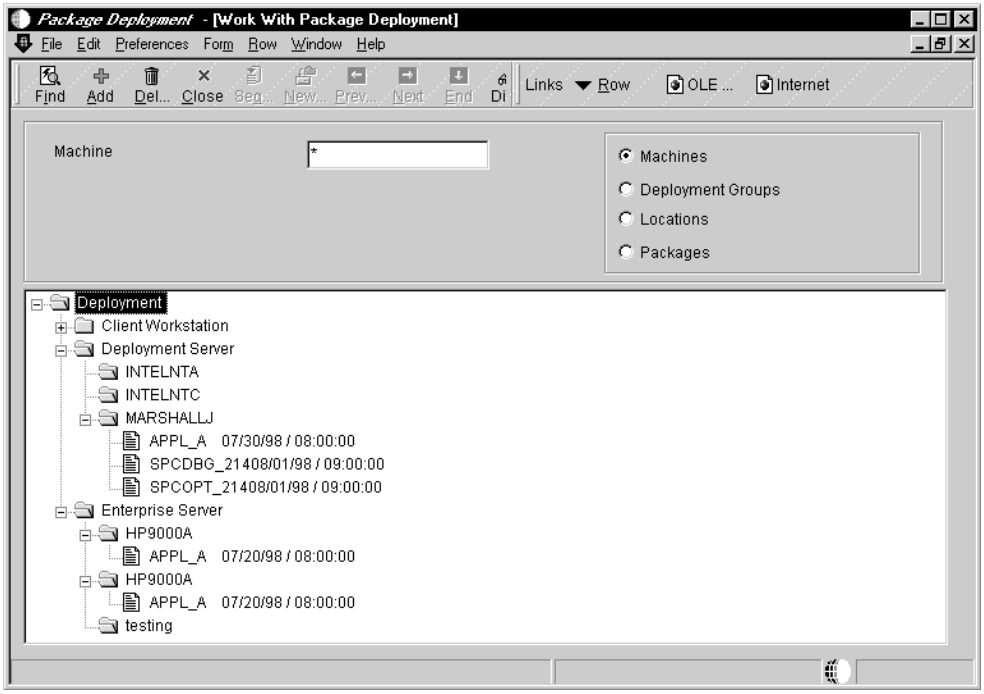
- Define the server package.
- Build the server package.
- Schedule the package for deployment to the appropriate server.

See Also

- *Scheduling the Push Installation Batch Application* for information about how to launch a batch application that deploys the package to servers

► To deploy a server package

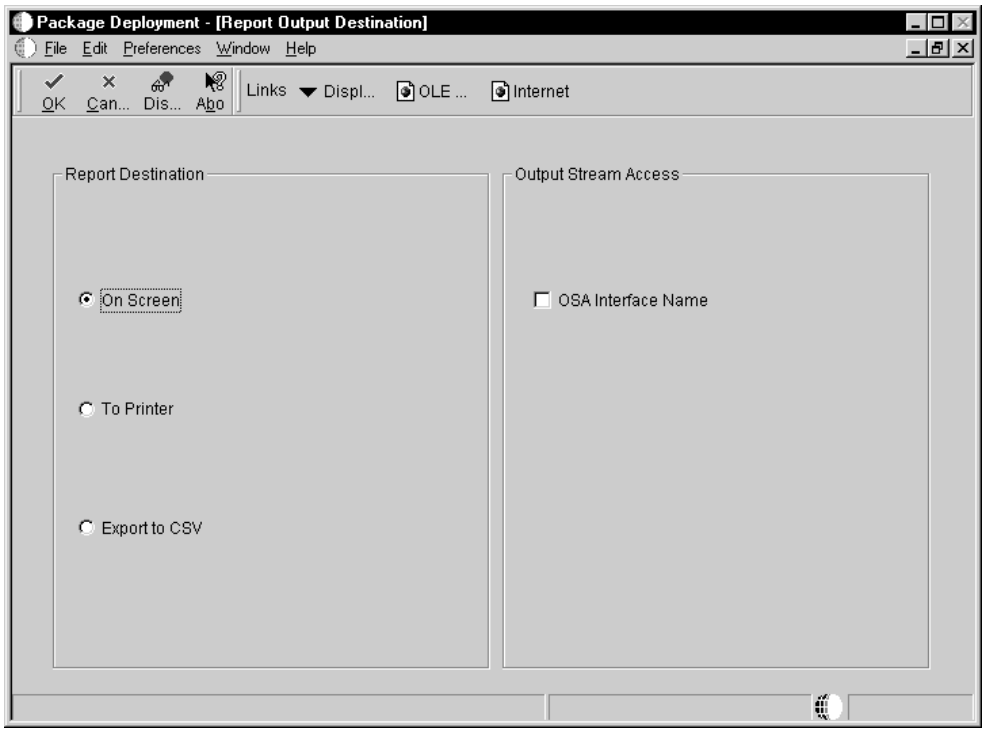
From the Package and Deployment Tools menu (GH9083), choose Package Deployment.



1. On Work with Package Deployment,locate the server package that you want to deploy.

   Alternatively, choose the enterprise server or, if the package is scheduled to deploy to more than one server, the Enterprise Servers folder.

2. Choose Deploy from the Row menu.



3. On Report Output Destination, select On Screen.
4. Click OK to continue.

Using Push Installation

Push installation is the only deployment method that provides automatic and unattended package deployment. This means that the ERP 9.0 system administrator can deploy a package (or several packages) to a workstation or group without requiring any action from workstation users.

For example, an administrator might schedule a package to deploy to a particular group after hours. When members of that group report to work the following morning, that package is available for immediate use.

Push installation is particularly useful in situations in which you need to quickly deploy packages with a minimum of intrusion or impact upon your normal production and development routines. By planning and scheduling package deployment judiciously, administrators can also minimize the impact upon network performance that can accompany large numbers of package deployments. Administrators can also use push installation to install ERP 9.0 on a workstation for the first time. This ability can greatly minimize downtime and provide maximum deployment flexibility.

Understanding Push Installation

During push installation, package contents are pushed from the deployment server to the workstation. In contrast to push installation, the Workstation Installation program pulls package contents from the deployment server to the workstation. Installations that are set up to use Package Deployment (P9631) for scheduled packages that are not push enabled also pull packages.

The end result of the deployment is the same, regardless of whether package contents are pushed or pulled. However, the advantage of a push installation is that no action is required from the workstation user other than to leave the workstation turned on during the time when the package is scheduled to deploy.

For a partial package or an update package that contains program specifications, the term *package contents* refers to specifications. For a full package or an update package that does not contain application specifications, *package contents* refers to objects.

The following list summarizes the steps in the push installation process:

1. Install a push installation Listener program on each workstation that will receive pushed packages. This Listener monitors the progress of the Push Package Installation batch program (R98825) that runs on the server and performs functions such as monitoring installation status. The Listener can run as either a local service or a network service.
2. Schedule the package using the Package Deployment program (P9631). The Push Package Installation batch program reads the scheduling table and sends a message to the Listener on all workstations for which a package has been scheduled.
3. Use the Schedule Jobs program (P91300) to launch the batch program on the enterprise server. This program enables you to specify the job name, version, start date, start time, and recurrence.
4. At the specified start time, the Schedule Jobs program launches the Push Package Installation batch program (R98825), which initiates the package installation process from the deployment server. The Listener and the batch program interact during the process until installation is complete. Codes are passed from the Listener to the batch program to indicate the installation status (such as failed, successful, in progress, and so on).
5. When the installation finishes, ERP 9.0 sends an e-mail message to the primary user of the workstation. This message indicates whether the installation was successful. E-mail notification works only if the package recipient is listed in the Machine Master table (F9650) and has an e-mail address in the profile.
6. If the push installation fails for some reason (such as when the package recipient neglects to leave the workstation turned on), the installation status changes to Failed. If you want to reschedule the installation, you must first delete the row with the failed job, and then schedule the job again.

If the push installation is not successful, when the user signs on to ERP 9.0, the standard scheduling screen appears. At this point, the user can either accept the mandatory package or exit ERP 9.0.

Preparing the Enterprise Server for Push Installation

To set up your server for push installation, you must first install and configure the Microsoft Domain Name Service (DNS) that is included with Microsoft Windows NT Server 4.0. If you have not yet set up a domain name service, you can install Microsoft DNS by selecting the Network icon in the Control Panel, then selecting the Services tab, and then adding Microsoft DNS Server.

After you add Microsoft DNS, you must configure the DNS by specifying your domain name and servers.

UNIX and AS/400 Considerations

In an environment configured for Dynamic Host Configuration Protocol (DHCP), a server must run Windows NT Server 4.0 to resolve workstation addresses because the Windows NT server dynamically assigns them.

To enable name resolution, you need to configure your servers to resolve their IP address lookup through a Windows NT DNS server, which, in turn, must be configured to review the WINS database when DHCP is enabled in the network domain.

Configuring your servers in this way ensures that the following process flow occurs during the push installation process:

1. From the host server on which the Push Installation batch program (R98825) runs, a business function attempts to retrieve the machine host address from the DNS server.
2. Because the DNS server does not contain IP addresses, it retrieves the address from the WINS server.
3. The WINS server returns the address to the DNS server.
4. The DNS server returns the address to the host server.
5. The host server finds the Listener on the client workstation and sends workstation installation information.
6. The workstation installation process starts.

Preparing Workstations for Push Installation

Before you can push an installation to a workstation, you must install a Listener on the workstation, which interacts with a business function that runs on the server. You must install this Listener on all workstations that you want to be enabled for push installation, regardless of whether you want to deploy packages to a machine on which ERP 9.0 is already installed or a machine on which you are installing ERP 9.0 for the first time.

The Listener runs continuously on the Windows NT workstation as a service (and on a Windows 95 machine as a pseudo-service), and administrators can monitor it using the Task Manager. The Listener communicates with the batch application on the server, receiving and sending messages during the installation process. The Listener also monitors the progress of the installation and saves the installation completion code.

Installing the Listener

When you install the Listener on the workstations in your enterprise, you specify whether to run a local service or a network service. If you run the service locally on the workstation, the user must be signed on to receive a package that has been scheduled for push installation. If you run the service on the network, the Listener runs as a network account and the user does not have to be signed on to receive a package through push installation. The network service must have an administrator's user ID.

The disadvantage of running the service on the network is that it can be difficult to administer for all users on the enterprise. For example, because the parameters of the Listener apply to every user on the network, push installation users must install ERP 9.0 to and from the same locations. One user could not have ERP 9.0 on drive C and another user have the same release on drive D. Also, every time users change their signon passwords, the system administrator must update the Listener service with the new passwords in order for the service to work for those users. For these reasons, J.D. Edwards recommends that you install the Listener locally on each workstation.

Whether you run the Listener as a local service or network service, the workstation must be turned on to receive a scheduled package.

You can choose from one of the following ways to install the Listener on a workstation:

- Use a third-party software distribution system, such as the Tivoli Management Environment (TME10) Software Distribution System or the Microsoft System Management Server (SMS) software
- Distribute an executable installation program (a setup.exe file) and the accompanying ancillary files via an intranet Web site or the Worldwide Web
- Use Windows NT logon scripts (a .bat file) to call a C program
- Install from the Worldwide Web

Caution

If the Listener is not installed and running on the workstation (or if the workstation is turned off), the push installation cannot occur. After you schedule a package, remind package recipients to leave their workstations on and the Listener service running, even during off-hours. If you set up the Listener to run as a local service, also remind users to remain signed on.

Before You Begin

- ❑ Close any applications that are currently open.
- ❑ Verify that the destination directory where you will be installing the Listener has sufficient disk space. You need approximately 2MB of free disk space to install all of the Listener files and components.

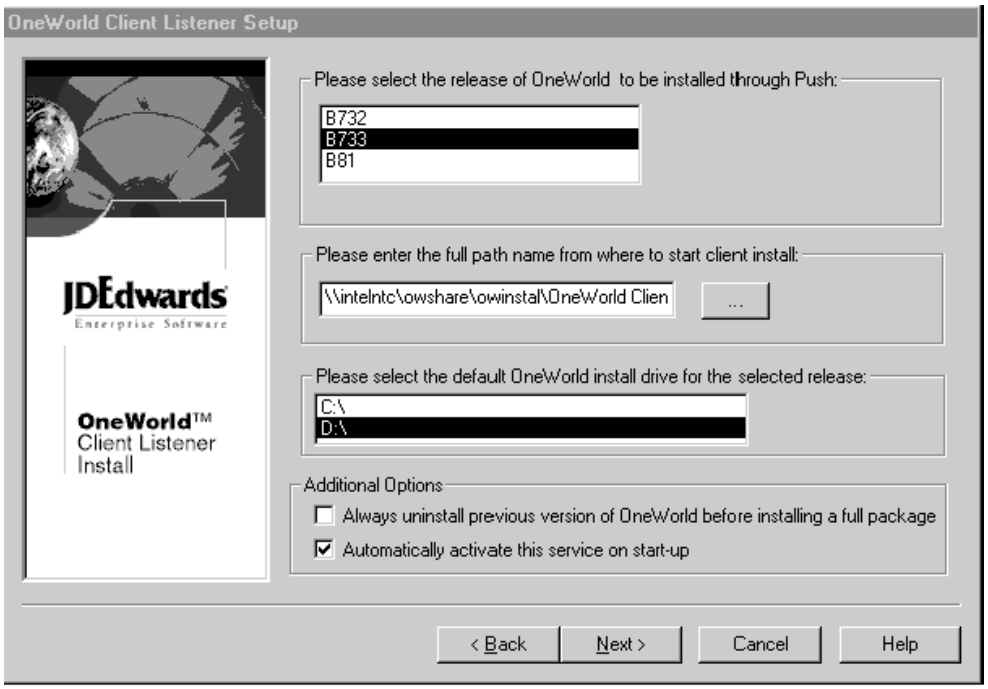
► To install the Listener on workstations

The following task describes how to install the Listener by launching an installation program (that is, a setup.exe program). You can distribute this program to users on your enterprise by using e-mail to send them either the program or a shortcut, or by describing where the program is located on your server.

If you have a previous version of the Listener already installed, the installation program removes the previous version before copying the new version to your workstation.

1. Launch the installation program by double-clicking the setup.exe icon.

2. On the first ERP 9.0 Client Listener Setup form, click Next.



3. On the second ERP 9.0 Client Listener Setup form, complete the following fields:

   - ERP 9.0 release

     Choose the ERP 9.0 release that you want to install through push installation.

   - Path name

     For the ERP 9.0 release that you selected, enter the full path name on the deployment server from which to initiate the ERP 9.0 installation.

   - ERP 9.0 installation drive

     Specify the drive on which you want to install the specified release of ERP 9.0.

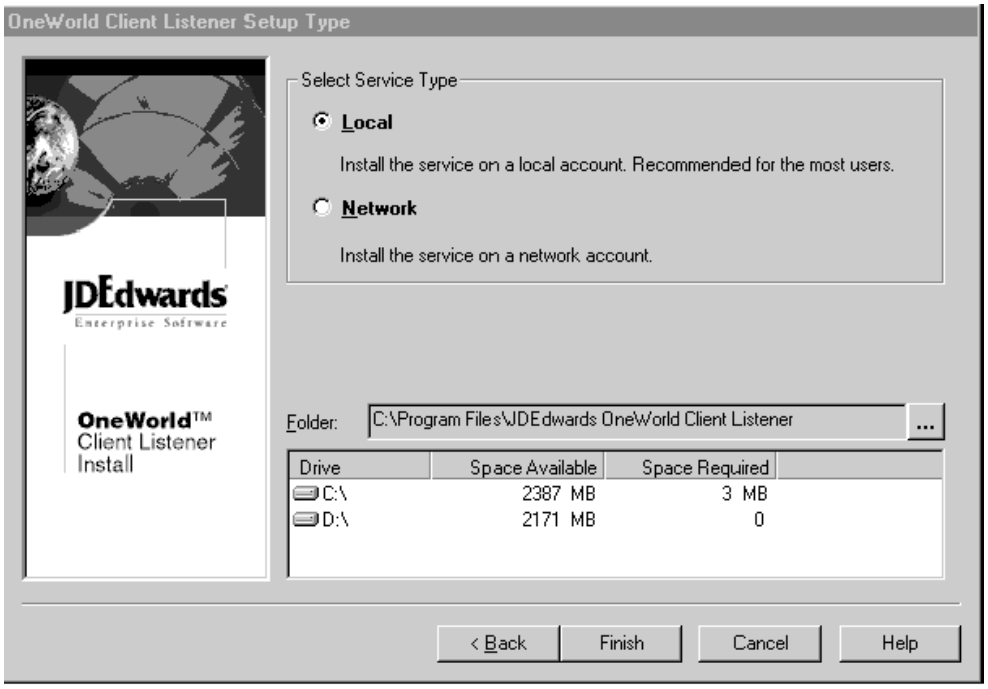
   - Uninstall previous releases

     Turn on the uninstall ERP 9.0 option to uninstall existing versions of ERP 9.0 before installing a new full package.

   - Start the Listener automatically on startup

     Turn on the autostart option to automatically start the Listener service whenever your workstation starts up.

4. Click Next to proceed to the next installation form.



5. Click one of the following options:

   - Local
   - Network

     Unless your system administrator tells you to install the Listener on the network, click Local to install the Listener on your local workstation.

6. Complete the following field:

   - Folder

     Specify the destination drive and folder in which the Listener files will reside.

7. Click Finish to complete the installation.

After you have successfully installed the Listener on the workstation, a small ear icon appears on the Windows taskbar, indicating that the Listener has been loaded. By right-clicking this icon, you can start or stop the Listener, or change the default parameter settings.

See Also

- *Changing Default Parameter Settings* in the *Package Management Guide* to reconfigure the Listener

Installing the Listener Using Silent Installation

In some cases, it might be more convenient to install the Listener on workstations via silent installation. Using this method, the ERP 9.0 administrator enters configuration settings for all workstations and distributes a batch file that automatically initiates the Listener installation the next time that the user signs on to the workstation.

The advantage of using silent installation to install the Listener is that the process is transparent to workstation users, and users are not required to enter configuration information or step through the installation process.

The following task describes how to install the Listener via silent installation. Typically, the ERP 9.0 administrator performs this task.

► To install the Listener using silent installation

1. Edit the following settings in the listen_silent_setup.inf file that is included on the software CD:

   - ServiceType

     Enter Local or Network, depending on where you want to run the Listener service.

   - WorkstationDirPath

     Enter the location on the workstation where you want to install the Listener program and related files. For example, C:\Program Files\OneWorld Client Listener.

   - Release

     Enter the ERP base release. For example, B9. Do not enter a cumulative update release, such as B9.1.

   - InstallPath

     Enter the location on the workstation where ERP 9.0 is installed. For example, D:\b9.

   - LaunchPath

Enter the deployment server name and the location from which the Client Workstation Installation program runs. For example, `\\`*server name*`\b9\OneWorld Client Install\setup.exe`.

- AutoStart

  Enter 1 to automatically start the Listener service when the workstation starts up. Enter 0 if you do not want to enable Autostart.

- UninstallPackage

  Enter 1 if you want to automatically uninstall previous versions of ERP 9.0 before installing a new full package. Enter 0 if you do not want to enable automatic uninstall.

2. Create or modify a batch file to include the silent installation parameter `/s` for the ListenSetup.exe program. The batch file must reside in the same location as the ListenSetup.exe program.

   For example, your batch file might contain the following line:

```
start \\servername\b9\client\misc\ListenSetup.exe /s
listen_silent_setup.inf
```

3. Distribute the inf file and the batch file to workstation users. You can distribute these files or place them on a network server where workstation users can copy the files to their workstation.
4. Instruct users to restart their workstations to run the batch file and load the Listener via silent installation.

After workstation users have successfully installed the Listener, the Listener icon appears on the Windows taskbar. Users can click this icon to start and stop the Listener or change Listener settings.

Caution

You cannot change the name of the Listener silent installation file that is shipped with ERP 9.0. The file name must be listen_silent_setup.inf.

Stopping the Listener

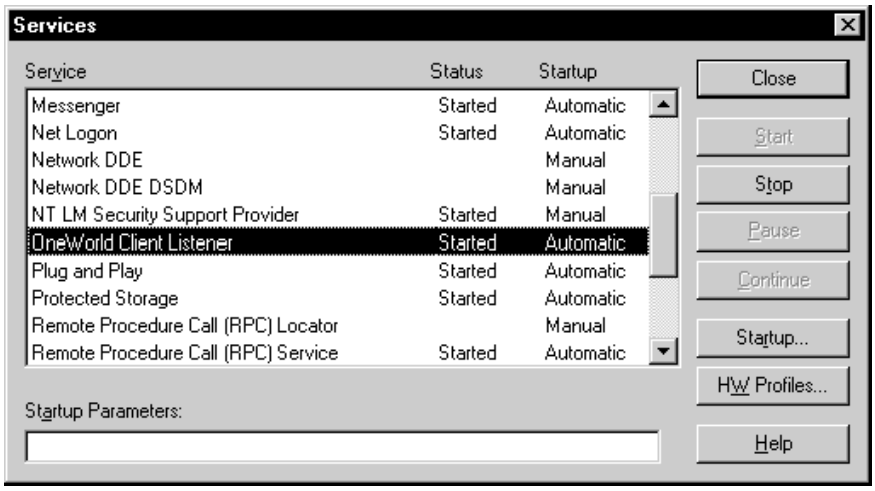
You can stop the Listener if you are certain that you do not want to use push installation to install ERP 9.0 packages. If you change your mind later, you can restart the Listener.

► To stop the Listener

The easiest way to stop the Listener is to right-click the Listener icon on the Windows taskbar and choose Stop Listener.

Alternatively, you can stop the Listener using the following steps:

1. Open the Control Panel.
2. Choose Services.



3. Choose OneWorld Client Listener.
4. Click Stop.

► To uninstall the Listener

1. Open the Control Panel.
2. Choose Add/Remove Programs.
3. Choose OneWorld Client Listener.
4. Click Remove All Components.

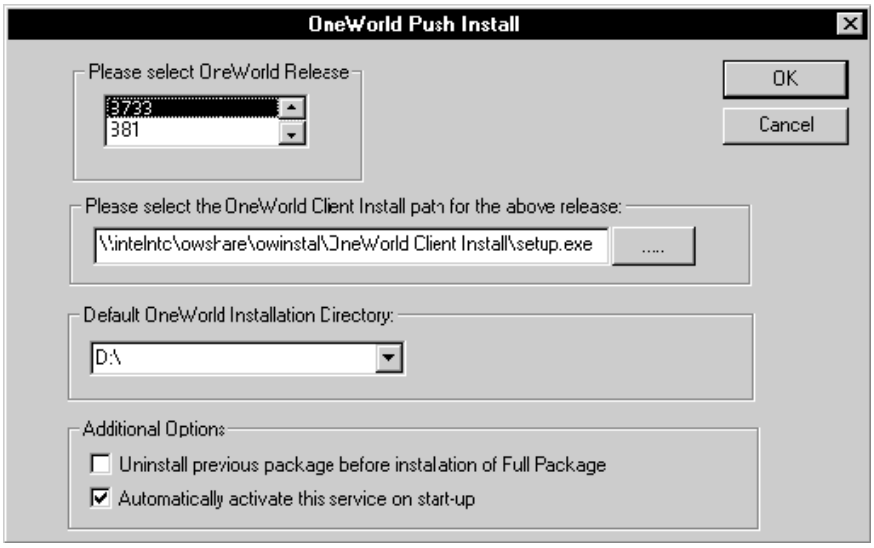
Changing Default Parameter Settings

You can use the icon for the Listener that appears on the Windows taskbar to change the default parameter settings that you entered when you initially installed the Listener. You can specify the following:

- The ERP 9.0 release and client installation path for that release
- The default ERP 9.0 installation directory
- Whether to uninstall the previous package before installing a new full package

► To change default parameter settings

1. Right-click the Listener icon in the Windows taskbar.
2. Choose Change Default Parameters.



3. Complete the following fields:

   - ERP 9.0 release

     Choose the ERP 9.0 release that you want to install through push installation

   - Path name

     For the ERP 9.0 release that you selected, enter the full path name on the deployment server from which to initiate the ERP 9.0 installation. If necessary, click the button to the right of the field to browse for available paths.

   - Default ERP 9.0 installation directory

     Specify the drive on which you want to install the specified release of ERP 9.0.

   - Uninstall previous package before installation of Full Package

     Indicate whether you want to uninstall previous packages before you install a full package. When you turn on this option, ERP 9.0 completely uninstalls the existing package before it installs a new full package.

   - Automatically activate this service on start-up

     Turn on this option if you want the Listener service to automatically start when the user signs on to the workstation.

4. Click OK when you are finished making changes.

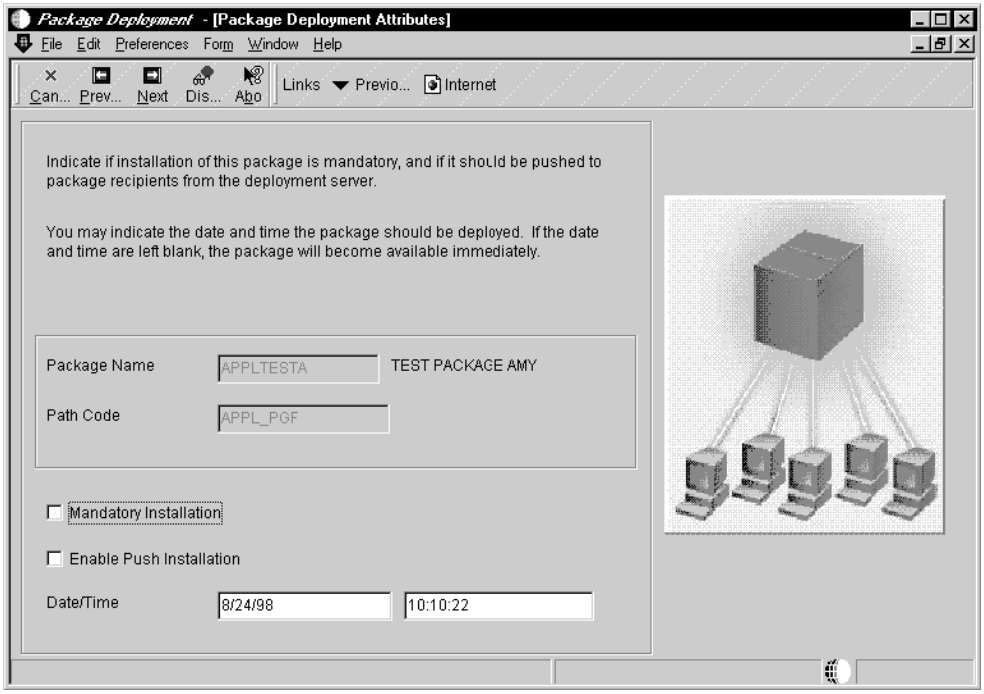
Scheduling a Package for Push Installation

After you have installed the Listener on your workstations, you can schedule a package for deployment.

The process for scheduling a package for push installation is identical to the process for scheduling a package using the Package Deployment program (P9631). When you schedule the package using this program, turn on the Enable Push Installation option on the Package

Deployment Attributes form. If you do not turn on this option, the package will be deployed through normal scheduled deployment.

When you use the Schedule Jobs program (P91300), you can set recurrence, which determines how frequently the job runs until it finishes successfully. If you do not set recurrence, the job runs only one time. In the case of Push Installation, recurrence determines the interval of time between installation attempts. After the package is successfully deployed, the job ceases to run.



As with scheduling any other package for deployment, all machine names (that is, package recipients) must be defined in the Machine Master (F9650) table. This table is populated when users sign on to ERP 9.0. Alternatively, you can enter machine names manually via the Machine Identification program (P9654A).

Scheduling the Push Installation Batch Application

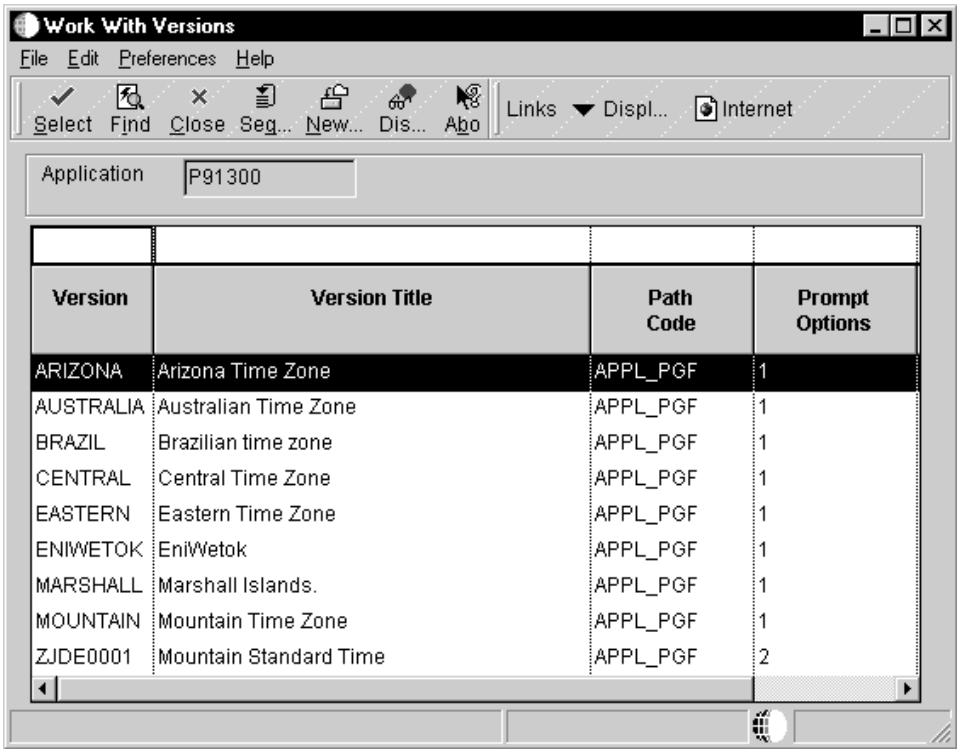
After you have installed the Listener on all affected workstations and have scheduled the package through the Package Deployment program (P9631), you must use the Schedule Jobs program (P91300) to run the Push Package Installation batch program (R98825) on the server.

Before You Begin

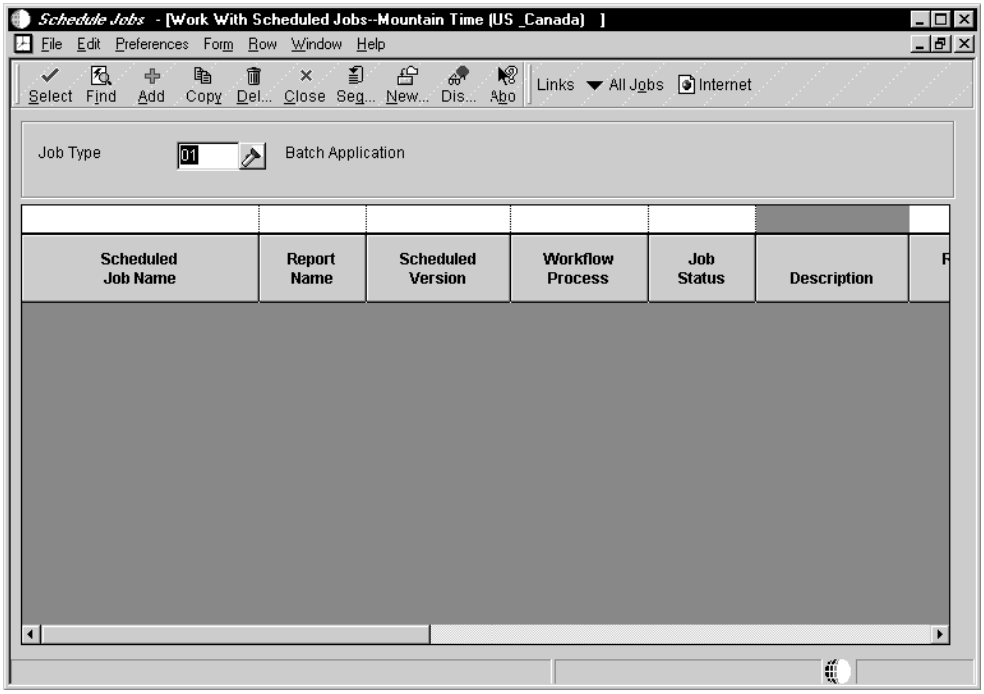
- Remind package recipients to leave their workstations turned on, even after hours.
- Remind users who are using a local service that they must be signed on.
- Remind package recipients to verify that the Listener is running.

► To schedule the Push Installation batch application

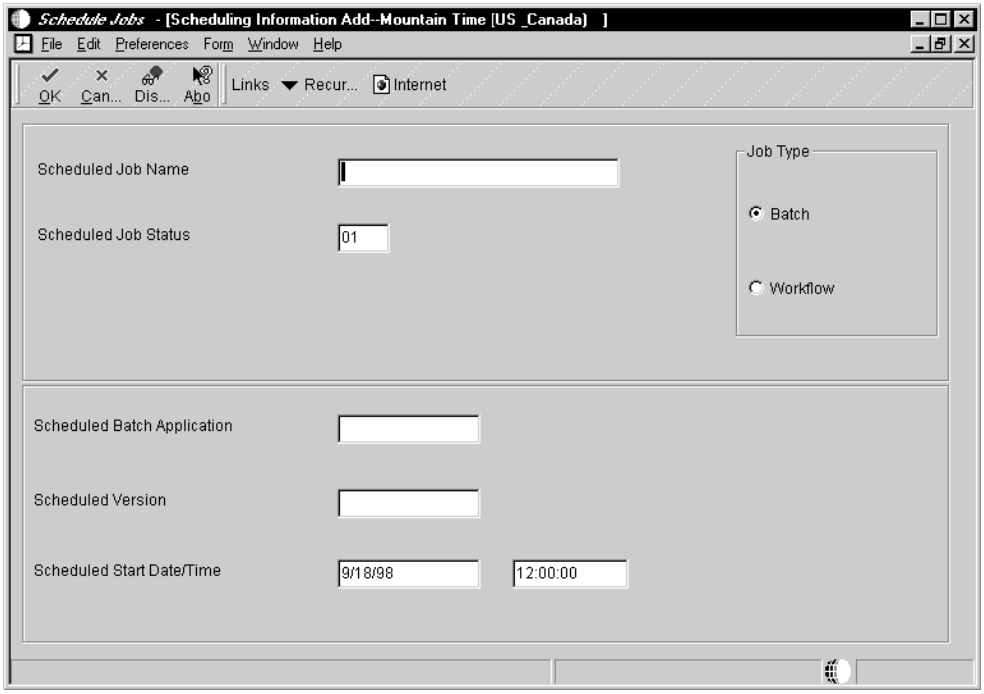
From the Job Scheduler menu (GH9015), choose Schedule Jobs (P91300).



1. On Work with Versions, choose the time zone that applies to your setup.



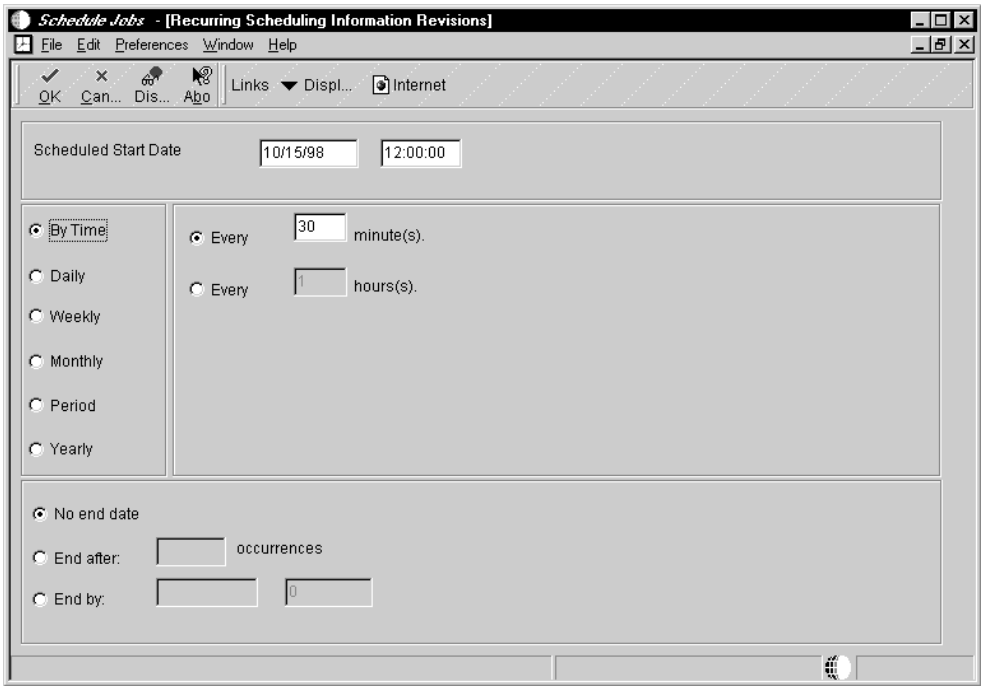
2. On Work with Scheduled Jobs, click Add to enter a new job.



3. On Scheduling Information, click the following option:
   - Batch
4. Complete the following fields:
   - Scheduled Job Name
   - Scheduled Job Status
   - Scheduled Batch Application

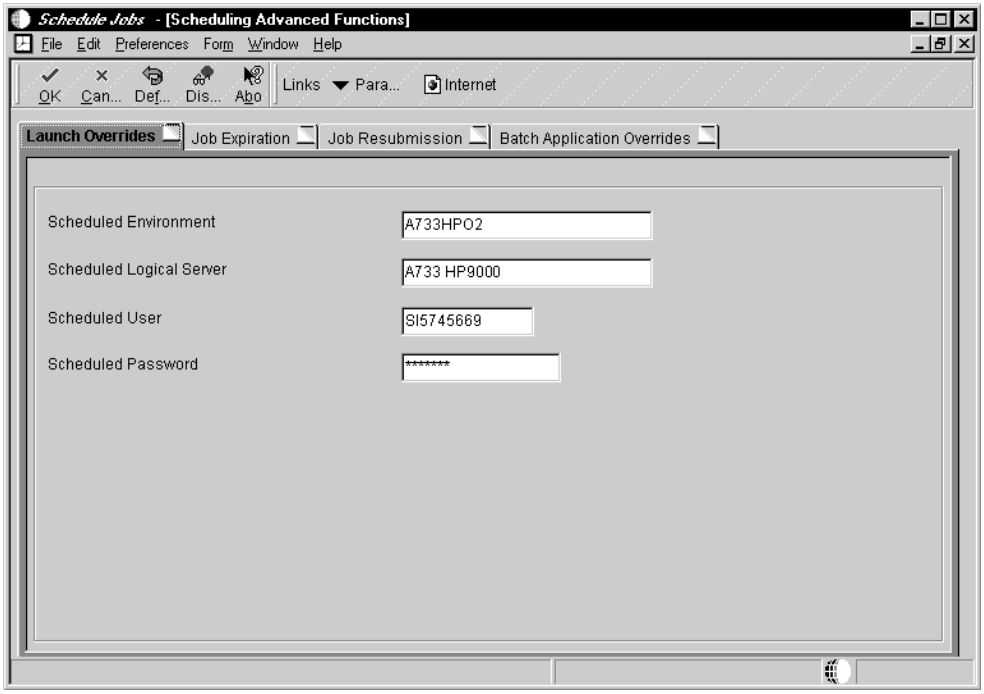
     Type R98825 into this field.
   - Scheduled Version

     For example, XJDE0001.
   - Scheduled Start Date/Time
5. To set the job recurrence (that is, to specify how frequently the job runs) choose Recurrence from the Form menu.



If you do not specify a recurrence by completing the fields on this form, the job runs only one time. J.D. Edwards recommends that for the Push Installation batch program, you set recurrence to run every 30 minutes.

6. On Recurring Scheduling Information Revisions, click OK.
7. On Scheduling Information, to enter any overrides, resubmissions, or expiration options, choose Advanced Options from the Form menu.



8. On Scheduling Advanced Options, click the tab that corresponds to the information that you want to enter or revise:
    - Launch Overrides
    - Job Expiration
    - Job Resubmission
    - Batch Application Overrides
9. Click OK.

After scheduling the job, you can use the Object Configuration Manager (P986110) to verify that the server on which the Push Installation batch program is running points to the same Package Deployment Scheduling (F98825) and Machine Master (F9650) tables that the Package Deployment program uses.

See Also

- *Scheduling Jobs* in the *System Administration Guide* for more information about using the Schedule Jobs program.

Running the Push Package Installation Results Report

From the Package and Deployment Tools menu (GH9083), choose Push Package Installation Results (R98825B).

This report provides the same information that you get when you run the Push Package Installation batch program (R98825).

The report includes the following information:

- Machine key
- Package name and path code
- User class or group
- Package status and status description
- Install status
- Package installation description
- Mandatory install (yes or no)

Push Installation Status Codes

The following table lists the status codes and descriptions that the Push Package Installation program (R98825) uses. Codes marked with an asterisk indicate conditions in which the Push Package Installation program continues to attempt the installation the next time that the Push Package Installation program runs.

Status Code	Description
200*	Scheduled
210*	In Progress
220	Successful Install
230	Install Failed
240*	Install Running
250*	ERP 9.0 Running
260*	Listener Not Started/Installed
270	General Error
280	Already Installed
290	Invalid Package
300	Install Attempted
310	Machine Down

Installing ERP 9.0 Workstations from CD

If your system includes a CD writer, you can build and deploy packages to the CD writer location. After copying the package to a CD, you can then use the CD as a portable deployment tier from which to perform workstation installations. That is, you can run from the CD the setup.exe program that launches the ERP 9.0 Workstation Installation program.

You can set up your enterprise so that you can deploy packages to the CD writer and install ERP 9.0 from a CD.

Defining the CD Writer Location

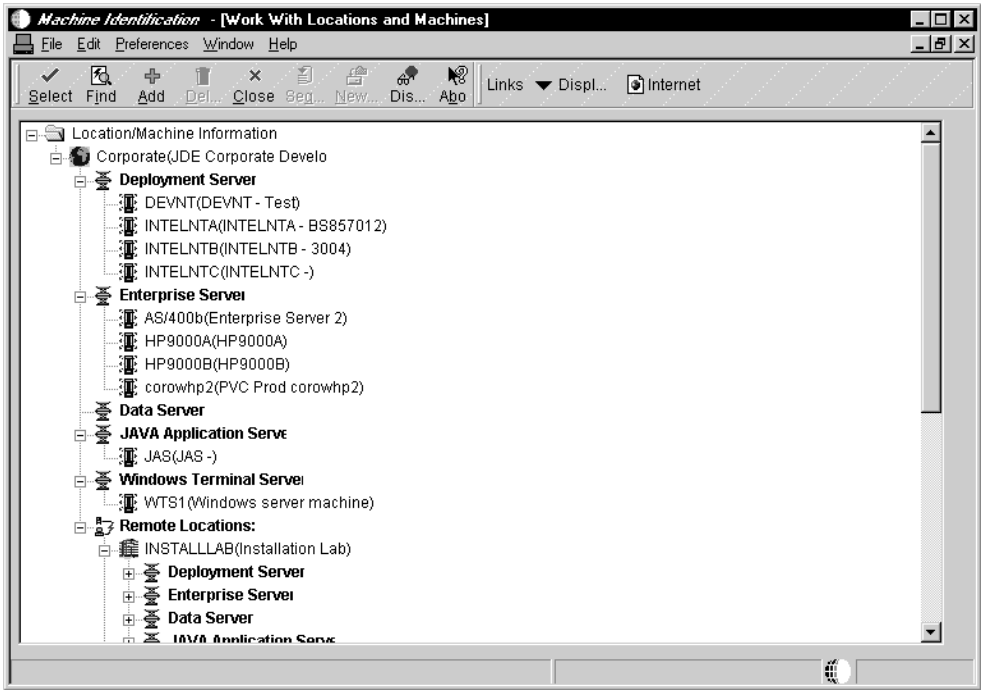
The first step in the process of configuring your system for deployment from CD is to define the CD writer location, if it is not already defined. In this step, you essentially create a pseudo deployment server from which you will later copy package data onto the CD by using the software for your CD writer.

When you define the CD writer location in the Machine Identification application, you must also add the correct pathcodes to the Environments exit.

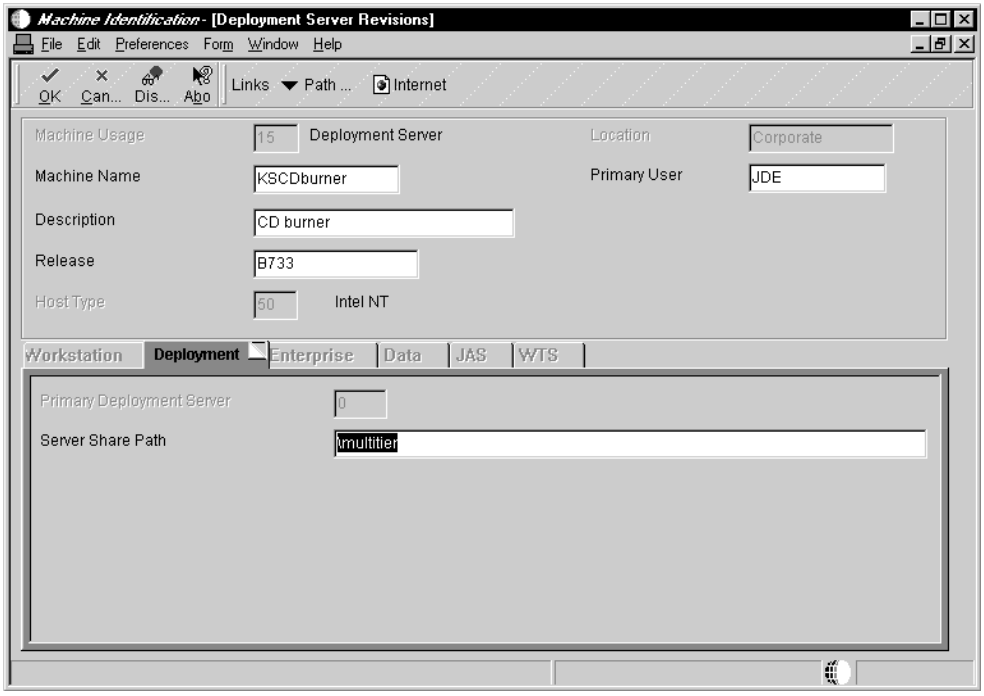
The process for defining this location is identical to the process for defining any other new deployment server.

► To define the CD writer location

From the Package and Deployment Tools menu (GH9083), choose Machine Identification.



1. Subordinate to the appropriate location, choose Deployment server.
2. Click Add to add a new machine.



3. On Deployment Server Revisions, complete the following fields:
   - Machine Name
   - Description
   - Release
   - Primary User
   - Server Share Path
4. If you want to specify a location for data, a foundation, or help files, do so by choosing Data, Foundation, or Helps from the Form menu.

   If you do not specify a location for data, foundation, or helps, the system uses the default locations.
5. Click OK to return to the Work with Locations and Machines form.
6. Click Close to exit from the Work with Locations and Machines form.
7. In Windows Explorer, locate the folder named ERP 9.0 Client Install.
8. Copy this folder by dragging the folder to the CD writer location.

   The location is the server share path that you entered on the Deployment Server Revisions form.

Deploying a Package to the CD Writer Location

After you define the CD writer as a deployment server, you are ready to deploy a package to the CD writer location that you specified. This task involves the following two procedures:

- Deploy to the CD writer location the package that you want to write to the CD.
- Modify the Install.inf and Package.inf files in preparation for writing the package contents to the CD.

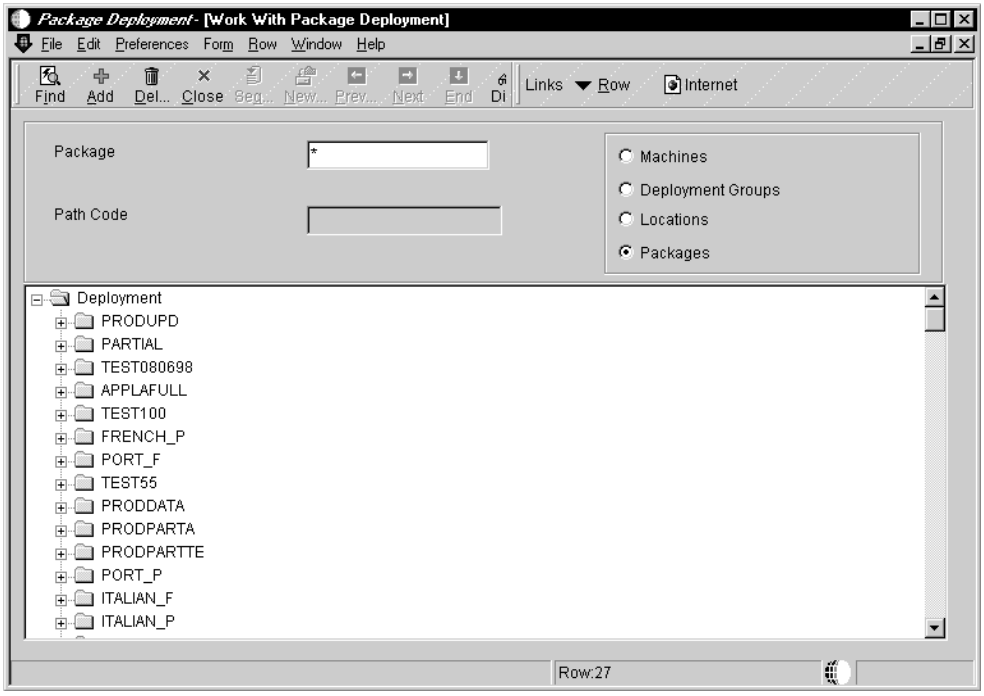
Before You Begin

- ❑ Assemble, define, and build the package that you want to write to the CD

► To deploy the package to the CD writer location

From the Package and Deployment Tools menu (GH9083), choose Package Deployment (P9631).

The Work With Package Deployment form appears.



1. On Work With Package Deployment, click Add. The Package Deployment Director launches.
2. Complete the forms in the Deployment Director in the same way as you would for any other package.
3. From the Work with Package Deployment form, find and choose the package that you just scheduled for deployment, and then choose Deploy from the Row menu to deploy the package.

After you deploy the package to the CD writer location, the directory structure for that location will look similar to the following:

```
Multitier\package_inf\Appl_B.inf
Multitier\systemcomp\system.cab
Multitier\datacomp\data.cab
Multitier\helpscomp\helps.cab
Multitier\Appl_pgf\Package\Appl_B
Multitier\package_inf\Appl_B.inf
```

In the previous example, Multitier is the name of the server share path. Appl_B is the package name.

Note

The server share path name is not included when you copy folders to the CD. In the previous example, the items copied to the CD are \package_inf\Appl_B.inf, \systemcomp\system.cab, and so on.

► To modify the Install.inf and Package.inf files

1. In Windows Explorer, find the CD writer location and open the folder that contains the package that you deployed.

   This folder has the name that you entered in the Server Share Path field on the Deployment Server Revisions form when you defined the CD writer location. In the previous example, the server share path name is Multitier.

2. Open the folder ERP 9.0 Client Installation, and then open the file Install.inf.

   That is, double-click the icon for the file to launch the Microsoft Notepad application.

3. In the section [FileLocations], modify the line so that two periods and a backslash (\) precede the package_inf entry. The line should look like the following after you modify it:

   ```
   [FileLocations]
   PackageInfs=..\package_inf
   ```

   [FileLocations]

   PackageInfs=..\package_inf

4. Similarly, open the Package_inf folder and open the *package name*.inf file.

   In the previous example, this file is named Appl_b.inf.

5. In the section [SrcDirs], modify each of the lines so that two periods and a backslash (\) precede each entry.

   After modification, the [SrcDirs] section should look similar to the following:

   ```
   [SrcDirs]
   SAPPL_PGF=..\APPL_PGF\package\APPL_B
   SSYS=..\systemcomp
   SAPPL_PGFDATA=..\datacomp
   ```

```
SHELP=..\helpscomp
```

Creating the ERP 9.0 Installation CD

After you deploy the package to the CD writer location and modify the Install.inf and Package.inf files, you are ready to copy the package contents to the CD. Use the software that came with your CD writer to accomplish this process, which typically involves copying the package contents to the CD. Refer to the documentation that came with your CD writer for more information about this process.

You copy the package to the CD by copying the subdirectories that are subordinate to the server share path directory. The server share path directory is not created on the CD. (In the previous example, the server share path directory is called Multitier, and it is the same name that you entered in the Server Share Path field on the Deployment Server Revisions form.

When you are finished copying the directories to the CD, the CD should contain the following directories:

- Appl_pgf (contains package information)
- datacomp (contains the database cabinet file)
- helpscomp (contains the helps cabinet file)
- systemcomp (contains the foundation cabinet file)
- package_inf (contains the package.inf file)
- ERP 9.0 Client Install (contains the Workstation Installation program)

Note

Actual names might not be the same as those listed because each system might be different.

Deploying Partial Package Templates

ERP 9.0 users who use laptop computers might find that a full package requires too much disk space. Also, many of these users never need to use all of the ERP 9.0 programs that are included in a full package. To accommodate these users, J.D. Edwards allows ERP 9.0 administrators to create and deploy a partial package template that is specifically designed for laptop computer users. A partial package template is based on the standard ERP 9.0 partial package, and it also contains the objects necessary for laptop computer users to run specific programs.

Because each situation is unique and each company has different needs, the actual components that an administration includes in the partial package template vary. For example purposes, we will create a partial package template for the Sales Force Automation program.

You create and deploy a partial package template using the same tools that you would use to create any other partial package: the Package Assembly Director (P9601), Package Build Director (P9621), and Deployment Director (P9631).

Assembling a Partial Package Template

The first step in creating a partial package template is to use the Package Assembly (P9601) to assemble a partial package. The process is identical to assembling any other partial package. When you assemble the package, be sure to add the object components that users need to run the program after they load the partial package template.

▶ To assemble a partial package template

From the Package and Deployment Tools menu (GH9083), choose Package Assembly (P9601).

1. Click Add to assemble a new package.

   The Package Assembly Director opens.

2. From the Welcome form, click Next.

   The Package Information form appears.

3. Complete the following fields and then click Next:

   - Package Name
   - Description
   - Path Code

   The Package Type Selection form appears.

4. Click the Partial option to indicate that you are assembling a partial package.
5. Click Next.

   The Foundation Component form appears.

6. On the Foundation Component form and the following two forms (the Help Component form and Database Component form), accept the default location by clicking Next, or click Browse to specify another location for the foundation, help file, or database that you want to use.
7. Click Next. The Object Component form appears. This form allows you to specify the individual objects that you want to include in your package.
8. Click Browse to display the Object Component Selection form.

   Use this form to locate and select the objects that you want to include in your package. These are the components that users need to run the program for the package template that you are creating.

   The objects that you specify are in addition to the objects that are automatically included in every partial package. You must include the following objects in your partial package:

   - Interactive and batch applications (APPL and UBE objects)
   - Data structures (DSTR objects)
   - Business views (BSVW objects)

For example, if you are creating a package template for the Sales Force Automation application, you need to include the following system codes using the APPL (interactive application), DSTR (data structure), and BSVW (business view) object types:

- 00 Foundation Environment
- 01 Address Book
- 17 CSM
- 32 Configuration Management
- 40 Inventory/OP Base
- 41 Inventory Management
- 42 Sales Management
- 45 Advanced Pricing

9. When you are finished adding objects, click Close to return to the Object Component form.
10. Click Next to display the Language Component form.

    This form allows you to add to your package language specifications for a language other than English.

11. Click Next to display the Package Component Revisions form.

    This form allows you to review the current foundation, help, and database locations, as well as the objects in the package and any language selection.

12. To change any of these package components, click the icon for the component that you want to change.

    The form for that package component appears.

13. When you are finished assembling the package, click End to exit from the Package Assembly Director.
14. From the Work with Packages form, choose the package that you just created and activate it by choosing Active/Inactive from the Row menu.

See Also

- *Entering a Database Location* for more information about entering database location
- *Entering a Help File Location* for more information about entering a help location
- *Entering a Foundation Location* for more information about entering a foundation

Building a Partial Package Template

After you assemble the partial package, you are ready to define the package build using the Package Build Definition Director.

► To build a partial package template

From the Package and Deployment Tools menu (GH9083), choose Package Assembly.

1. On Work With Packages, choose the partial package that you just assembled, and then choosing Build Director from the Row menu.

   The Package Build Definition Director form appears.

   If the definition of the package is still In Definition, you must change this status to Definition Complete before you can build the package.

   To change the status of the package, choose the package, and then choose Activate from the Row menu.

2. Click Next.

   The Package Build Location form appears. Because you are building a partial package, you cannot build the package for a server.

3. To build a package for workstations, click the Client option, and then click Next.

   The Build Specification Options form appears.

4. Click the Build Specification Options option to turn it on, and then click Next.

   The Business Function Options form appears. If a full package has been built recently, it is not necessary to turn on the Build Functions Options.

5. On Business Function Options, click Next.

   The Compression Options form appears.

6. Click the Compress Options option to turn it on, and then click Next.

   The Package Build Revisions form appears. This form allows you to review the current build options, business function options, and compression options that you specified for the package.

7. To change any of these options, click the tab for the type of option that you want to change.
8. When you are finished reviewing or changing your build options, click End to exit from the Package Build Definition Director.

   The Work With Package Build Definition form appears.

9. Choose the partial package and choose Active/Inactive from the Row menu.
10. When you are ready to initiate the package build, choose Submit Build from the Row menu.
11. Choose one of the following destinations for the build report, and click OK:
    - On Screen

- To Printer

The form automatically closes and ERP 9.0 builds the package. Build time varies, depending on the number and size of the items in your package. When the build is finished, the report either appears on the screen or prints, depending on the destination that you specified.

Deploying a Partial Package Template

After you create the partial package template, you should deploy the package to a workstation and test it before making the package available for all of your laptop users. The procedure for deploying the partial package template is identical to the procedure for deploying any other package.

► To deploy a partial package template

From the Package and Deployment Tools menu (GH9083), choose Package Deployment.

1. Click the Packages option, and then click Find to locate the partial package template.
2. Select the package that you want to deploy, and then choose Active/Inactive from the Row menu to activate the package.
3. Click Add to launch the Package Deployment Director.

   The Package Deployment Director form appears.

4. Click Next.

   The Package Selection form appears.

5. Choose the package that you want to deploy, and then complete the remaining forms in the Package Deployment Director.

   These forms enable you to do the following:

   - Specify the package that you want to deploy
   - Enter attributes such as mandatory installation, push installation, and deployment date and time
   - Indicate the workstations that should receive the package

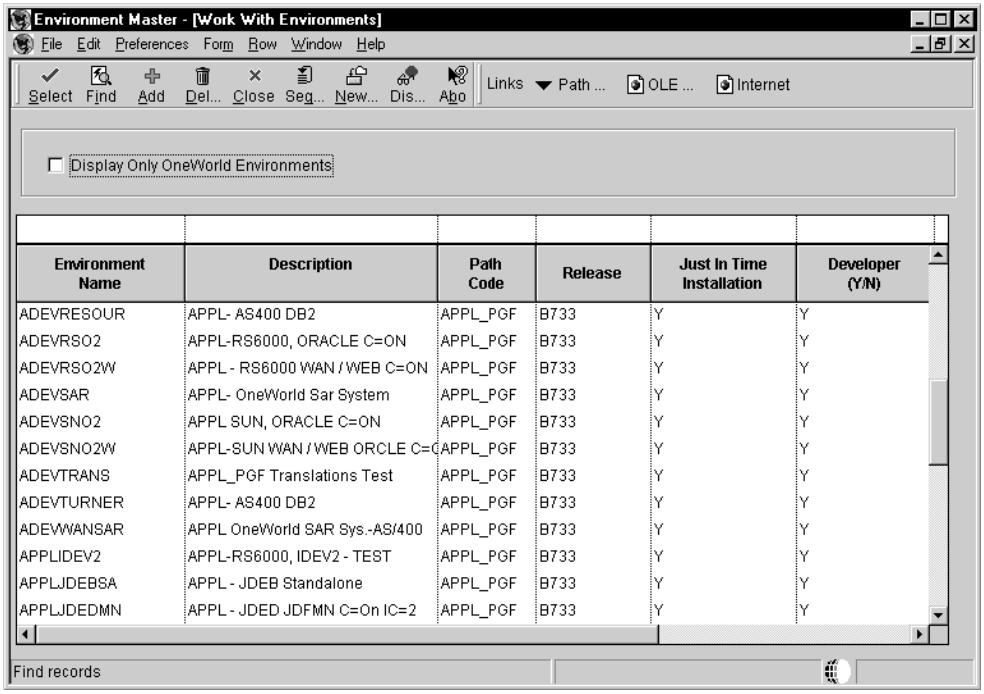
After you schedule your package for deployment, at the specified time on the date that you specified, the package will be deployed to workstations. This package becomes available to users when they sign on to ERP 9.0.

Testing a Partial Package Template

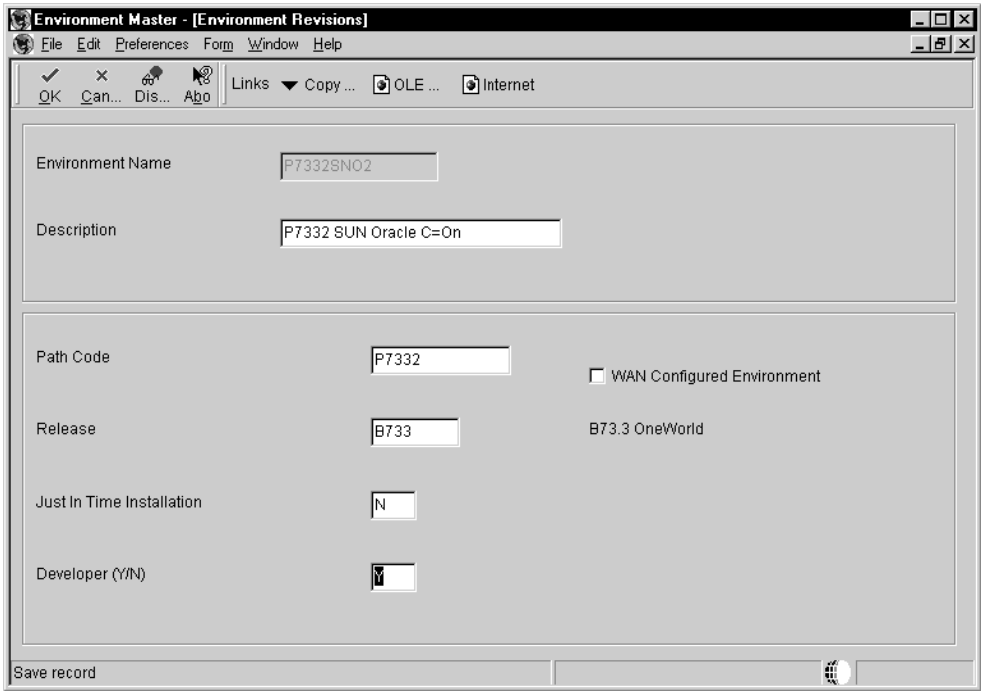
After you install the partial package template on a workstation (laptop), you should test it to see if you can run the program that you included in the package. You should be able to run the program while the laptop is disconnected from the network, and you should not have to install objects on the laptop via just-in-time installation.

► To test a partial package template

1. Ensure that the laptop is connected to the network.
2. Launch ERP 9.0 and sign on.
3. From the Environments menu (GH9053), choose Environment Master.



4. Find and select the environment that you used to create the partial package template.



5. Verify that the Just In Time Installation field is set to N.

   Setting this field to N turns off just-in-time installation. Just-in-time installation should be turned off when the laptop is not connected to the network.

6. Disconnect the laptop from the network and run the application.

You should be able to run the program while the laptop is disconnected from the server. If you did not include in the package an object that the program requires, the system displays an error message informing you that just-in-time installation has been disabled for the environment that you are using.

If you receive this message, you must determine the missing objects and build another package before you can use ERP 9.0 on the laptop while it is disconnected from the server.

Multitier Deployment Terminology

The following table lists and describes multitier deployment terms:

Primary Deployment Location (tier 1) The primary or base deployment location is the location in which the package to be deployed to secondary or remote locations is built. ERP 9.0 requires at least one deployment server for installing and maintaining the software. The server at the primary deployment location should be dedicated solely to deploying and operating ERP 9.0 and should not be used for any other purposes in your company. See the ERP 9.0 Installation Guide for this machine's hardware and software requirements.

Tier Deployment Location (tier 2) Tier deployment locations (also known as remote or secondary locations) have one or more deployment servers that allow you to install ERP 9.0 software on the workstations at that location. These servers receive their packages from the deployment server at the primary or base location. Machines at the tier deployment locations cannot use Object Librarian functions, such as object check-in and check-out. These machines are designed for deployment use only and are not designed to be used for remote development. The number of tiered locations that you can have depends on the network and server capacity.

Tier Workstations Tier workstations are workstations that connect to a tier deployment location for software installation. The number of workstations per deployment location depends on the network and machine load. All tiered workstations must also have a Windows NT domain connection that enables them to connect to, read, and copy files from the shared drives of their respective deployment locations.

See Also

- *ERP 9.0 Installation Guide* for the hardware and software requirements of these machines

Package Build Time Comparison

For comparison purposes, the following table shows package build and compression times for a typical full package, a partial package, a partial package with a limited numbers of objects, and the Sales Force Automation partial package that we used as an example.

Package Type	Package Size	Build Time	Compression Time	Total Build Time
Full	1.25 GB	8 - 9 hours	NA	8 - 9 hours
Partial	137 MB	25 minutes	13 minutes	38 minutes
Partial with 1447 A/R objects	183 MB	30 minutes	15 minutes	45 minutes
Partial with 1307 A/P objects	190 MB	33 minutes	16 minutes	49 minutes
Partial with 2676 SFA objects	293 MB	58 minutes	20 minutes	78 minutes

Multitier Deployment

ERP 9.0 software is normally distributed to workstations from a centralized location. In many cases, you can minimize the performance impact on a single deployment server by using systematic scheduling for software installations. For example, if your site has more than 50 workstations that require a package installation, but you release software only four times a year, you can effectively schedule the installations over a weekend, at night, or during off-peak hours.

While this distribution method is the simplest approach for software deployment, network capacity constrains configurations that have either multiple remote sites or large numbers of users at a single site. For example, software installations to workstations that are connected to the centralized deployment location by a 56KB circuit can take 4 to 6 hours to run.

Multitier deployment provides sites the flexibility of installing packages on workstations and servers from more than one deployment location and more than one deployment server. These additional deployment locations and servers are called deployment tiers. Specifically, instead of installing multiple workstations across a WAN circuit, multitier deployment enables you to transfer a compressed package from the centralized location to the remote workgroup server that acts as a second deployment tier. Hence, multitier deployment means deploying from more than one deployment tier.

For example, you might have one deployment server at your main location, and a second deployment server that serves a remote location. Because the server at the remote location is responsible for deploying to workstations and servers at that location, you do not need to deploy packages from the main deployment server across a WAN, as you would in a single-tier deployment configuration.

Workgroup servers can also be used as second tier deployment locations in a LAN environment where large numbers of workstations need to install software concurrently. J.D. Edwards recommends that you implement multitier deployment if your site has more than 50 workstations receiving ERP 9.0 software installations per day.

Understanding Multitier Deployment

The primary function of multitier deployment is to reduce network traffic (and the delays that result from heavy traffic) by enabling enterprises with more than one geographic location to deploy from a secondary deployment server at the remote site. Instead of installing packages across the WAN from the deployment server to workstations at a remote location, you can copy the package and the package.inf file from the deployment server at the primary location to the deployment server at the remote location. The server at the remote location can then deploy the package to the workstations and servers at that location.

Consider implementing a multitier deployment configuration when either of the following applies to you:

- To many workstations are installing packages from the same location, causing the server and network performance to suffer.
- The workstations are remotely connected to the deployment server over a WAN, which requires unacceptably long installation times.

Normally, you decide whether to implement multitier deployment during CNC implementation. Although you can enable this function at any time, you typically set it up after you have already installed ERP 9.0 and are preparing your production sites to go live.

To set up multitier deployment, you must define the machines (and their associated path codes) that are used for deployment. In addition, you can use a scheduler function to define when software should be pushed to the tiered deployment location.

You must also define individual user characteristics for multitier deployment. Normally, you do this by modifying the user profiles to indicate the deployment location from which a user pulls a package.

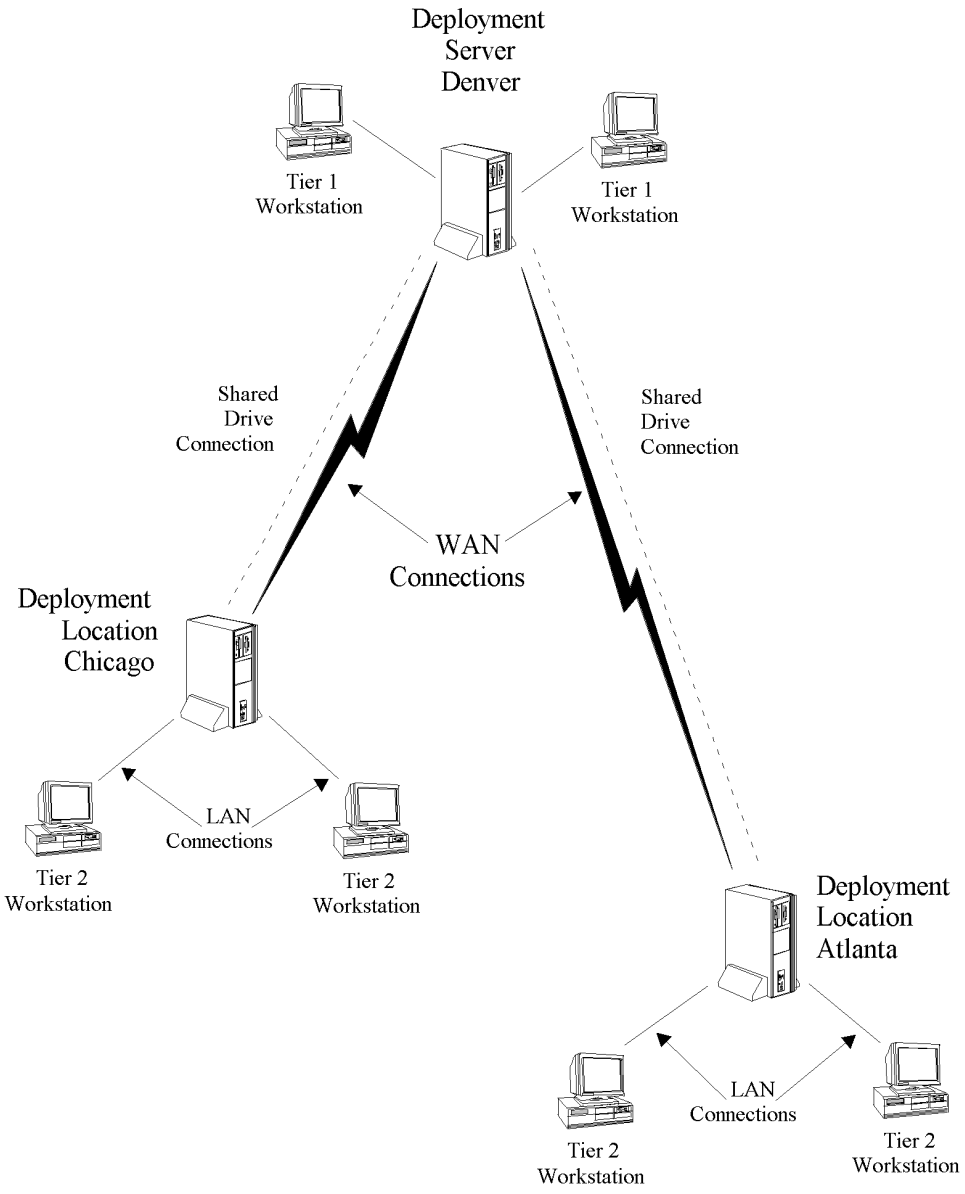
Multitier Deployment Features

Multitier deployment offers the following features:

- You can deploy workstations from any number of deployment servers.
- You can easily add deployment locations, and the deployment machine does not need to be a server; it could be a Windows 95 or Windows NT workstation.
- Setup and administration are straightforward tasks.
- You maintain central control over files and data that is transferred to remote deployment locations.
- You can easily schedule software for deployment to remote sites.
- Multitier deployment is integrated into the Package Deployment Director, so the process for deploying is essentially the same as the process for deploying in a single-tiered setup.

Example: Two-Tier Deployment Strategy

The following illustrates a typical two-tier deployment strategy:


Deployment
Server
Denver
Tier 1
Workstation
Tier 1
Workstation
Shared
Drive
Connection
Shared
Drive
Connection
WAN
Connections
Deployment
Location
Chicago
LAN
Connections
Tier 2
Workstation
Tier 2
Workstation
Deployment
Location
Atlanta
LAN
Connections
Tier 2
Workstation
Tier 2
Workstation

Multitier Implementation

Packages are always built on the deployment server at the primary location. After you build the package that you want to deploy to remote locations, you must follow these steps to implement multitier deployment.

14. Define deployment locations

    You must define each physical location to which you want to deploy. For example, if your main office is in Denver and you want to deploy to your branch office in Seattle, you must define the Seattle deployment location

15. Create Deployment Server Definitions

    You must also define the deployment server at each remote location.

    Note
    This step is not necessary if you used the Remote Location Workbench to create deployment server definitions when you installed ERP 9.0.

16. Schedule the Package

    The next step in the process is to schedule the package for deployment using the Package Deployment program (P9631). The process of scheduling a package for multitier deployment is identical to the process for scheduling any other package.

17. Deploy the Package to Workstations

    After you deploy the package to the deployment server at the remote location, that server can deploy to the workstations at that location.

Before You Begin

- Ensure that you have a thorough understanding of the CNC concepts that will allow you to define and implement packages, workstation installations, path codes, central objects, replicated objects, and user profiles.
- Ensure that adequate disk space exists on the disk drive for the machine that you will be using as a tier deployment location. Also, remember that each full package that you deploy adds approximately 1.4GB, and each additional partial package adds approximately 165MB. The amount of required disk space varies, depending on the amount of data that you replicate to a J.D. Edwards Supported Local Database.

See Also

- *Defining Locations* in the *Package Management Guide* for information to set up a location
- *Defining Deployment Servers* in the *Package Management Guide* for information to set up a deployment server
- *Distributing Software to Deployment Locations* in the *Package Management Guide* for information to deploy from remote servers to workstations
- *ERP 9.0 Disk Space Requirements* in the *ERP 9.0 Installation Guide* for more information about minimum disk space requirements for a deployment server
- *Running Installation Workbench* in the *ERP 9.0 Installation Guide* for more information about the Remote Location Workbench

Defining Deployment Servers

The Machine Identification application (P9654A) enables you to either add a new deployment server definition or modify an existing definition. When you add a new deployment server definition, ERP 9.0 creates a record in the Deployment Locations Definition table (F9654) for each deployment location. Each server at each deployment location must be defined.

Typically, the table contains one record for the primary deployment server and one record per deployment location for each release. If you have multiple releases of ERP 9.0, you must create multiple records for the servers at each deployment location.

If you used the Remote Location Workbench to create deployment server definitions when you installed ERP 9.0, you do not need to define deployment servers again.

See Also

- *Running the Installation Workbench* in the *ERP 9.0 Installation Guide* for more information about the Remote Location Workbench

Defining a New Deployment Server

The following task describes how to define a new deployment server. You need to do this, for example, if you open a branch office in another city and need to define a secondary deployment server that can handle deployment to the workstations at that location.

Before You Begin

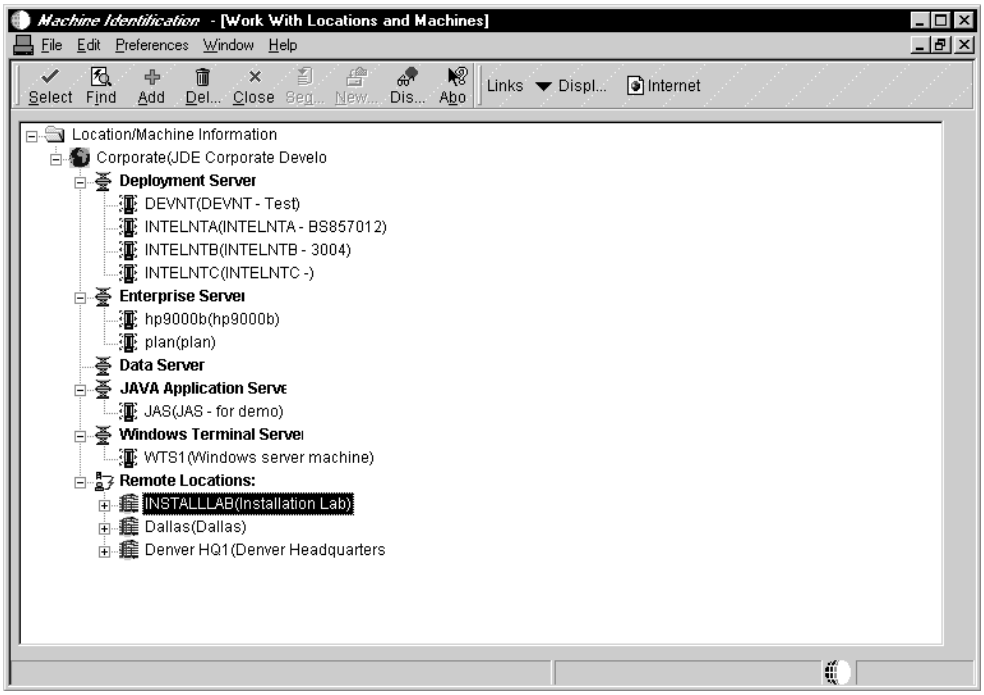
- If you are adding a new definition for a deployment server at a remote location, you first need to define that location. For information about defining locations, see *Defining Locations.*

Note

J.D. Edwards recommends that you do not define deployment locations with a DEV path code because the DEV path code is normally associated with developers who are not located at remote locations.

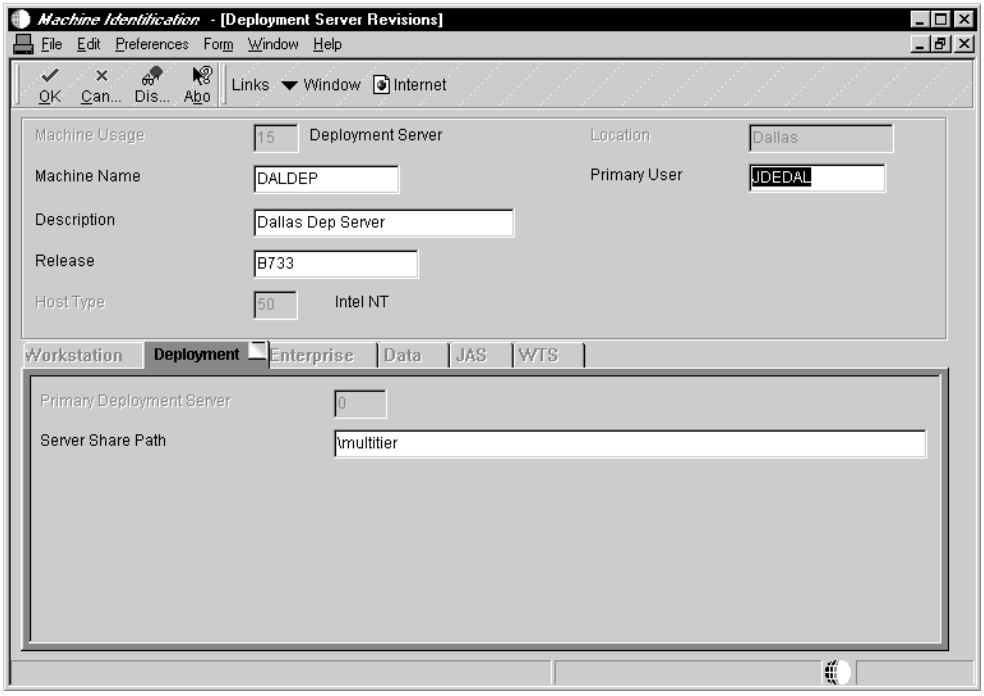
► To add a new deployment server definition

From the Package and Deployment Tools menu (GH9083), choose Machine Identification (P9654A).



1. Find the location where the deployment server resides, and expand the tree, if necessary, to display the different machine types.
2. Choose the Deployment Server heading and then click Add.

   The Deployment Server Revisions form appears.



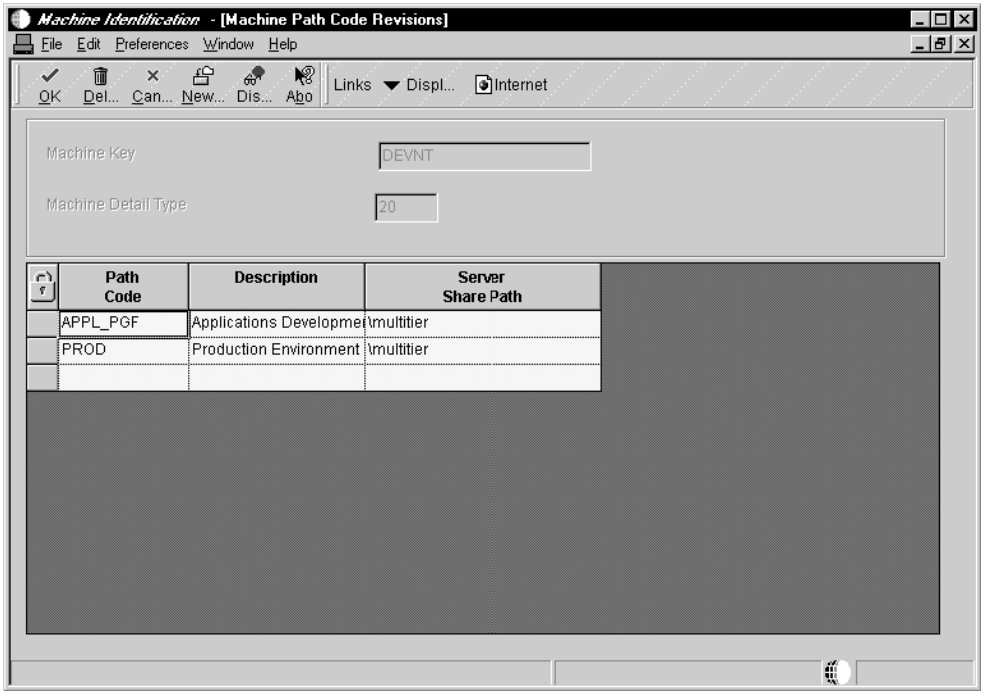
3. On the Deployment Server Revisions form, complete the following fields:

   - Machine Name
   - Description
   - Release
   - Primary User
   - Server Share Path

   Because you are entering a definition for a server other than your primary deployment server, the field Primary Deployment Server is not available. You can enter this field only if you first delete the definition for your current primary deployment server.

4. Choose Path Code from the Form menu.

   The Machine Path Code Revisions form appears.



5. On the Machine Path Code Revisions form, enter the path code for the primary or base location from which the secondary location will receive the package. When you enter the path code, the server share path will default to the base server share path for that machine.

   Important
   You must specify the path code for each deployment server at all secondary locations. If you do not indicate the path code, multitier deployment may not work.

6. If your package includes a user-defined foundation, data, or help files, specify those items by choosing Foundation, Data, or Helps from the Form menu.
7. On the Deployment Server Revisions form, click OK to return to the Work With Locations and Machines form.

Revising an Existing Deployment Server Definition

In many situations, you might need to modify the definition for a deployment server that you already defined. For example, you would need to change the definition if the server share path or ERP 9.0 release changes, or if you want to designate a different server as your primary deployment server.

The process for revising an existing deployment server definition is very similar to the process for adding a new definition.

► To revise an existing deployment server definition

From the Package and Deployment Tools menu (GH9083), choose Machine Identification (P9654A).

1. Find the deployment server for which you want to modify the definition, and then click Select.

   The Deployment Server Revisions form appears.

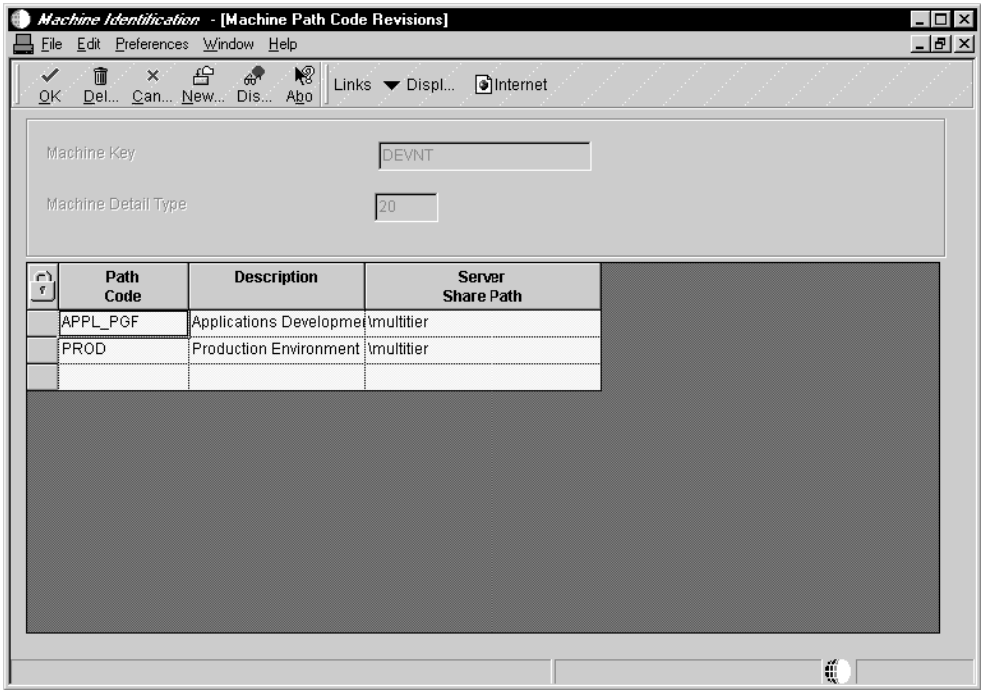
2. On Deployment Server Revisions, you can modify the following fields:

   - Description
   - Release
   - Primary User
   - Server Share Path

   If you are modifying the primary deployment server, you can also change the primary deployment server. For any other server, you cannot change the value in this field. A 1 in this field indicates that the deployment server is the primary deployment server. You can have only one primary deployment server.

3. Choose Path Code from the Form menu.

   The Machine Path Code Revisions form appears.



4. On the Machine Path Code Revisions form, enter the path code for the primary or base location from which the secondary location receives the package. When you enter the path code, the system displays the default server share path, which is the base server share path for that machine.

Important
You must specify the path code for each deployment server at all secondary locations. If you do not indicate the path code, multitier deployment might not work.

5. If your package includes a user-defined foundation, data, or help files, specify those items by choosing Foundation, Data, or Helps from the Form menu.
6. On the Deployment Server Revisions form, click OK.

See Also

- *Defining Machines, Locations, and Deployment Groups* in the *Package Management Guide* for more information about defining machines, locations, and deployment groups

Scheduling Packages for Deployment Locations

You can define which packages your primary deployment server distributes to your deployment locations. You can also define when and where the packages are deployed.

Workstations that install from tier deployment locations access the tier deployment server for the help files unless you choose to deploy the help files to the workstation.

After you create or modify a deployment location, you can schedule packages for deployment by using the Package Deployment program (P9631). The process for scheduling packages to a secondary deployment server is identical to the process for scheduling a package to a primary server. Be sure to specify the secondary server location when you schedule the package.

Note

When you create a distribution schedule, remember that you cannot schedule multiple packages to deploy at the same time.

Before You Begin

- Create or modify an existing deployment location and deployment server definition. See *Defining Deployment Servers* and *Defining Machines* for more information.

Distributing Software to Deployment Locations

You use the Package Deployment program (P9631) or the Multitier Deployment batch program (R98825C) to distribute software to deployment locations. Use the Package Deployment program to define the scheduling parameters or to deploy the package immediately. Otherwise, use a version of the Multitier Deployment batch application to distribute the software to deployment locations.

Whether you push or pull the software depends on the machine on which you run the deployment function or batch program for the scheduling program. You push the software if you run the program or report on the primary deployment server or from a workstation.

Conversely, you pull the software if you run the program from a workstation at the tier deployment location. In either case, execute the application on the primary deployment server machine for push installation, or the destination deployment location for pull installation.

Caution

If you push the software, you must have full read and write privileges for both deployment servers (that is, the primary deployment server and the tier deployment location or destination machine), even if you do not run the batch application. If you do not have read and write authority on both servers, the deployment will fail.

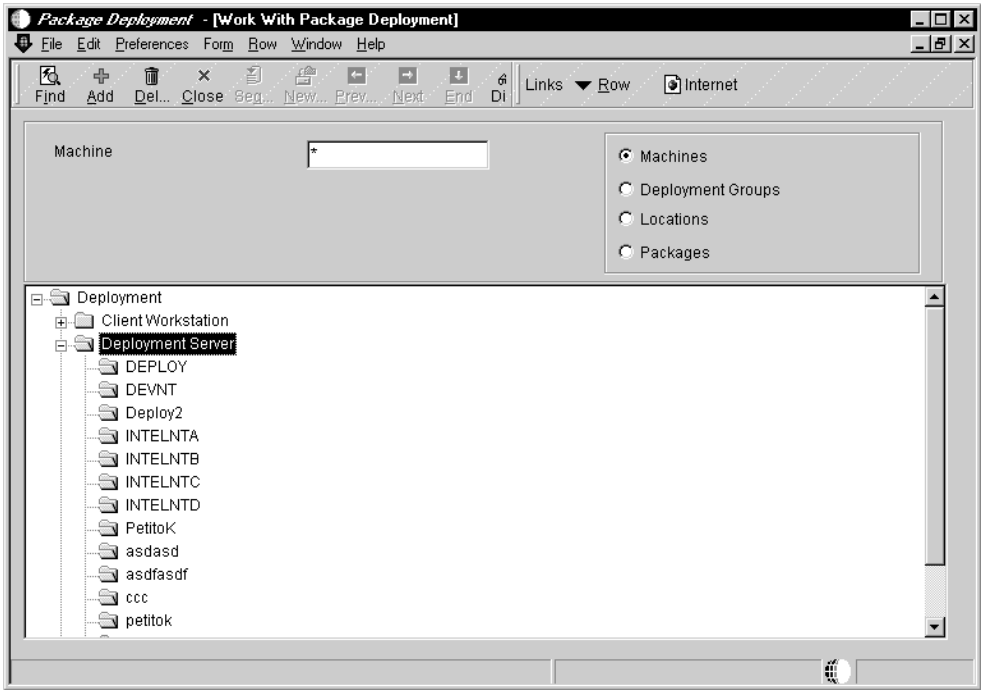
After the package software is automatically distributed through the Package Deployment program or the Multitier Deployment batch program, you must manually copy the workstation installation programs from the primary deployment server to the tier deployment location. These programs are located in the client portion of the base installation directory.

Distributing Software through Package Deployment

Use this method when you want to distribute software immediately after you define a deployment schedule. If you choose this option, the software will be distributed immediately, regardless of the timing parameters that you specify in the scheduling fields.

► To distribute software through Package Deployment

From the Package and Deployment Tools menu (GH9083), choose Package Deployment.



1. On Work with Package Deployment, click the Machines option, and then click Find.
2. Choose the deployment server to which you want to deploy.

If you have scheduled to deploy packages to more than one deployment server, you can choose the Deployment Server folder to deploy to all applicable servers, rather than deploying to each individual server.

3. Choose the deployment server name to deploy all packages that are listed under the server name.

   Alternatively, choose only the package that you want to deploy.

4. From the Row menu, choose Deploy.

   This option launches the Multitier Deployment batch program (R98825C), which enables you to deploy to deployment servers.

   You can also use this Deploy option to launch a batch application that deploys enterprise server packages. Therefore, if you choose an enterprise server name or the Enterprise Server folder on the Work With Package Deployment form, the Enterprise Server Deployment batch program (R98825D) launches, not the Multitier Deployment batch program. When you choose a workstation, the Deploy option is not available.

Deploying a Server Package with Multitier Deployment

After you create the server package that you want to deploy using multitier deployment, you can use the Package Deployment program (P9631) to schedule that package to a server at the same location or a remote location. Verify that the server package is compressed because, unlike client packages, you cannot deploy the server package using multitier deployment unless it is compressed.

Also, before you can deploy a server package with server multitier deployment, the package status must be either 50 (Build Completed Successfully) or 70 (Build Completed with Errors). If the package does not have a status of 50 or 70, it will be unavailable when you schedule the deployment.

The process for scheduling a server package for multitier deployment is essentially the same as the process for scheduling packages within the same location.

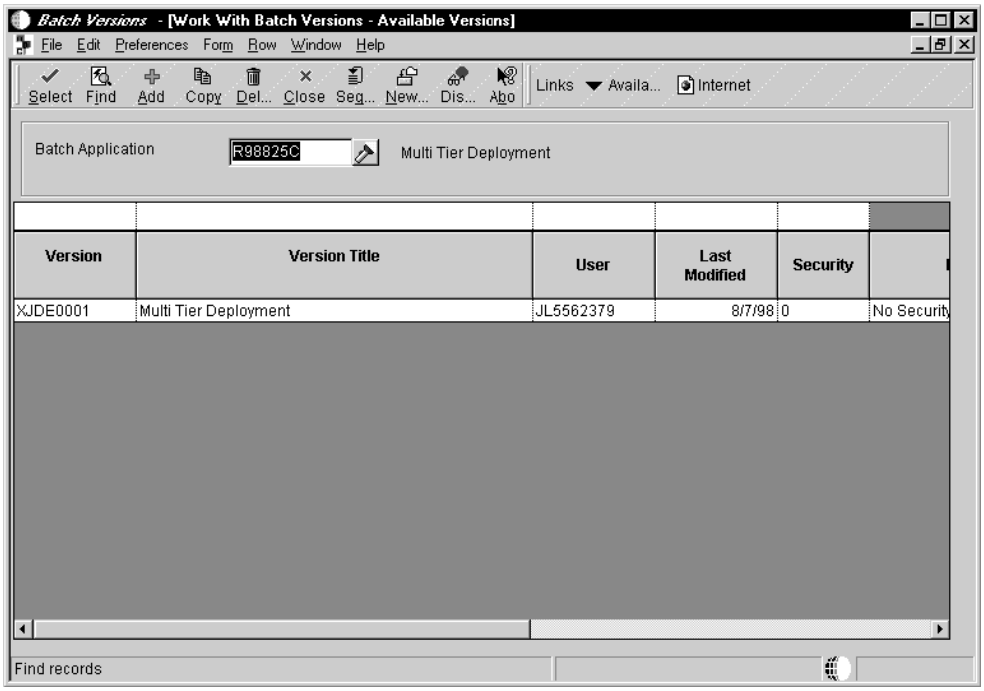
Distributing Software Through the Multitier Deployment Batch Process

Use this method if you want to distribute software that has been previously defined and scheduled through the Package Deployment program (P9631). Running the batch program deploys the software if the scheduled date and time have already passed.

For example, if you schedule a report to run at 1300 hours on June 1, 2005, running the report at 1300 hours on May 31, 2005 has no effect. In order to distribute the software in this example, you must run the batch process at some time after 1300 hours on June 1, 2005.

► To distribute software through the Multitier Deployment batch application

From the System Administration Tools menu (GH9011), choose Batch Versions.



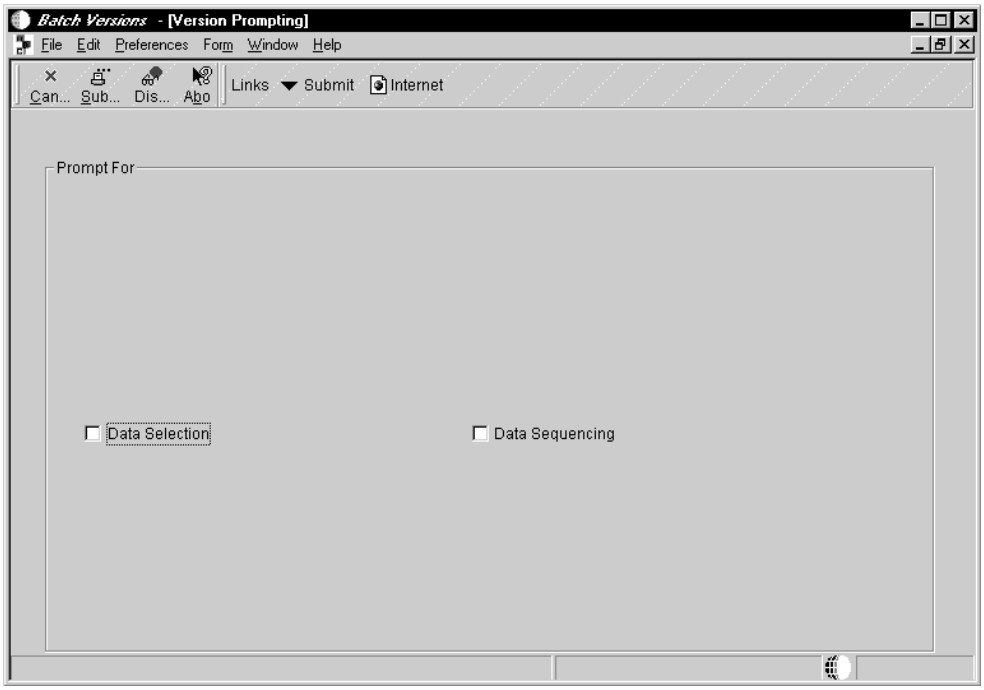
1. Complete the following field:

   - Batch Application

     Enter R98825C this field.

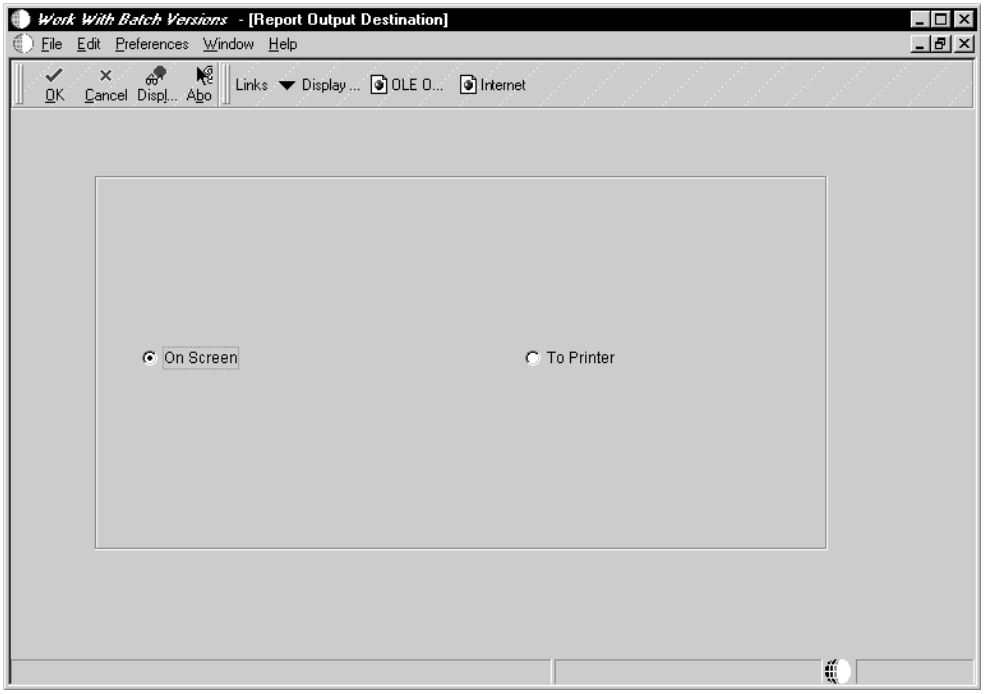
2. Find and choose the Multitier Deployment version that you want to use.

   The Version Prompting form appears.



3. Click Submit.

   The Report Output Destination form appears.



4. On Report Output Destinations, click one of the following options:
   - On Screen
   - Printer

The output is directed to the Adobe Acrobat Reader. This output should indicate that the report completed successfully.

Copying Workstation Installation Programs to Deployment Locations

Regardless of which method you use to distribute the package, you must manually copy the Client Workstation Installation programs from the primary deployment server to the tier 2 deployment locations. You must also change the package.inf file to reflect the machine name and environment of the remote deployment location.

These steps ensure that ERP 9.0 runs the Client Workstation Installation program locally at the deployment location. If you do not complete these two steps, ERP 9.0 searches the base location for the Client Workstation Installation program and its associated files, and copies the files across the WAN to the deployment location.

► **To copy workstation installation programs to deployment locations**

1. Connect to each deployment location and use Windows Explorer to drag the following client directory to the tier 2 workstation:

   ```
   \\Tier1DeploymentServerName\B9\OneWorld Client Install
   ```

2. Open the package.inf file and locate the [FileLocations] section. Change the PackageInfs line to reflect the machine name of the deployment server and the environment at the deployment location. For example:

   ```
   PackageInfs=\\machine
   name\environment\ENVIRON\Evapps\appl_pgf\package_inf
   ```

3. Save your changes by choosing Save from the File menu.
4. Choose Exit from the File menu.

Installing Workstation Packages from Deployment Locations

The process for installing packages to workstations from deployment locations is very similar to the process for a normal package installation. Basically, it consists of running the setup.exe program.

In the case of multitier deployment, this program resides on the tier deployment location in the client subdirectory that is subordinate to the base installation directory. The workstation that is attached to the deployment location must have read access to this directory on the deployment location.

Multitier Deployment for Server Packages

Server multitier deployment lets you automatically deploy a server package from one deployment server to another. The target deployment server to which you are deploying the package can be either in the same location as the source deployment server that sends the package, or in one or more remote locations.

You typically build packages on the primary deployment server at the base location, but using a different server for installations from the base location might have some advantages. In

some situations, you might prefer to deploy a server package to a server at a remote location, rather than require remote users to access the server package over a WAN.

Server package multitier deployment allows users to choose which package builds they want to deploy. For example, if you are building a Prod package for multiple server platforms, you can select the build for the platform that has been successfully built. The benefit of having the ability to choose builds is that users are no longer required to wait to install a new client package.

You use the Package Deployment program (P98631) for server multitier deployment. When you deploy a server package, the system copies the following components:

- Foundation, data, and helps
- Subdirectories that are subordinate to the package name directory, including the following:
  - bin32
  - include
  - lib32
  - obj
  - source
  - spec
- Subdirectories that are subordinate to the server build directories, including the following:
  - bin32
  - spec
  - obj
  - lib32
- The ServerPackage.inf file in the server build directories, including the following:
- The package.inf file
- The pack directory that is subordinate to the package name directory

When server builds are deployed only without client builds, none of the subdirectories that are subordinate to the package name directory are copied, except for the pack subdirectory.

The following table shows where major package components are copied from and to during the deployment process. The server share path is derived from the Machine Detail table (F9651).

Package Component:	**Copied From:**	**Copied To:**
Package	Copied from the path indicated in the SrcDirs section of the package.inf file.	Copied to <*server share path*>\<*path code*>\<*package name*> on the destination deployment server.

Package Component:	Copied From:	Copied To:
Foundation, data, and helps	Copied from the path indicated in the SrcDirs section of the package.inf file.	Copied to *<server share path>\systemcomp* on the destination deployment server, if the default location is selected.
Package.inf file	If the source deployment server is the primary deployment server in the base location, this file is copied from the path indicated in the Object Path Master File table (F00942). Otherwise, this file is copied from the path to the package.inf file in the Machine Detail table (F9651).	Copied to *<server share path>\package_inf* on the destination deployment server.

Smart Deployment

Server multitier deployment incorporates the smart deployment feature, which makes available only the server packages that match the available servers for the package destination. For example, if server packages exist at the base location for the HP9000 and the AS/400, but the destination location has only an HP9000, only the HP9000 package is made available for deployment to that location. In other words, you cannot deploy a server package unless the package destination supports that server platform.

Even if you have multiple locations, smart deployment ensures that only the server packages that match the platform of the destination are available.

Automatic Package.inf File Updating

When you deploy a server package to a remote location, the following sections of the package.inf file are updated:

- SrcDirs
- Attributes
- DeploymentServerName
- Location

The package.inf file is not copied when you deploy to a deployment server in the base location.

Before You Begin

- ❑ Assemble and build the server package as described in *Package Build*.

► To schedule a server package for multitier deployment

From the Package and Deployment Tools menu (GH9083), choose Package Deployment (P9631).

1. Click Add to open the Package Deployment Director.

   Unless specified otherwise, after the Package Deployment Director opens, click Next to proceed to the next form.

2. Select the server package that you want to deploy.
3. On Package Deployment Targets, click the Deployment Server option.
4. On Package Deployment Attributes, enter the deployment date and time, and turn on any deployment options.
5. On Deployment Server Selection, enter the machine name to which you want to deploy the server package.

   Select the machine by double-clicking in the row header.
6. On Build Selection, select the build (or version) of the package that you want to deploy by double-clicking in the row header.

   Because of the Smart Deployment feature, the system allows you to select only the package builds that match the configuration at the destination location. For example, if package builds exist for an AS/400 and an HP 9000, but the destination location has only an HP9000, you cannot select the AS/400 build.
7. Click Close to exit the Build Selection form.
8. On Work With Package Deployment, click End to finish the server package scheduling process.

See Also

- *Distributing Software to Deployment Locations* in the *Package Management Guide* for information about deploying the package either through the Package Deployment program (P9631) or through the Multi Tier Deployment batch program (R98825C).

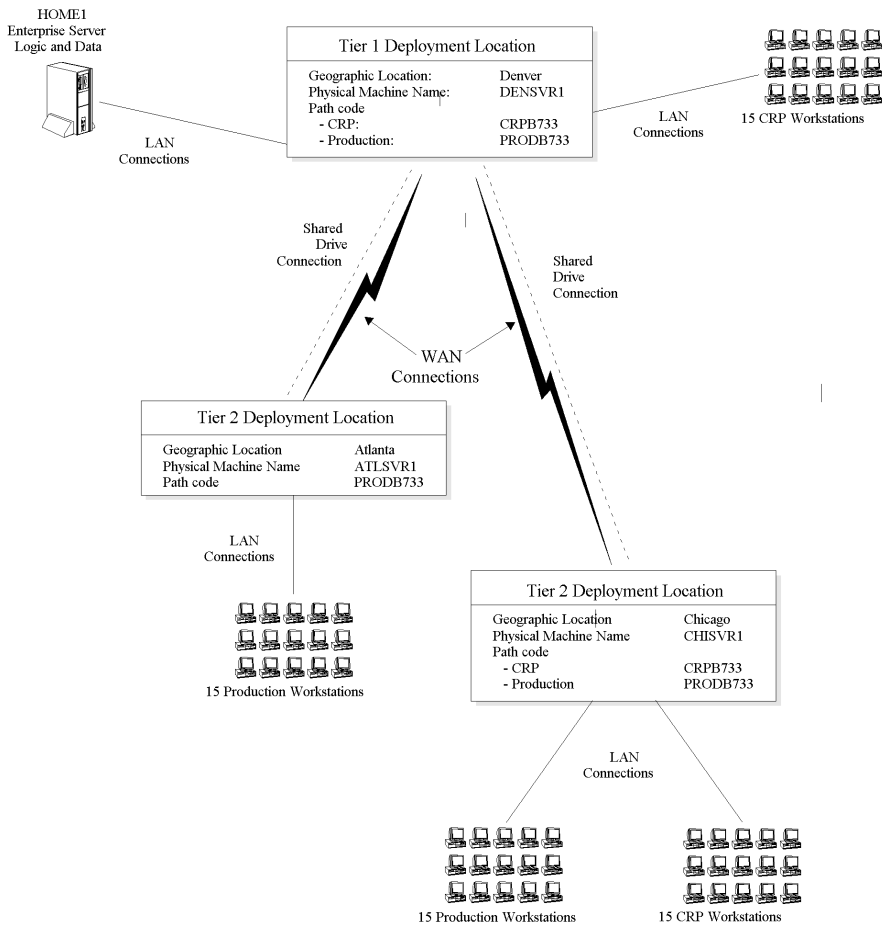
Understanding a Multitier Deployment Case Study

The following is a case study for a two-tier deployment environment. As this case study shows, deploying ERP 9.0 packages to workstations using WAN connections is generally not efficient. Instead, you should deploy from a primary deployment server to tier deployment locations. After that, you can install packages to LAN-attached workstations from each local deployment location.

For ERP 9.0 package installations, a remote deployment location functions as a file server. You cannot build packages at a remote deployment location; packages must be built at the primary deployment location.

While locally-attached workstations can pull packages from the tier deployment location, these workstations still require ERP 9.0 enterprise server and database server connectivity.

The following diagram presents an illustration of this case study.



The following table shows the assumptions used by the Tier 1 location in this case study.

Denver (home office, Tier 1 deployment location)

Characteristic	Setting	
Enterprise server name	HOME1	
Deployment server name	Denver	DENSVR1
	Atlanta	ATLSVR1
	Chicago	CHISVR1
Prototype workstations	Denver	15
	Atlanta	0
	Chicago	15
Production workstations	Denver	0
	Atlanta	15
	Chicago	15
ERP 9.0 release	B9	
Deployment tier	Denver	1
	Atlanta	2
	Chicago	2
Path codes	Denver	PD9
		PY9
	Atlanta	PD9
	Chicago	PD9
		PY9

► To configure your system for multitier deployment

The following summarizes the steps necessary to configure your system for multitier deployment.

1. Define the Deployment Locations

   Define the deployment locations on the deployment server (DENSVR1 in this example). Use the Machine Identification application (P9654A) to define all deployment locations.

   For this case study, complete the following fields to define three locations, one for each deployment location: Denver, Atlanta, and Chicago.

   - Location

     Enter the name of the deployment location.

     In this case study, you assign the following names for each physical deployment location:

Denver

Atlanta

Chicago

- Description

  Enter a description (any value up to 30 characters) for each deployment location.

  For example:

  Denver: Denver - Tier 1

  Atlanta: Atlanta - Tier 2

  Chicago: Chicago - Tier 2

- Location Code

  Enter the current location for ERP 9.0 deployment.

  For example, DEN.

- Parent Location

  Enter the name of the parent location for the location that you are adding.

  For example, Corporate.

2. Create Deployment Server Definitions.

   Use the Machine Identification program (P9654A) to create a definition for each deployment server at the deployment locations that you created. For this case study, you need to define a deployment server for Atlanta and Chicago. The deployment server in Denver is already defined because it is the primary (tier 1) server.

   For this case study, complete the fields on the Deployment Server Revisions form as shown in the following table:

Field	Value
Machine Name	Enter the name of the physical machine. In this case study, enter the following values: Denver DENSVR1 Atlanta ATLSVR1 Chicago CHISVR1
Description	Enter a description (any value up to 30 characters). For example, Multitier Deployment - Denver.
Release	Enter the ERP 9.0 release. For example, B9.
Primary User	Enter the primary user for the machine that you entered.
Server Share Path	Enter the name of the shared directory for the path code in which system files and other files reside. For example, \B9.

3. Schedule the Package.

   Schedule the package to be deployed from the tier 1 deployment server to the tier 2 deployment servers through the Package Deployment program (P9631) or the Multi Tier Deployment batch program (R98825C).

See Also

- See *Defining Deployment Servers* in the *Package Management Guide to set up a Deployment Server*
- See *Distributing Software to Deployment Locations* in the *Package Management Guide* to schedule the package for deployment
- See *Defining Locations* in the *Package Management Guide* to set up a deployment location