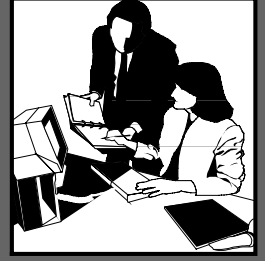


May 2001



J D E D W A R D S ®

OneWorld Package Management



J.D. Edwards World Source Company
One Technology Way
Denver, CO 80237

Copyright © J.D. Edwards World Source Company, 2001

All Rights Reserved

J.D. Edwards is a registered trademark of J.D. Edwards & Company. The names of all other products and services of J.D. Edwards used herein are trademarks or registered trademarks of J.D. Edwards World Source Company.

All other product names used are trademarks or registered trademarks of their respective owners.

The information in this guide is confidential and a proprietary trade secret of J.D. Edwards World Source Company. It may not be copied, distributed, or disclosed without prior written permission. This guide is subject to change without notice and does not represent a commitment on the part of J.D. Edwards & Company and/or its subsidiaries. The software described in this guide is furnished under a license agreement and may be used or copied only in accordance with the terms of the agreement. J.D. Edwards World Source Company uses automatic software disabling routines to monitor the license agreement. For more details about these routines, please refer to the technical product documentation.

Table of Contents

Overview of Package Management	1
Understanding OneWorld Roles.....	3
CNC Consultant and CNC Administrator	3
Custom Solution Consultant and Application Developers	3
Application Consultants and Application Project Leaders.....	4
Hardware, Network, and Third-Party Software Consultants and Administrators	4
Understanding the Package Management Guide.....	5
Understanding Packages.....	7
Understanding the Deployment Programs.....	8
Understanding Workstation Installation	8
Understanding Package Deployment	8
Understanding Server Packages.....	9
Understanding Multitier Deployment	9
Package Management Planning and Setup	10
Working with the Recommended Path Codes.....	12
Suggested Package Names.....	13
Ensuring the Integrity of the Production Environment.....	14
Disabling Just-In-Time Installation	14
Understanding the Development Processes.....	16
Working with the Normal Development Process	16
Developing Short-Term Fixes	19
Comparing the Deployment Methods	21
Understanding Cumulative and Noncumulative Update Packages.....	21
Recommendations for Sites Using Partial Packages With JITI	21
Recommendations for Sites Using Full Packages Without JITI.....	22
Recommendations for Developers	22
Multitiered Deployment and Just-in-Time Installation.....	22
Comparing the Deployment Methods.....	23
Deploying Various Types of Modifications.....	24
Deploying Data.....	28
Deploying a New Replicated Local Database	28
Deploying a Special Database for Store-and-Forward Users.....	30

OneWorld Modification Rules.....	31
What an Upgrade Preserves and Replaces.....	33
General Rules for Modification.....	34
Interactive Applications.....	34
Reports.....	36
Application Text Changes.....	38
Table Specifications.....	38
Control Tables.....	39
Business Views.....	40
Event Rules.....	41
Data Structures.....	42
Business Functions.....	43
Versions.....	44
Object Management.....	45
Understanding Object Movement.....	46
Table (Object Type TBLE).....	47
Business View (Object Type BSVW).....	47
C Business Function (Object Type BSFN, Source Language C).....	48
Business Function Data Structure (Object Type DSTR).....	48
Embedded Event Rules.....	49
Business Function Event Rule (Type BSFN, Source Language NER).....	49
Media Object Data Structure (Object Type GT).....	50
Interactive Application (Object Type APPL).....	51
Batch Application (Object Type UBE).....	52
Data Dictionary Items.....	52
Understanding Files Created By the Build Process.....	54
Workstation Build.....	54
UNIX Server Build.....	55
Windows NT Server Build.....	57
AS/400 Server Build.....	58
Correlating Replicated and Central Objects.....	61
Performing Backups and Restoring Objects.....	63
Copying Records Between Data Sources.....	65
Package Build.....	68
Understanding Packages and Deployment.....	69
Understanding Full and Partial Workstation Configurations.....	70
Understanding Just-in-Time Installation.....	72
Understanding Package Types.....	73
Understanding Features.....	76
Overview of Creating and Deploying a Package.....	76
Understanding Server Packages.....	80
Considerations for Users of Sun Operating Systems.....	86

Assembling Packages	88
Understanding the Package Assembly Director.....	88
Assembling a Package	91
Activating an Assembled Package.....	115
Revising an Existing Package.....	116
Copying a Package.....	117
Deleting a Package.....	118
Defining Package Builds.....	119
Understanding the Build Process	120
Understanding the Package Build Definition Director.....	121
Defining a Package Build.....	123
Revising Build Options for a Package	137
Compressing the Parent Package (After Building the Update).....	138
Copying a Built Package	138
Viewing Package Build History and Resubmitting Builds	141
Incorporating Features Into Packages.....	142
Feature Build and Deployment Process Overview.....	143
Understanding the Feature Based Deployment Director	144
Creating a Feature.....	146
Defining the Feature Components	149
Reviewing the Feature Components.....	160
Revising Features.....	161
Copying Features	162
Adding Features During Package Assembly.....	163
Configuring Features During Package Build Definition.....	166
Installing Packages Containing Features	170
Understanding Feature Entries in the Package.INF file.....	171
Viewing Package Build History and Logs.....	172
Viewing the Package Build History.....	173
Viewing Logs.....	174
Resubmitting a Package Build.....	182
Changing the Build Status	183
Understanding Package INF Files	186
Sample INF File.....	186
Understanding Feature INF Files	193
Sample Feature INF File	193
Deployment.....	199
Understanding Package Deployment	200
Deployment to Workstations without OneWorld.....	200
Deployment to Workstations with OneWorld Already Installed.....	201
Deployment to Servers	201
Deployment to Tiered Deployment Locations	202
Deploying to Workstations from CD.....	202
Defining Machines, Locations, and Deployment Groups	203
Defining Machines	204

OneWorld Package Management

Defining Locations	210
Defining Package Deployment Groups	212
Revising a Deployment Group	215
Working with the Package Deployment Director	217
Understanding the Package Deployment Director.....	217
Using the Package Deployment Director.....	219
Activating the Scheduled Package	231
Installing a Scheduled Package	232
Deploying Server Packages	234
Using Push Installation	237
Understanding Push Installation	237
Preparing the Enterprise Server for Push Installation	239
Preparing Workstations for Push Installation	239
Scheduling a Package for Push Installation	248
Scheduling the Push Installation Batch Application.....	249
Running the Push Package Installation Results Report	253
Installing OneWorld Workstations from CD.....	255
Defining the CD Writer Location	255
Deploying a Package to the CD Writer Location	258
Creating the OneWorld Installation CD	260
Deploying Partial Package Templates.....	261
Assembling a Partial Package Template	261
Building a Partial Package Template	264
Deploying a Partial Package Template	265
Testing a Partial Package Template.....	266
Package Build Time Comparison	268
Multitier Deployment	269
Understanding Multitier Deployment	271
Multitier Deployment Features.....	271
Multitier Deployment Terminology	272
Multitier Implementation.....	274
Defining Deployment Servers	276
Defining a New Deployment Server.....	276
Revising an Existing Deployment Server Definition	280
Scheduling Packages for Deployment Locations.....	282
Distributing Software to Deployment Locations.....	283
Distributing Software through Package Deployment	284
Distributing Software Through the Multitier Deployment Batch Process.....	285
Copying Workstation Installation Programs to Deployment Locations.....	287
Installing Workstation Packages From Deployment Locations	289
Multitier Deployment for Server Packages.....	290
Smart Deployment.....	291
Automatic Package.inf File Updating	292

Deploying a Server Package with Multitier Deployment..... 292
Understanding a Multitier Deployment Case Study..... 294

Index 298

Overview of Package Management

The *Configuration Planning and Setup* suite is designed for Configurable Networking Computing (CNC) specialists, OneWorld system administrators, and network/server administrators. The assumption throughout these guides is that the initial OneWorld installation is complete and the standard data sources, path codes, and environments are defined. Use the information in these guides 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:
 - Middleware
 - Understanding and verifying data sources
 - Creating path codes and environments
 - Object Configuration Manager
 - Understanding the different modes of processing
 - Typical OneWorld customer configuration
- *System Administration Guide*. This guide is written primarily for OneWorld 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
 - OneWorld naming conventions
 - jde.ini file

OneWorld Package Management

- *Package Management Guide*. This guide is written primarily for OneWorld system administrators and others who manage custom modifications to the OneWorld environments. *Package Management* contains information about the following topics:
 - Package management planning and setup
 - OneWorld modification rules
 - Object management
 - Package Build
 - Deployment
 - Multitier deployment
- *Server and Workstation Administration Guide*. This guide is written primarily for network administrators and contains information about the following topics:
 - Snapshot (multiclient 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 may find that the information needed to perform the various duties for a position is described in more than one guide. For example, the person who is responsible for setting up path codes, environments, and data sources (described in the *Configurable Network Computing Implementation Guide*) might also be responsible for building and deploying packages (described in the *Package Management Guide*).

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

- Initial installation of OneWorld. See the *OneWorld Installation Guide*.
- OneWorld upgrade and cumulative updates. See the *OneWorld Upgrade Guide*.
- Network infrastructure and third-party software setup and maintenance. This information is provided by the applicable software or hardware vendor. J.D. Edwards does not provide documentation.

You do not need a complete understanding of the installation process to perform configuration planning and setup tasks. However, in order to use the *Configuration Planning and Setup* guides it is important that you understand what the installation accomplishes.

Understanding OneWorld Roles

The OneWorld implementation methodology defines specific roles:

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

Each of these roles is performed by both a consultant and a customer. After implementation, the role of the consultant is diminished. Therefore, it is critical that customers receive adequate training for each of the roles they assume.

CNC Consultant and CNC Administrator

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

Custom Solution Consultant and Application Developers

OneWorld 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 for application development cycle for your business.

Application Consultants and Application Project Leaders

After OneWorld is installed, configured, and rolled out, the application consultants continue in their role as product experts. Although application consultants do not

implement the CNC configurations, they must understand how OneWorld handles distributed processing, data replication, environments, and so on, because these application issues influence the CNC decisions. In addition, application consultants must become very good at troubleshooting potential problems.

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

Implementing OneWorld includes many tasks that are outside the scope of J.D. Edwards services. Third-party consultants provide these services as well as supplementing our staff as CNC consultants, network architects, custom modification consultants, and so on.

Understanding the Package Management Guide

This guide describes how to set up and maintain processes to develop and deploy custom modifications created with the OneWorld Tools.

This guide contains the following topics:

- ❑ Package management planning and setup
- ❑ OneWorld modification rules
- ❑ Object management
- ❑ Package build
- ❑ Deployment
- ❑ Multitier deployment

The first several sections of this guide describe how to set up an environment in which you can deploy custom modifications made with OneWorld Development tools. These sections include information about OneWorld modification rules, transferring objects, checking out development objects, and working with the data dictionary.

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

Topics

- ❑ Package management planning and setup
- ❑ OneWorld modification rules
- ❑ Object management
- ❑ Package build
- ❑ Deployment
- ❑ Multitier deployment

The first several sections of this guide describe how to set up an environment in which you can deploy custom modifications made with OneWorld Development tools. These sections include information about OneWorld modification rules, transferring objects, checking out development objects, and working with the data dictionary.

OneWorld Package Management

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

Understanding Packages

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

There are three package types:

- Full
- Partial
- Update

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

The partial package is a minimum configuration of OneWorld. Its chief advantage is that it saves a tremendous amount of disk space. A user who installs this type of package can subsequently load at runtime only the applications needed, rather than receiving many unused OneWorld 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 only deploy an update package to a workstation that already has OneWorld loaded. 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 deploy software fixes or enhancements quickly.

Understanding the Deployment Programs

OneWorld offers several deployment programs, each with its own specific purpose and advantages. The method you select depends mainly on the type of package you want to deploy.

- Workstation Installation
- Package Deployment
- Multitier Deployment

Understanding Workstation Installation

The Workstation Installation program is used to deploy full and partial packages. Update packages cannot be deployed using Workstation Installation.

The Workstation Installation program retrieves the items specified in the package. A full or partial package is like a bill of materials or a kit with instructions. To pull all of the necessary components, the Workstation Installation program deploys to the local computer. Workstation installations are always initiated by the user.

Understanding Package Deployment

The Package Deployment program enables the OneWorld 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 OneWorld until they load the package.

Users who receive a scheduled package are given the opportunity to load the package immediately after they sign on to OneWorld on the specified date. If they opt to load the package, the installation program launches, and the package loads. The user can also opt to load the package later, or decline 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.

Update packages always use Package Deployment. The full and partial package types are normally deployed via Workstation Installation, although you can use package deployment for a full or partial package in cases where OneWorld has already been loaded on a machine and you want to redeploy.

Understanding Server Packages

A server package is a set of OneWorld objects 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 to multiple servers and multiple path codes.

All OneWorld application development takes place on a workstation with objects stored in a central location. OneWorld allows you to partition business applications to an enterprise server. Developers and system administrators can push 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, Package Build, and Package Deployment. 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 workstations.

Understanding Multitier Deployment

Multitier deployment enables OneWorld workstations to install software from more than one deployment location and more than one deployment machine. J.D. Edwards recommends that you consider multitier deployment if your site has more than 50 workstations performing OneWorld software installations per day, or if you are deploying OneWorld software across a WAN connection.

Package Management Planning and Setup

This section contains recommendations, suggestions, and considerations you should read before you deploy software modifications. You should familiarize yourself with this entire guide before considering your own software deployment strategy.

Managing modifications can be more difficult than the actual programming changes. It is important to have a solid version control plan for tracking changed objects. You can avoid many software problems by tracking changed objects.

You should build and deploy packages only as often as necessary. Deploying excessively could be a huge strain on your version control group, and would become a logistical nightmare as far as planning and tracking your development. If you perform many development changes, you should build and deploy packages on a set schedule. It is very important to keep a schedule so that everyone involved knows when objects are due and when you are going to build and deploy the package.

To give you an idea of how version control might work, at J.D. Edwards we have several hundred developers and an extremely complex Configurable Network Computing (CNC) configuration. For that number of developers, our Product Version Control group consists of the following:

- One manager who oversees the coordination of the department
- One supervisor who coordinates the package builds, coordinates object transfers, and troubleshoots problems
- Two server specialists who build server packages
- Four technical specialists who build workstation packages, perform object transfers, and run automated testing before releasing the package to Quality Assurance
- One night operator who builds workstation packages, builds server packages, and clears build errors

Topics

- ❑ Working with the recommended path codes
- ❑ Ensuring the integrity of the production environment
- ❑ Understanding the development processes
- ❑ Comparing the deployment methods

- ❑ Deploying various types of modifications
- ❑ Deploying data

Working with the Recommended Path Codes

If you are not planning any development projects, you need only three path codes: PY7333, PD7333, and JD7333. You should create a development path code if you plan to do extensive software modification.

The fewer path codes you use, the better. With each additional path code comes version control maintenance that is worth the effort only if there is a true reason for the additional path code. Even when making extensive software modifications, you should have only four path codes (sets of central objects):

- DV7333** Use this path code for "normal" development. Upon successful testing, transfer the objects to your PY7333 path code using Object Transfer, and distribute to your users using the package build deployment process.

- PY7333** This path code contains a practice set of objects that are tested during conference room pilot before transferring objects to production. It is for deploying quick fixes or making minor modifications that you will quickly transfer to production. It can also be used as a place to test modifications that were done in the development path code before taking the risk of transferring them to the production path code.

- PD7333** This is your production path code. Just-in-time installations come directly from this location, and production server objects are also deployed from here. After testing software changes in PY7333, transfer them to PD7333 and then deploy the changes to your enterprise servers and workstations.

- JD7333** This is the set of pristine objects shipped from J.D. Edwards. You should not make changes to this path code other than paper fixes from J.D. Edwards. This path code is used to compare J.D. Edwards standard software to any custom solutions you have implemented in other path codes. You should keep a copy of this path code so that you have a "clean" copy of OneWorld in case you need to refresh anything.

All path codes share the same Object Librarian tables and the same system data source, and normally the same data dictionary. The only distinct tables across path codes are the central objects/specifications (F987*), the version list (F983051), the processing options text (F98306), and the user overrides (F98950).

Suggested Package Names

At J.D. Edwards we have found that if you have an A and B version of each package, you can alternate between these 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 same 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 would have four packages per path code. This setup gives you two full packages (A and B) for production, and two partial packages (A and B) for production. For example:

- PD7333FA** Standard Production Full A
- PD7333FB** Standard Production Full B
- PD7333PA** Standard Production Partial A
- PD7333PB** Standard Production Partial B

Update packages might be named as follows:

- PD7333UA** Production Update Package 1
- PD7333UB** Production Update Package 2
- PD7333UA** Production Update Package 3
- PD7333UB** Production Update Package 4

Ensuring the Integrity of the Production Environment

As soon as you transfer objects into your PD7333 path code, the changes can be accessed by end users. Therefore, you must make sure to test the modified objects before transferring them to production.

There are two ways that objects transferred into the production path code can be immediately deployed to end-users:

- Anyone using just-in-time installation (JITI) will have an application installed from the central objects data source the first time it is used. This includes people who are using a partial package or anyone who has received an update package containing an application without specifications. For more information about just-in-time installation, see [Understanding Just-in-Time Installation](#).
- If you build an update or partial package and do a business function build for that package, the business function included in that package is built, then globally linked with all other business functions in the path code's check-in location.

The only way you can ensure that modifications transferred to your production path code are not immediately available to end users is to avoid using partial packages or update packages containing applications without specifications. Also, do not transfer business functions into the production path code until you are ready to deploy, because a global build of business functions done during a package build will automatically include the new functions. You can be assured that when you transfer changes into the production path code they will not be available until you build a full or update package.

Disabling Just-In-Time Installation

There are two methods for disabling just-in-time installation:

- Through Environment Master (P0094), you can disable JITI for the environment. That is, anyone who signs on to that environment and doesn't have specifications for an application will not be able to retrieve them because JITI is turned off. J.D. Edwards recommends that you use this flag during the cumulative installation process when you update your production central objects with the new J.D. Edwards changes. Any user who has installed a partial package or received an update package containing

applications without specifications will not be able to access that application when JITI is disabled.

- You can disable JITI for a particular application, for an end user, or for a group profile through application security. When a user accesses an application whose specifications do not reside on his workstation, the JITI engine first checks Environment Master to see if JITI is disabled for the entire environment. If JITI is on for the environment, the engine looks to see if that user is prevented through application security from using JITI. Application security has a field, "not allowed to install." For more information, see the *Security* section in the *System Administration Guide*.

Even if you disable JITI, data dictionary items are still copied using just-in-time-installation to the global tables. Due to the data dictionary structure, this process cannot be disabled.

Understanding the Development Processes

You should review this section to understand the development processes that J.D. Edwards recommends. The section includes information about developing long-term enhancements and short-term fixes, and consists of the following topics:

- ❑ Working with the normal development process
- ❑ Developing short-term fixes

Topics

- ❑ Working with the normal development process
- ❑ Developing short-term fixes

Working with the Normal Development Process

The following list provides an overview of how you should perform your normal development cycle.

In the DV7333 Path Code:

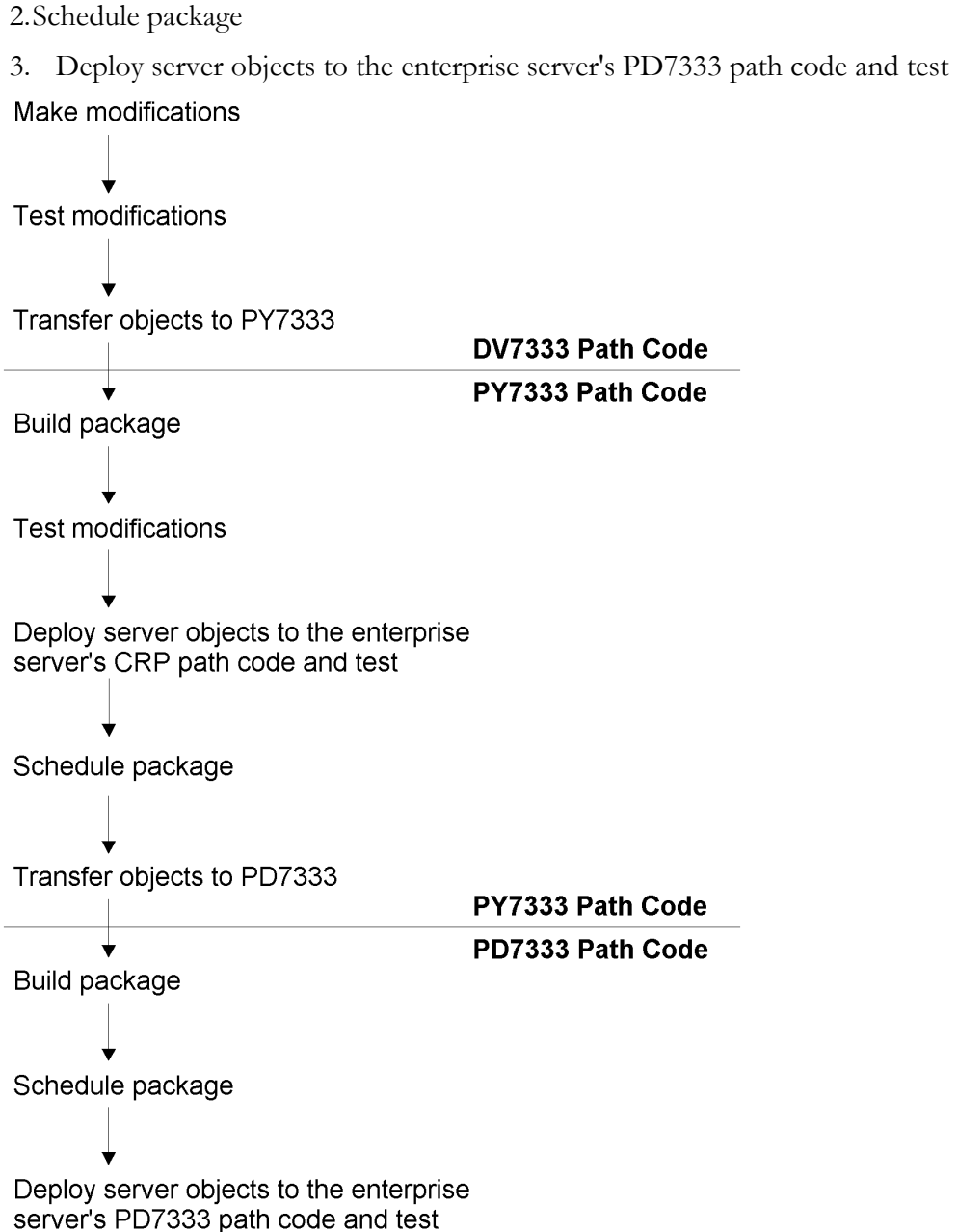
1. Make modifications
2. Test modifications
3. Transfer objects to PY7333

In the PY7333 Path Code:

1. Build package
2. Schedule package
3. Deploy server objects to the enterprise server's CRP path code and test
4. Schedule package
5. Transfer objects to PD7333

In the PD7333 Path Code:

1. Build package



Follow this procedure during the normal development process:

6. Make modifications.
 - Check out your objects from the DV7333 path code. Modify the objects, test them, and check them back in.
 - Use the SAR system (or any numbering system) to track your changes. Always check in the objects with a SAR number.
7. Test modifications.

If this object needs to be on the logic server, transfer the object to the server's DV7333 path code. Test the object against the server.

8. Transfer objects.

Use a SAR number with Object Transfer to transfer the object to the PY7333 path code. Use the check-out log to confirm the transfer (optional). This object is not in production, but it is now available for you to build a test package in the PY7333 path code.

9. Build a package.

Build a package (full, partial, or update).

10. Test modifications.

- Test the newly built, but unreleased, package in the PY7333 path code (you can only test against workstation processes, not server processes). Remember, even though the name of this package will probably be PY7333U1 (update package number 1 for the CRP path code), it is a test package right now because you have not released it to your users.
- Schedule the update package to a test machine and test in an environment using CRP objects with CRP data.

11. Deploy server objects.

- Deploy server objects to the PY7333 path code on the enterprise server and test. If you prefer, you can build the server package and schedule deployment at the same time you build and schedule the workstation package.

12. Schedule the new package to CRP users.

13. Transfer objects.

- Use a SAR number with Object Transfer to transfer the object to the PD7333 path code. Use the check-out log to confirm the transfer (optional). This object is now in production. It is now available for you to build a package in the PD7333 path code.

Note: If just-in-time installation is enabled, the objects can be accessed immediately.

14. Build a package.

- Build a client workstation package (full, partial, or update).
- Perform a server package build. (Optionally, you can transfer the server package now or wait until it has been tested on a workstation.)

15. Schedule the new package to end user workstations.
16. Deploy server objects.

Deploy server objects to the PD7333 path code on the enterprise server and test. If you prefer, you can build the server package and schedule deployment at the same time you build and schedule the workstation package.

Developing Short-Term Fixes

Sometimes you will want to make a simple change to an application that is undergoing major enhancement work in the DV7333 path code. When objects have been modified in the DV7333 path code, it might be too risky to make a simple change and get it deployed quickly. Therefore, you should make the change in PY7333 so that you can quickly get the fix to users. Do not forget to make the change to the PY7333 and the DV7333 path codes.

Follow this procedure to develop a short-term fix:

1. Make modifications.
 - Check out your objects from the PY7333 path code. Modify the objects, test them, and check them back in.
 - Use the SAR system (or any numbering system) to track your changes; always check in the objects with a SAR number.
2. Transfer objects.
 - Use a SAR number with Object Transfer to transfer the object to the PD7333 path code. Use the checkout log to confirm the transfer (optional). This object is now in production. It is now available for you to build a package in the PD7333 path code.

Note: If just-in-time installation is enabled, the objects could be accessed immediately.
3. Build a package.
 - Build a package (full, partial, or update).
4. Schedule the new package to end user workstations.

Comparing the Deployment Methods

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

Understanding Cumulative and Noncumulative Update Packages

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

17. Change the package status in order to add the new modification(s).
18. Add the changed or new items to the package.
19. Redeploy the package.

If you use a noncumulative strategy, you would create and deploy a different package every time instead of using the same package. For example, if you deploy one modification per week for 10 weeks, there would be 10 different packages, each containing only the software changes for that week.

Recommendations for Sites Using Partial Packages With JTI

For sites using partial packages and JTI, J.D. Edwards recommends that you adopt a noncumulative update package strategy. Each week that you build a new update package, if you have a modification to deploy that cannot be done through just-in-time installation (such as business function changes), we recommend that you also rebuild the partial package for that week to ensure that the partial package contains the latest changes. If you follow this recommendation, when you want to load a new workstation (or completely refresh any machine), you simply need to install the partial package; you don't need to install any update packages.

Recommendations for Sites Using Full Packages Without JTI

For sites using full packages with JTI, 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 it. The advantage of this strategy is that you do not need to rebuild your full package each week. By using this strategy to load a new workstation (or to completely refresh any machine) you will need to install the full package and the one cumulative update package.

The disadvantage of using this strategy is that the update package may become so big that deployment time is increased. You will need to determine your threshold for when to rebuild the full package that new workstations will load, while existing users will install the new update package.

Recommendations for Developers

Developers must use full packages. You may deploy update packages to them, but developers do not use partial packages because partial package machines never receive development objects.

When installing a full or update package, machines with compilers will automatically receive development objects. Developers receive source, headers, and libraries. The installation process builds the object into the business function library and DLL (dynamic link library), and thereby preserves any business function changes the developer made. For more information about package types, see [Understanding Packages and Deployment](#).

Multitiered Deployment and Just-in-Time Installation

Just-in-time installation (JIT) can be used in remote locations that are using multitiered deployment to install packages. However, you might find that performance time for the just-in-time installation is unacceptable. In this case you may wish to use full packages and deploy software changes through update packages.

Comparing the Deployment Methods

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

- A new user loading a new machine should use the Workstation Installation 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 keep track of which update packages must be applied after installing a particular package.
- All update packages must use the Package Deployment program to be scheduled to workstations unless you are using the Push Installation feature.
- The full and partial packages can also use Package Deployment if OneWorld has already been loaded on a machine. (You can use the Push Installation feature to deploy to a machine that does not have OneWorld installed.)
- If workstation disk space is a consideration, consider deploying a partial package that provides the minimum configuration of OneWorld, and yet allows users to load only the applicable OneWorld applications. In this situation you could use either the Workstation Installation 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 program to install from more than one deployment location. You should consider this method if you have more than 50 workstations performing OneWorld software installations per day.
- Use the Package Deployment program when you need to push objects in a server package from the central objects data source to enterprise servers.

Deploying Various Types of Modifications

It is important that you understand which types of objects (and therefore modification) can be deployed through each package type.

Partial packages include the following:

- All consolidated business function dynamic linked libraries (DLLs)
- Specifications for only those applications necessary to sign on and get to OneWorld Explorer
- All icons
- Helps, data, and 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
Vocab Overrides (FDA text)	X	X	X
Data Structure	X	X	X
Process Options (Report)	X	X	X
Business Functions:			
C Language Source/Include/Object (if there is a compiler)	X	X	
Consolidated BSFN DLLs	X	X	
Data Structure	X	X	

Modification	Full Package	Update Package With Specs	Update Package With no Specs
Table ER	X	X	
Named ER	X	X	
Batch Applications:			
Report	X	X	X
Imbedded ER in Report	X	X	X
Report Data Structure	X	X	X
Report Vocab Overrides	X	X	X
Report Processing Options	X	X	X
Versions and Processing Option Values (depends on processing options)	X	X	X
Imbedded ER in Versions	X	X	X
Processing Option Templates	X	X	X
Business Views:			
Fields Added/Changed	X	X	X
Tables:			
Structure (specifications)	X	X	
Indexes	X	X	
Joins	X	X	
Miscellaneous:			
Helps Stored on the Server			

Modification	Full Package	Update Package With Specs	Update Package With no Specs
Helps Stored on the Workstation	X	X	
Generic Text Data Structure	X	X	
Data Dictionary Items (see Data Dictionary Administration in the System Administration Guide)			
Foundation Code (required for full and partial packages, and optional for update packages)		X	X
Foreign Languages	X	X	X
Non-OneWorld Objects (custom items can be deployed through any package type)	X	X	X
Replicated Local Data (required for full and partial packages, and optional for update packages)	X	X	X
New Icons	X	X	

For an update package with JITI (Just-in-time-installation) types, the following changes are installed:

Applications:

- Imbedded ER
- Vocab Overrides (FDA text)
- Data Structure
- Process Options (Report)

Batch Applications:

- Report
- Imbedded ER is Report
- Report Data Structure
- Report Vocab Overrides

- Report Processing Options
- Versions and Processing Option Values (depends on processing options)
- Imbedded ER in Versions

Business Views:

- Fields Added/Changed

Miscellaneous:

- Foundation Code (foundation is required for full and partial packages, and optional for update packages)
- Foreign Languages
- Non-OneWorld Objects (custom items can be deployed through any package type)
- Replicated Local Data (data is required for full and partial packages, and optional for update packages)

Deploying Data

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

Deploying a New Replicated Local Database

Normally you do not need to deploy a new Microsoft Access database because data replication takes care of pushing changes to end users. The database in the package cannot be set up as a subscriber because it is a snapshot in time of your database. Therefore, if you want to keep the database synchronized with the publisher tables, you need to refresh the data periodically.

You should consider keeping the database synchronized with the publisher tables. When an existing OneWorld machine does a complete refresh by installing a full or partial package, it receives old data. It is not marked as out of sync, so data replication has no way of knowing that the machine reinstalled OneWorld with an out-of-date Microsoft Access database.

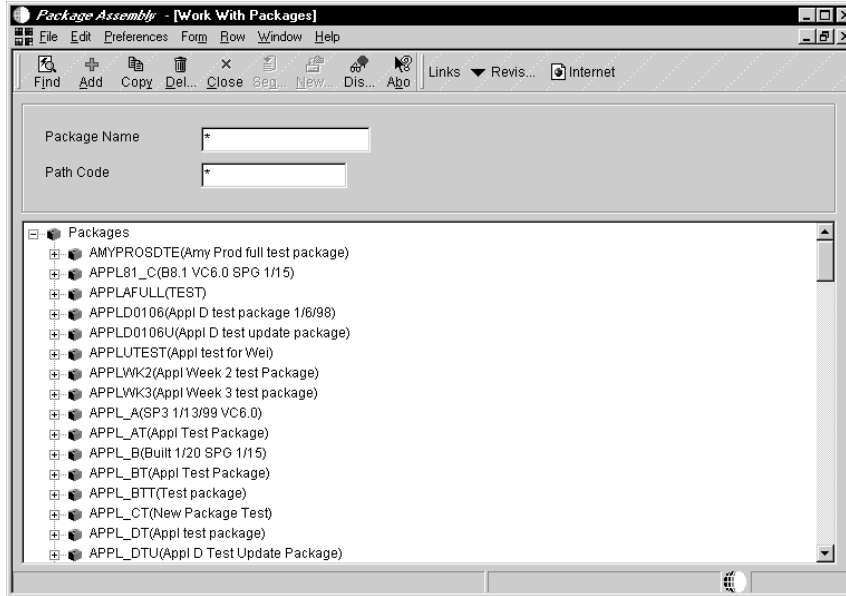
If a new machine is set up, the administrator would have to set up the user as a subscriber, and that machine will be marked as out of sync. Then, when signing on to that new machine for the first time, the user will be prompted to receive the changed data.

► To refresh an existing package with a new database

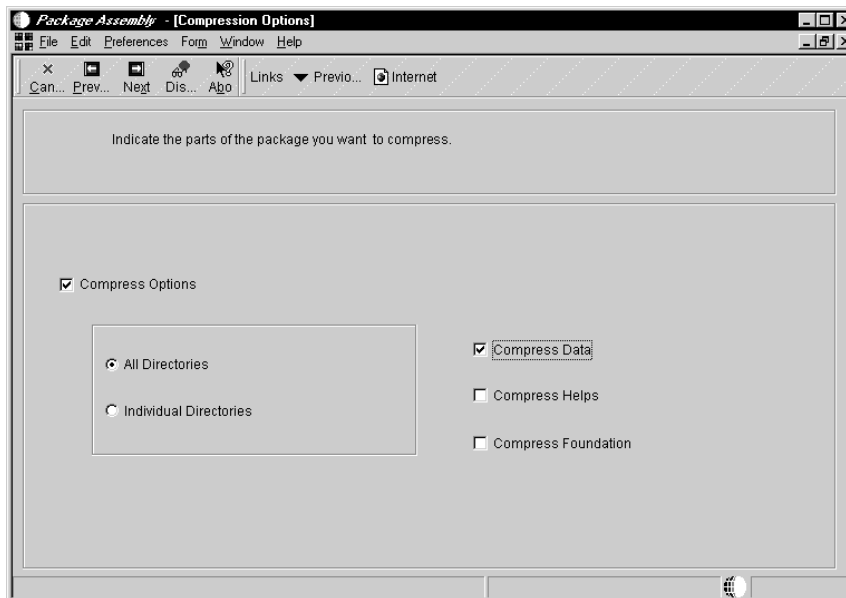
1. Locate a Microsoft Access database from a subscriber machine that has received all data changes.
2. Copy that database to the data location for that package.

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

The Work With Packages form appears.



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



5. On the Compression Options form, click only the Compress Options and Compress Data options.

This recompresses the database, and anyone who installs this package will get current data.

Note:

To verify that compression was successful, check the size and date of the JZ file under `jdwardsoneworld/B7333/pathcode/package/data`. If the size is only 1KB, this means the compression failed.

Deploying a Special Database for Store-and-Forward Users

Store-and-forward users require Microsoft Access databases that are much larger than normal users. Therefore, you will want to build an update package that has one line item for the special Access database. Store-and-forward users must take full packages because they will not be connected to the network to install their application through just-in-time installation.

For this reason, they 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 created for store and forward users. See *Working with Store-and-Forward* in the *Configurable Network Computing Implementation Guide* for more information.

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

1. Define the update package with one line for a database.

Ensure that the special database that you have created for store-and-forward users actually resides in the source destination defined through Database Item Revisions.

2. Choose Package Assembly (P9601) from the Package and Deployment Tools menu (GH9083).
3. On Work with Packages, choose the package and choose Define Build from the Row menu.
4. Proceed through the Package Assembly Director forms until you reach the Compression Options form.
5. On the Compression Options form, click the Compress Data option.
6. Schedule the package to all machines that will be doing store-and-forward transactions.

OneWorld Modification Rules

Because the OneWorld 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 mods," which are modifications that you can perform easily without the help of a developer. You can perform modless mods on the following:

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

This kind of 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 functionality using processing options

However, even with the full commitment of J.D. Edwards to making the tools as flexible and robust as possible, if the need does arise to modify OneWorld, there are certain rules and standards to ensure that software modifications perform like modless mods for a seamless and predictable upgrade to the next release level.

You should prepare for the upgrade before making any custom modifications to ensure a smooth upgrade. If you plan modifications properly, you will have minimal work to do following an upgrade. This should result in the least amount of disruption to your business and a reduction in the overhead cost involved during an upgrade because the upgrade time is reduced.

OneWorld keeps track of all custom modifications as you check them into the server. Using this feature, you can run the Object Librarian Modifications (R9840D) report prior to a merge to see a list of the changed objects.

OneWorld consists of control tables (such as menus, user defined codes, versions, and the data dictionary) and transaction tables (such as Address Book and the Sales Order File). J.D. Edwards provides control tables with data that you can modify, while transaction files contain your business data.

During an upgrade, both sets of tables go through an automatic merge process. Control tables are merged with new J.D. Edwards data, whereas transaction files are converted to the new specifications without change to your existing data. For the object specification merges (such as business view, tables, data structures, processing options, event rules, and applications), there are processes in place that merge the specifications or replace them depending upon the rules defined in OneWorld. See the following section for more information.

What an Upgrade Preserves and Replaces

If your business needs require custom modifications, use the following general rules for OneWorld modifications to help ensure a smooth and predictable upgrade. These rules describe which modifications the upgrade process preserves and which modifications it replaces.

- Preserve means that, during an upgrade, OneWorld automatically merges your modifications with the new J.D. Edwards applications shipped with the upgrade, and that you do not lose your modifications. If there is a direct conflict between your specifications and J.D. Edwards specifications, the upgrade process uses yours. When there is no direct conflict between the two, then the upgrade process merges the two specifications.
- Replace means the upgrade does not merge those types of modifications and, therefore, J.D. Edwards replaces your modifications. You will need to do them again after the upgrade completes.

J.D. Edwards provides you with the Object Librarian Modifications (R9840D) report, which you can run before the upgrade process to identify objects that you modified.

Topics

- ❑ General rules for modification
- ❑ Interactive applications
- ❑ Reports
- ❑ Application text changes
- ❑ Table specifications
- ❑ Control tables
- ❑ Business views
- ❑ Event rules
- ❑ Data structures
- ❑ Business functions
- ❑ Versions

General Rules for Modification

The following general modification rules apply to all OneWorld objects:

- When adding new objects, use system codes 55 - 59. OneWorld uses reserved system codes that enable it to categorize different applications and vertical groups. By using system codes 55 - 59 for your custom usage, OneWorld 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 are reserved prefixes for standard version templates that J.D. Edwards sends out for you to copy, to create new templates or versions. Using these prefixes does not preserve your custom versions in case of a naming conflict.
- For upgrades, you should 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 you can access those specifications for comparison or for troubleshooting purposes. See the *OneWorld Upgrade Guide* for information.

Interactive Applications

Do not delete controls, grid columns, or hyperitems on existing OneWorld applications. Instead, hide or disable them. OneWorld might use these items for calculations or as variables, and deleting them might disable major functionality.

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		<p>There are two ways to create an application: create it from scratch, or copy an existing application using the Application Design Aid's "Copy" feature. This allows you to copy all of the application specifications, including event rules.</p> <p>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</p>

Object	Preserved	Replaced	Comments
			J.D. Edwards might have made to the original application you copied.
New hyperitems 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, so if you adjust the style you need to do the same for any new controls added to an application.
Code-generator overrides	X		
Data dictionary overrides	X		
Location & size changes	X		If J.D. Edwards decides to place a new control in the new release of the software in the same place you have placed a custom control, the controls display on top of each other. This does not affect the event rules or the functionality of the application. After the upgrade, you can rearrange those controls using Application Design Aid.
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		X	Instead of putting custom forms on existing OneWorld applications, it is preferable to create a

Object	Preserved	Replaced	Comments
OneWorld applications			custom application using system codes 55 - 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. From a performance standpoint there is no difference if an external application is called from a row exit or a form within the application.

Reports

The following rule applies to Report Design Aid specifications:

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

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		There are two ways to create a report: create it from scratch or copy an existing report using Report Design Aid "Copy" feature. This feature enables you to copy all the report specifications, including event rules. If you use it to copy an existing report for some modifications, during an upgrade your new report does not receive any J.D. Edwards updates made to the original report you copied.
New constants added to existing reports	X		
New alpha variables added to existing	X		

Object	Preserved	Replaced	Comments
reports			
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, so if you have adjusted the style, you need to do the same for any new controls added to a report.
Location and size changes for objects	X		If J.D. Edwards decides to place a new object, such as a control, in the new release of the software in the same place you have placed a custom object, the objects display right next to each other. This does not affect the event rules or the functionality of the report in any way. After the upgrade, you can rearrange objects using Report Design Aid.
Data dictionary overrides	X		

Object	Preserved	Replaced	Comments
overrides			
Custom sections on existing OneWorld 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 - 59. There is no difference in the performance of a report being called through report interconnections.

Application Text Changes

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 done in Application Design Aid		X	
Overrides done in Report Design Aid		X	
Overrides done in Interactive Vocabulary Override		X	
Overrides done 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 Indexes to OneWorld tables	X		

OneWorld tables			
Columns added to or removed from existing OneWorld tables		X	This includes changing field length, field type, and decimal position. Instead of adding a new column to an existing J.D. Edwards table, use a tag file with system codes 55 – 59.

For custom tag files, be aware of data item changes in the J.D. Edwards data dictionary. From one release to the next, J.D. Edwards might change certain data item attributes such as data item size, which can 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 takes care of the data dictionary changes by upgrading the OneWorld database to the new release level. An upgrade preserves custom indices over 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 OneWorld tables, as the basis to do 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 OneWorld data items, your changes override. Create new data items using system codes 55 - 59.

User defined codes	X		The upgrade process merges any new hard-coded OneWorld values. (Values owned by J.D. Edwards are system 90 and above, and H90 and above.) The process also reports any OneWorld hard-coded values that conflict with your custom values.
Menus	X		In case of a conflict with OneWorld base menus, your custom changes override.
Columns added or removed from existing OneWorld 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 at the application design level using either Application Design Aid or Report Design Aid. There is not much of a performance gain by deleting a few columns from a business view.

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 OneWorld business views	X		
Columns removed from J.D. Edwards business views.		X	

Event Rules

What an Upgrade Preserves and Replaces

The following table shows the event rules 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 OneWorld applications, reports, and tables that do not have any OneWorld event rules attached to the same event	X		
Events for OneWorld business functions that do not have any OneWorld event rules attached to the same event	X		
Events for OneWorld 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 event rules. The merge prints a report (R9840D) to notify you of any event rules that the upgrade process disabled.
Events for OneWorld 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 event rules. The merge prints a report (R9840D) to notify you of any event rules that the upgrade process disabled.

To restore your custom event rules to OneWorld 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 (R9840D) report to identify modified applications
- Print the event rules for the modified application so that you can see the logic behind the event when restoring 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	Comments
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 OneWorld forms data structures		X	
Modifications to existing OneWorld processing options data structures		X	
Modifications to existing OneWorld reports data structures		X	
Modifications to existing OneWorld business functions data structures		X	
Modifications to existing OneWorld generic text data structures		X	

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

Business Functions

For any new custom business functions, create a new (custom) parent DLL to store your custom modifications. See the *OneWorld 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 for manually refitting 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 modifications to the source of existing base OneWorld business functions, use a third-party source-compare tool, such as Microsoft WinDiff. To determine modifications to APIs within business functions, see the OneWorld online help feature for the most current 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 OneWorld 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 *OneWorld 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	Comments
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 using JDE processing option templates. A warning is issued at runtime, and some data may be lost.

Also, event rule modifications for custom versions on JDE templates are not reconciled with the JDE 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. In OneWorld, an object is a reusable entity that is based on software specifications. A specification is a complete description of a OneWorld object. Each object has its own specification, which is stored on both the server and the workstation.

OneWorld stores objects in two places:

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

This section contains conceptual information about objects, such as how objects are moved between the central objects location and the object destinations. For information about how to actually move and manipulate objects (such as checking objects in or out, adding objects to a project, or getting objects without checking out,) see *Object Management Workbench* in the *OneWorld Development Tools Guide*.

Topics

- ❑ Understanding object movement
- ❑ Understanding files created by the build process
- ❑ Correlating replicated and central objects
- ❑ Performing backups and restoring objects
- ❑ Copying Solution Explorer records between data sources

Understanding Object Movement

This chapter describes the movement of objects between the central objects location and the object destination when you do any of the following tasks:

- Check objects in or out
- Add existing objects to a project
- Perform a Get Object
- Run the Workstation Installation program

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.

See Also

Object Management Workbench in the *OneWorld Development Tools Guide*, for more information about checking objects in or out, adding objects to a project, or performing a get object.

Topics

- ❑ Table (Object Type TBLE)
- ❑ Business View (Object Type BSVW)
- ❑ C Business Function (Object Type BSFN, Source Language C)
- ❑ Business Function Data Structure (Object Type DSTR)
- ❑ Embedded Event Rule
- ❑ Business Function Event Rule (Object Type BSFN, Source Language NER)
- ❑ Media Object Data Structure (Object Type GT)
- ❑ Interactive Application (Object Type APPL)
- ❑ Batch Application (Object Type UBE)
- ❑ Data Dictionary Items

Table (Object Type TBLE)

Check In/Out, Add Object, and Get Object

With Check In/Out, Add Object, and Get Object, 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)

Workstation Installation

See "Workstation Installation" under Interactive Application (Object Type APPL) for information.

The table header is not the same as the actual table residing in a database. The table itself is created through Table Design Aid when you generate. The Workstation Installation program copies this table to the workstation if it is stored in Microsoft Access.

Business View (Object Type BSVW)

Check In/Out, Add Object, and Get Object

With Check In/Out, Add Object, and Get Object, specifications move.

Workstation Installation

See "Workstation Installation" under Interactive Application (Object Type APPL) for information.

C Business Function (Object Type BSFN, Source Language C)

Check In/Out, Add Object, and Get Object

With Check In/Out, Add Object, and Get Object, the following objects move:

Specifications

Source files (*.c)

Header files (*.h)

Object files (*.obj)

Workstation Installation

See "Workstation Installation" under Interactive Application (Object Type APPL) for information.

Business Function Data Structure (Object Type DSTR)

Check In/Out, Add Object, and Get Object

With Check In/Out, Add Object, and Get Object, the following objects move:

Specifications

Workstation Installation

See "Workstation Installation" under Interactive Application (Object Type APPL) for information.

Embedded Event Rules

Check In/Out, Add Object, and Get Object

You cannot check out embedded event rules. Embedded event rules are moved when the object into which the event rule is embedded is checked out. For example, if there are embedded event rules attached to a table, interactive application, or batch application, when you move them, the specifications for the embedded event rule moves with them.

Workstation Installation

See "Workstation Installation" under Interactive Application (Object Type APPL) for information.

Business Function Event Rule (Type BSFN, Source Language NER)

Check In/Out, Add Business function event rules (object type BSFN) can be checked out.

Object, and Get Object

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.

Embedded event rules are moved when the object into which the event rule is embedded is checked out. For example, if there are embedded event rules attached to a table, interactive application, or batch application, when you move them, the embedded event rule moves with them.

With Check In/Out, Add Object, and Get Object, the following objects move:

Specifications

Source (*.c)

Header (*.h)

Object (*.obj)

Workstation Installation

See "Workstation Installation" under Interactive Application (Object Type APPL) for information.

Media Object Data Structure (Object Type GT)

Check In/Out, Add Object, and Get Object

With Check In/Out, Add Object, and Get Object, data structure specifications move.

Workstation Installation

See "Workstation Installation" under Interactive Application (Object Type APPL) for information.

Interactive Application (Object Type APPL)

Check In/Out, Add Object, and Get Object

With Check In/Out, Add Object, and Get Object, the following specifications move, which include the following:

Application

Form

Form data structure

Embedded event rules

Workstation Installation With the Workstation Installation program, all objects and the Microsoft Access database, which contains replicated data, are copied from the package to the workstation. This does not apply to objects in the package for which you do not include specifications. In this case objects are not replicated, but installed on the workstation through just-in-time installation.

Batch Application (Object Type UBE)

Check In/Out, Add Object, and Get Object With Check In/Out, Add Object, and Get Object, report and event rule specifications move. You must check in/out the version separately from the report.

Workstation Installation See "Workstation Installation" under Interactive Application (Object Type APPL) for information.

Batch applications are interpretive and therefore do not have a DLL.

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 you want replicated to servers and workstations. OneWorld 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 Detail (F00165)

When you change data dictionary items, make sure you replicate those changes across your enterprise. In a coexistence configuration, also make sure your WorldSoftware and OneWorld data dictionaries are synchronized.

There are two methods of replicating data dictionary changes:

- Use the Data Replication (P98DREP) program to replicate data dictionary changes to workstations. See the *Data Replication* section in the *System Administration Guide*.
- Use the batch replication method to replicate data dictionary changes to other servers.

Understanding Files Created By the Build Process

This chapter contains the following topics:

- ❑ Workstation Build
- ❑ UNIX Server Build
- ❑ Windows NT Server Build
- ❑ AS/400 Server Build

Topics

- ❑ Workstation Build
- ❑ UNIX Server Build
- ❑ Windows NT Server Build
- ❑ AS/400 Server Build

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 contained in each DLL. It prevents memory allocation problems and avoids platform-specific limits of exported procedures per DLL.

The production environment PD7333/bin32 directory contains the DLLs that are created on the workstation. All of the business function source files are in the PD7333/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 provide, builds the business functions into their respective DLLs, and creates the following files:

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

When calling a business function from a business function, you must use the `jdecallocobject` API.

See *Creating and Specifying a Custom DLL* in the *OneWorld Development Tools Guide* for details about how to build business functions into the custom DLL.

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`

UNIX Server Build

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)

Business function shared libraries (equivalent to a Windows NT workstation's DLLs) are consolidated. Therefore, one shared library is created for each parent DLL found in the Object Librarian Detail file (F9861). If you are creating custom business functions, make sure you use a custom parent DLL instead of one of J.D. Edwards parent DLLs.

Where Business Functions Are Stored

Business function shared libraries on UNIX platforms are grouped by related business functions. This grouping limits the size and number of procedures contained in each shared library. It prevents memory allocation problems and avoids platform-specific limits of exported procedures per shared library. Beneath the environment PD7333 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:

```
PD7333
  source
    CAEC
    CALLBSFN
    CCORE
    CDESIGN
    CDIST
    CFIN
    CHRM
    CMFG
    JDBTRIG
```

In each directory are the business function source files belonging to the shared library. All shared libraries are installed in the PD7333/bin32 directory, and have the same name as the shared library subdirectories with a prefix of lib and a suffix of .sl (HPUX) or .so (AIX).

Specification Files

Specification files must be consolidated into individual files before being transferred to a UNIX server for translation. A specification file is a collection of two files, grouped by the same prefix (RDATEXT for example). The two suffixes, which comprise a complete specification file, are .ddb (the data file) and .xdb (the index file).

Windows NT Server Build

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 as they are on the UNIX platform or workstation. Therefore, one shared library is created for each parent DLL found in the Object Librarian Detail file (F9861). If you are creating custom business functions, make sure you use a custom parent DLL instead of one of J.D. Edwards parent DLLs.

Where Business Functions Are Stored

Business function DLLs on Windows NT platforms are grouped by related business functions. This grouping limits the size and number of procedures contained in each DLL. It prevents memory allocation problems and avoids platform-specific limits of exported procedures per DLL.

Beneath the environment PD7333 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:

PD7333

source

CAEC
CALLBSFN
CCORE
CDESIGN
CDIST
CFIN
CHRM
CMFG
JDBTRIG

Beneath each directory are all of the business function source files belonging to the DLL. All DLLs are installed in the PD7333\bin32 directory, and have the same name as the DLL subdirectories with the .dll suffix.

Specification Files

Specification files must be consolidated into individual files before being transferred to a Windows NT server for translation. A specification file is a collection of two files, grouped by the same prefix (such as RDATEXT, for example). The two suffixes, which comprise a complete specification file, are .ddb (the data file) and .xdb (the index file).

AS/400 Server Build

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:

- Modules (*MODULE type)

- Binding directories (*BNDDIR type)
- Service programs (*SVRPGM type)

Business function service programs (equivalent to a Windows NT workstation's DLLs) are consolidated. Therefore, one service program is created for each parent DLL found in the Object Librarian Detail file (F9861). If you are creating custom business functions, make sure you use a custom parent DLL instead of one of J.D. Edwards parent DLLs.

Where Business Function Source Members Are Stored

Shared libraries on AS/400 platforms are grouped into service programs of related business functions. This grouping limits the size and number of procedures contained in each service program. It prevents memory allocation problems and avoids platform-specific limits of exported procedures per service program.

In the environment PD7333 directory are three groups of files: the service program file, the H file for include files, and the HXX file for event rule header files. The PD7333 library should have the following files:

```
PD7333
  CAEC
  CALLBSFN
  CCORE
  CDESIGN
  CDIST
  CFIN
  CHRM
  CMFG
  H
  HXX
  JDBTRIG
```

Included in the PD7333 library but not listed above are the modules for all business functions.

Beneath each service program file are the business function source members. All service programs are installed in the PD7333 library, and have the same name as the service program files above, with a type of *SVRPGM.

Specification Files

Specification files must be consolidated into individual files before being transferred to an AS/400 server for conversion. A specification file is a collection of two files grouped by the same prefix (RDATEXT for example). The two suffixes, which comprise a complete specification file, are .ddb (the data file) and .xdb (the index file).

Correlating Replicated and Central Objects

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

Replicated Object	Central Object and Description
DDTABL	F98710, Table Header. One record for each table.
DDCLMN	F98711, Table Columns. One record for each column in a table.
DDPKEYH	F98712, Primary Index Header. One record for each table index.
DDPKEYD	F98713, Primary Index Detail. One record for each column in an index.
BOBSPEC	F98720, Business View Specifications. One record for each business view.
GBRLINK	F98740, Event Rules Link. 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	F98741, Event Rules Specifications. One record for each line of event rules.
DSTMPL	F98743, Data Structure Templates. One record for each business function, processing option, form interconnection, report interconnection data structures.
FDATEXT	F98750, Form Design Aid text. Override text for applications.
FDASPEC	F98751, Form Design Aid Specifications. One record for every column, grid line, button, hyperitem, control, and so on, in the application.

ASVRHDR	F98752, FDA/SVR Header Information. One record for each application (if the application has processing options, that information is also stored in the record).
ASVRDTL	F98753, FDA/SVR Detail Information. One record for each form (includes references to the data structures).
RDATEXT	F98760, Report Design Aid Text. Override text for batch reports.
RDASPEC	F98761, Report Design Aid Specifications. One record for each section, column, sort, constant, and so on, in Batch Reports and Versions.
JDEBLC	F98762, Business Function Source Information. 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	F9200, Data Item Master
	F9202, Data Field Display Text
DDINDEX: obsolete in B73.2 - indexes for dictionary	F9203, Data Item Alpha Description
	F9207, Data Dictionary Error Message Information
DDTEXT: Data Dictionary text	F9210, Data Field Specifications
NEXTID	F98701, Next IDs Table. Local record of next IDs assigned to each workstation.
GLBLTBL	Cache Information From Data Dictionary and Table Specifications. 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 you need.

J.D. Edwards recommends the following:

- **Developer Workstation backups:** Developers at J.D. Edwards do not back up their OneWorld 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 broken applications into the DEV path code.

- **End User Workstation backups:** End users should not have nonreplicated 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 for the day. If they cannot do so, you should have a procedure in which they back up their local database to a file server with their name as a subdirectory.
- **Development Server backups:** At J.D. Edwards, our IT department backs up both our development file server (normally your deployment server) and necessary databases (central objects, Object Librarian, and data dictionary). When someone needs to restore a particular object from backups, our database administrators restore the export to a path code called "Restore." A developer would then check the object out from Restore, ensure that the objects are as expected, and check into the normal development path code.
- **Deployment Server backups:** The entire server does not need to be backed up nightly. The only directories that might change on a daily basis are:
 - The DEV path code if you are modifying objects, building new packages or updating the Access database delivered during a workstation installation.
 - HELPS if you are modifying help files.
 - MEDIA OBJ if your media objects are residing on the deployment server. OneWorld 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:
 - You should back up DBMS nightly. J.D. Edwards recommends using the backup tool provided by the RDBMS vendor.
 - Back up OneWorld objects by backing up the entire OneWorld directory. However, you do not need to back up any directories other than the PROD and DEV path codes and the JDE.INI file. Path codes are updated by the Version Control administrator when deploying an object modified by developers who are authorized to access the Server Package application, and by end users who create new batch versions to be executed on the server.

For more information about backing up and restoring, see the *Server and Workstation Administration Guide*.

See Also

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

Copying Records Between Data Sources

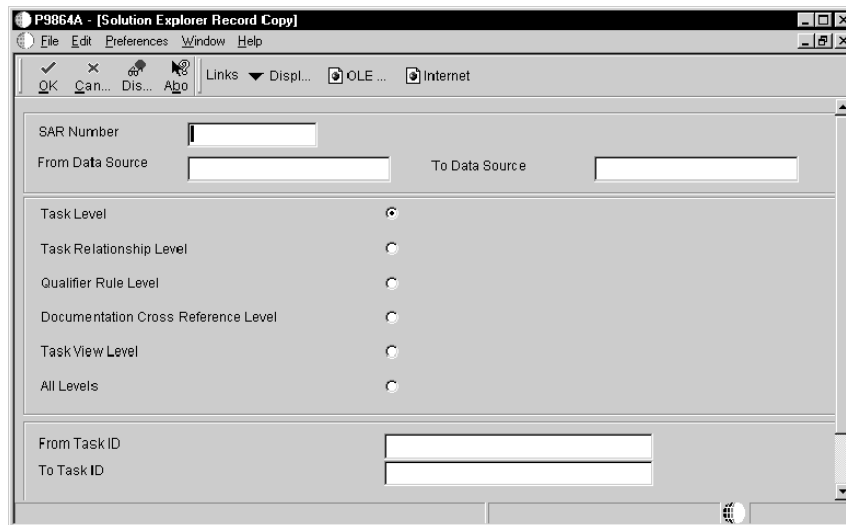
For OneWorld Xe, you use the Object Management Workbench tool to copy control table records from one data source to another. However, for Solution Explorer records you must use the Solution Explorer Record Copy application if you need, for instance, a separate set of UDCs, menus, or data dictionaries for your production and CRP environments. In this case, each change you make to the CRP environment can be copied to the production environment through the Solution Explorer Record Copy application.

This task explains how to copy control table records in Solution Explorer from one data source to another data source. You can copy the following control table records:

- Tasks
- Task Relationships
- Qualifier Rules
- Documentation Indexes
- Task Views
- All

► **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 a group of tasks you want to move. Even if the tasks are not associated with a SAR, type an arbitrary number in this field.
 - From Data Source
Type the data source you are copying the records from.
 - To Data Source
Type the data source you are copying the records to.
2. Click on one of the options below 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. OneWorld will transfer 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 associated with a parent task (task node). Complete the fields below to transfer tasks at this level:

- Parent Task -Type the parent task of the node or task you want to transfer.
- Child Task - Type a child task. If the child task is a node, OneWorld will transfer the node and all of its children.
- Task View - Type the task view for which you want to copy this task relationship. OneWorld will copy the task relationship only to the task view 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. OneWorld will transfer all Qualifier rules within the range you specify.
- Documentation Cross Reference Level
Choose Documentation Cross Reference Level to transfer documentation index records associated with task IDs. Type a range of task IDs to transfer a range of documentation indexes. OneWorld will transfer all indexes within the range you specify.
- Task View Level
Choose Task View Level to transfer all tasks, task relationships, qualifier rules, and documentation indexes associated with a task view. When you transfer these records within a given task view, OneWorld will only display those tasks within the view 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 OneWorld applications, business functions, and other objects change, you will need a means of making the changes available to users within your enterprise. You may also need to set up a new workstation with OneWorld software.

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

Topics

- ❑ Understanding packages and deployment
- ❑ Assembling packages
- ❑ Defining package builds
- ❑ Incorporating features into packages
- ❑ Viewing package build history and logs
- ❑ Understanding package INF files
- ❑ Understanding feature INF files

Understanding Packages and Deployment

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

- You want to set up a workstation for a new OneWorld 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 OneWorld. (To avoid this situation, J.D. Edwards recommends that all production users use partial packages.)
- You need to update your servers with custom modifications you've developed using the OneWorld toolset.

J.D. Edwards provides solutions to meet all of these needs. First, OneWorld provides the means to create a package describing where to find the necessary components that need to be distributed to your workstations. In order to deploy these components, you must define and build a package.

To provide maximum flexibility, there are three different package types to choose from: full, partial, and update. Partial packages are available only for workstations. The package type 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 or Package Deployment applications.

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 machines within the enterprise, a select group of machines, or individual machines. If you wish, you can schedule a package to be "pushed" from the deployment server to workstations. Push installation requires no interaction with the workstation user.

Topics

- Understanding Full and Partial Workstation Configurations

- ❑ Understanding Just-in-Time Installation
- ❑ Understanding Package Types
- ❑ Understanding Features
- ❑ Overview of Creating and Deploying a Package
- ❑ Understanding Server Packages
- ❑ Considerations for Users of Sun Microsystems, Inc. Platforms

Understanding Full and Partial Workstation Configurations

Before you begin assembling and building packages, you should have an understanding of the two types of workstation configurations available in OneWorld: full and partial. These terms refer mainly to the amount of disk space the installed package requires on the workstation. Having a good understanding of these two configurations will help 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 OneWorld and requires significantly less disk space compared to a full workstation. OneWorld 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, again depending on network traffic. However, because OneWorld adds applications to the minimum configuration as the user needs them, users with WAN access may find that performance for a partial workstation is unacceptably slow.

The main difference between the full and partial workstation configurations is the amount of specifications resident on the workstation. A full workstation configuration contains the full suite of OneWorld applications, including those you rarely or never use. The benefit of a full workstation configuration is that all applications are available immediately. The disadvantage is the disk space it requires 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 just-in-time installation performance is unacceptable on your network's configuration.

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

There are advantages to both full and partial workstation configurations, and you must evaluate your needs before determining which is best for your enterprise. The following summarizes the benefits and disadvantages of the different configurations:

Full Workstation

Advantages:

Includes the full suite of OneWorld applications (all specifications and foundations). Developers have local access to all OneWorld objects.

After initial client workstation installation, network traffic is lower than with a partial workstation configuration because all applications are installed at once.

Disadvantages:

Requires a larger amount of disk space on the workstation.

Initial installation time is much longer compared to a partial workstation configuration.

Partial Workstation

Advantages:

Requires much less disk space on the workstation compared to a full workstation configuration.

Initial installation times are much faster for a partial workstation configuration.

Only those objects required to begin using OneWorld are deployed during client workstation installation. All other applications are delivered through just-in-time installation when the user first enters an application.

Disadvantages:

Slight increase in network traffic 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, since development objects are not delivered.

Understanding Just-in-Time Installation

A partial workstation doesn't have any applications resident. Instead, applications are retrieved at runtime and loaded on the workstation the first time a user selects that application from the OneWorld Explorer menu. Loading happens only once; the next time the user selects the same application, it will still be 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 there are no applications 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 make a menu selection from OneWorld Explorer, the runtime engine checks to see if application specifications reside on your workstation. If local specifications do not exist (and the Environment Master just-in-time installation flag is set to Y, and your security profile allows the application to be installed), the specifications are transferred from the central objects data source for the path code associated with your environment. In this way, needed applications are installed "just-in-time" rather than when the user receives the package or at initial installation time.

The process for an update package works almost the same way. When you receive an update package that contains a changed application without the new specifications, OneWorld first checks to see if that application's specifications are resident on your workstation. If so, it deletes from your workstation old versions of that application. Then, the next time you select that application from the OneWorld Explorer menu, OneWorld 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 (comes 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 the package is based on. A full package contains everything developers need to run and develop in OneWorld. Specifically, this 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 users have every OneWorld application available for which they are licensed. Because specifications reside on the workstation, information is handled 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. A few specifications and tables may still be installed just-in-time when you deploy a full package, but the impact on network performance is insignificant.

Full packages are primarily for initial OneWorld installations and are normally deployed via the Client Workstation Installation application. You can also use Package Deployment to install a full package on a machine that already has OneWorld 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 will be given the option to do so at deployment time.

Partial Packages

Partial packages provide a minimum configuration of OneWorld. 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

OneWorld Explorer. Specifically, a partial package starts with an empty set of TAM files, and then adds the specifications for the objects that are included in the package. Also included in a partial package are a full set of DLLs and an INF file that defines where the foundation, data, help files, and package objects are located.

Applications are loaded through just-in-time installation the first time the user selects an application from the OneWorld 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 OneWorld installations and are normally deployed via the Client Workstation Installation application, although you can use Package Deployment to install a partial package on a machine that already has OneWorld 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, foundation, or help files. This package type is well-suited for quickly deploying software changes and fixes.

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 the applications are pulled directly from the central objects data source at the moment the application is first selected.

When a user receives an update package, OneWorld loads on the user's workstation all objects in the package after the user signs on. The objects in an update package replace those same objects on the workstation. All other objects on the workstation are left untouched.

Applications in a package can be secured through Security Workbench, which checks to determine if the user has permission to install an application. Only users authorized to load the application are able to do so. Therefore, your update package can contain a varying assortment of applications going out 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 will be given 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. Users on a WAN may find that performance is better if they include an application's specifications in the package.

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 parent package's objects, and new DLLs are created. Similarly, specifications from the update package are merged into the parent package's specifications.

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

Understanding Features

In addition to OneWorld 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 a OneWorld application or other OneWorld function. Like OneWorld objects, features are built into 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 Microsoft Access 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 Feature Based Deployment Director. You can then add the feature to a package by using the Package Assembly Director and Package Build Definition Director.

For more information about adding features to a package, see *Incorporating Features Into Packages*.

Overview of Creating and Deploying a Package

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

3. Assemble the package.

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

The process of assembling the package is simplified by the Package Assembly Director, which displays a series of forms that guide you through the steps of naming your package and assembling the objects you want to include in the package.

See [Assembling Packages](#) in this section for details on how to assemble a package.

4. 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, 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 Definition Director 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.

See [Defining Package Builds](#) in this section for information on how to define a package build.

5. Build the package.

During the actual build process, OneWorld takes the information you provided when you assembled and defined the package, and copies and converts central objects to the package. It also performs a global build of the business functions included in the package, and then compresses the package.

6. 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 OneWorld), you must specify the

date and time to deploy the package. When you schedule the package you can make package installation 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.

See [Deploying Packages](#) for information on how to schedule a package. For information on push installation, see [Using Push Installation](#).

7. Deploy the package to deployment and enterprise servers.

Use Package Deployment to move any changed objects to the enterprise server.

If you specify a server during the package build definition process, OneWorld 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 [Deploying Server Packages](#) for information.

How OneWorld Builds a Full Package

Following is an overview of how OneWorld builds a full package:

8. Creates the package build directories.
9. Creates the INF file.
10. 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, OneWorld does not copy bin32, lib32, and object (.obj) files, because the BusBuild program creates them.

11. Builds the TAM specification files from the information in the relational database.
12. Runs the BusBuild function 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.
13. Compresses the directories.

How OneWorld Builds a Partial Package

Following is an overview of how OneWorld builds a partial package:

14. Creates the package build directories.
15. Creates the INF file.
16. Builds an empty set of TAM specification files.
17. 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, OneWorld does not copy bin32 and lib32 files, because the BusBuild program creates them.

18. Builds the specifications for the objects in the *LITE record.
19. Runs the BusBuild function 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 this function separately.

20. Compresses the directories. (Optional)

How OneWorld Builds an Update Package

Following is an overview of how OneWorld builds an update package.

21. Creates the package build directories.
22. Creates the INF file.
23. For each object in the F9631 table, gets the information out of the relational database and adds it to the TAM specification files.
24. Runs the BusBuild function 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 OneWorld 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. OneWorld allows you to partition business applications to an enterprise server. In order to ensure that modifications and enhancements developed on the workstation are reflected on the server, you must build a server package containing those modifications and enhancements.

Through the same Package Assembly and Package Build applications 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 application.

In a development environment, developers can create server packages whenever they create or modify an object and there is a need for that object to be on the enterprise server. In a production environment, a OneWorld 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 OneWorld help information or a Microsoft Access database.
- Foundation code is not typically deployed as part of a server package.
- Some sets of TAM (table access management) 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.

There are three reasons to create a separate server package that corresponds to the workstation package:

- Business functions are not compatible across platforms. For example, a business function compiled on a Windows NT workstation will not run on a UNIX server. Also, not all business functions are compiled on the server.
- Data alignment is different across platforms. TAM files built on the workstation won't run on the server unless you convert them. Also, integer representation (the way numbers are recognized and identified) is different on servers than on the workstation.
- ASCII to EBCDIC conversion must take place for the AS/400.

In addition, there are differences between server platforms. 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, the purpose for creating server packages is different. The main purpose of the server package build process is to enable OneWorld administrators to build TAM specifications and business functions on the enterprise servers.

The Package Build application provides the following benefits for building server package builds:

- Provides complete integration with client workstation package builds
- Allows administrators to build a package on multiple servers simultaneously

- Enables administrators to build individual package components simultaneously on the server
- Supports the ability 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 the administrator is able to determine what has been built
- Creates compressed files and loads them onto the deployment server for easier mastering to CD
- Supports both full and update packages
- Provides restart capabilities for builds that do not complete 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 Package Assembly) and build the server package (using Package Build) at the same time you assemble and build client workstation packages.

The following describes what actually happens during the build process after you have assembled the server package and defined the package build:

1. After the build process starts, OneWorld creates packed TAM files.
2. OneWorld submits a server package batch application locally.
3. This batch application calls a business function, which in turn calls the server package engine.
4. The build engine uses records created by the Package Assembly Director and Package Build Director 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 F9860 table 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 F9860 table is not used.
- Start a build master process on the server when all business function source is transferred. This build master starts one or several individual build processes simultaneously. Each DLL has its own build process. The number of processes that run simultaneously is determined by a JDE.INI file setting.
- 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 put the compressed files on the deployment server when building is complete. This happens only if compression was specified when the package was built through the Package Build application. 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/jdedwardsoneworld/b7333/packages	The location on the server where the package will be built.
Optimization Flags=	+02 (default for HP 9000)	Machine dependent. These compile flags

	-O2 (default for RS/6000 and Sun)	are used when building business functions in "Release" mode. You should not change these flags.
DebugFlags=	-g -y -D_DEBUG _DJDEDEDEBUG (default for HP 9000) -g -qfulpath -qdbextra -D_DEBUG -DJDEDEDEBUG (default for RS/6000) -g -D_DEBUG -DJDEDEDEBUG (default for Sun)	Machine dependent. These compile flags are used when building business functions in "Debug" mode. You should not change these flags.
InliningFlags=	blank (default)	Yes turns on inlining on the AS/400. No turns it off. This entry 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)	
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)	Machine dependent. Valid compiler flags. The spill flag sets the stack space when business functions are compiled. J.D. Edwards has found that 1024 is adequate to compile the delivered business functions.

OSReleaseLevel=	+DAportable -q32 (for AIX)	Release level you are compiling to. 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 -bloodmap:loadmap (default for RS/6000) -G -L\$(ORACLE_HOME)/lib (default for Sun)	Machine dependent. These flags are used when linking business functions. You should not change these flags.
LinkLibraries=	blank (default)	Libraries to which business functions are linked. (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>	(AS400 only) Specify a queue name if you want the jobs for building dlls to go to a different Queue besides the default.

See Also

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

JDE.INI Settings for Workstations

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

Building Specs and Business Functions

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

```
WaitForBusbuild=Y
```

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

Compressing Server Packages

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

```
DoCompression=1
```

Note: This setting is only valid for SP15 and later service packs. Contact Customer Support if you want to compress server packages on systems running earlier service packs.

This setting compresses packages 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 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 (\bin32, \specs, etc.) under 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 that includes the information necessary to deploy the compressed files. The process creates a different type of compressed file for each type of server:

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

When you deploy the package to another enterprise server, OneWorld 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 Sun Operating Systems

If you develop custom modifications on Sun servers that are running the Solaris operating system, there is one important step you might need to take before assembling, defining, and building packages that contain your modifications. This step ensures that the package build process completes successfully.

The Sun Solaris compiler expects a newline character at the end of every source code file 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 could cause the package build to fail.

To ensure that this does not happen, before you assemble the package that will contain your custom modifications, you must ensure that this newline character is present in the compiled source code.

There are some cases where the newline character is automatically added, and you will not need to take this extra step. 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, J.D. Edwards has added a process to ensure that the newline character is added to all business function source code files shipped with OneWorld.

However, if you edit source code on a PC workstation and then transfer the file to the server for compiling, be aware that PC workstation editors typically do not add the newline character. In this situation you should check your source code and verify that the newline character exists.

Assembling Packages

The first step in building and deploying a package is to assemble the package, which entails selecting the type of package you want to build, providing a package name and detailed description, and assembling the objects you want to include in the package. This same package name and description appear during Workstation Installation when the user chooses a package to install.

This package assembly process is greatly simplified by the Package Assembly Director, which steps you through the process. During package assembly, the status will always be either In Definition or Definition Complete. Once package assembly definition is complete you can then define the package build as explained in [Defining Package Builds](#).

Topics

- Understanding the Package Assembly Director
- Assembling a package
- Activating an assembled package
- Revising an existing package
- Copying a package
- Deleting a package

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. In the Director, you always have the option of going to the previous or next screen. Also, you can always cancel the assembly process.

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

Welcome form	Provides a description of the Package Assembly Director.
Package Information form	Enter the package name, description, and corresponding path code on this form.

Package Type Selection form

Indicate on this form 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 also have the option of including 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 a user selects that application. That is, the application specifications are pulled dynamically from the central objects data source the first time a user selects the application after receiving the package.

Foundation Component form

Enter a foundation location on this form. A OneWorld foundation is the code required to run all OneWorld applications. It is required for all full and partial packages. If you do not specify a foundation path for full and partial packages, the default foundation path will be used. Update packages use the parent package's foundation unless one is specified.

Help Component form

Enter a help location on this form. The help component determines where the system looks for the help files associated with the package. For full and partial packages, if you do not specify a help location, the default help file path will be used. Update packages do not require help files.

Including a help component in a package does not mean helps are installed on the user workstation. The help component tells OneWorld where to find the help files that are associated with the package.

Database Component form

Specify on this form 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 default database path will be used. 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 objects included in the package will be pulled from the deployment data source that is associated with the path code you have specified for the package.

Object Component form (For partial and update packages only)

Specify on this form the individual objects 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 OneWorld application or other OneWorld functionality.

**Language
Component form**

Use this form to include in your package language specifications for a language other than English.

**Package
Component
Revisions form**

Review your selections and default values for the Foundation, Help, Database, Objects, Features, and Language 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 a default value and, after making sure you want to use the displayed 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 associated with the package.
- Help: The default help location is the server share path under the path code associated with the package.
- Database: The default database location is the server share path under the path code associated with 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

Package Component Revisions form also has a Set Default option accessible from the Form menu that accomplishes the same thing.

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, you will have an empty package.

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

Assembling a Package

This section describes how to assemble a package using the Package Assembly Director.

Topics

- Running the Package Assembly Director
- Reviewing the Package Assembly Selections
- Entering a Foundation Location
- Entering a Help File Location
- Entering a Database Location
- Adding Objects to a Package
- Adding Features to a Package

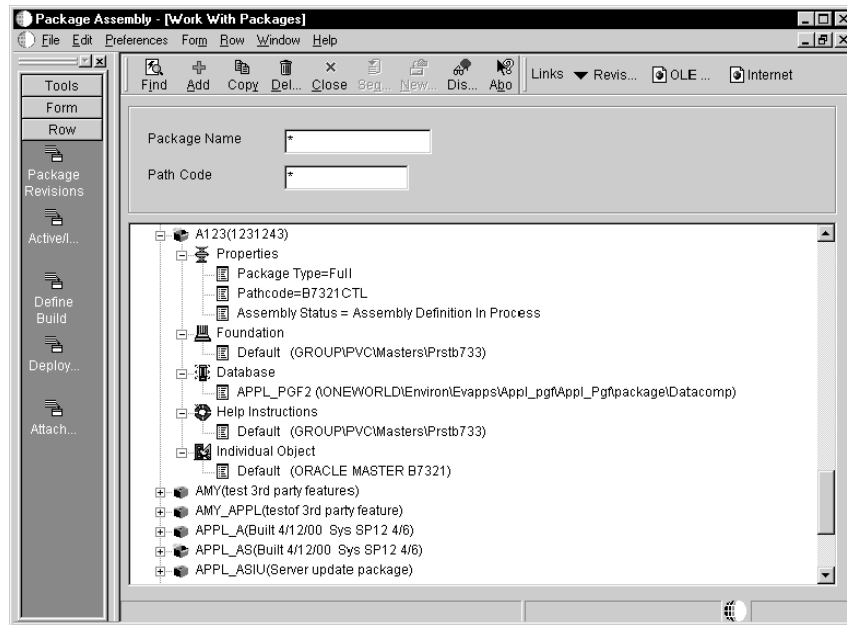
Running the Package Assembly Director

Complete the following task to enter default information into each of the main forms of the Package Assembly Director. For details on 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 (P9601).

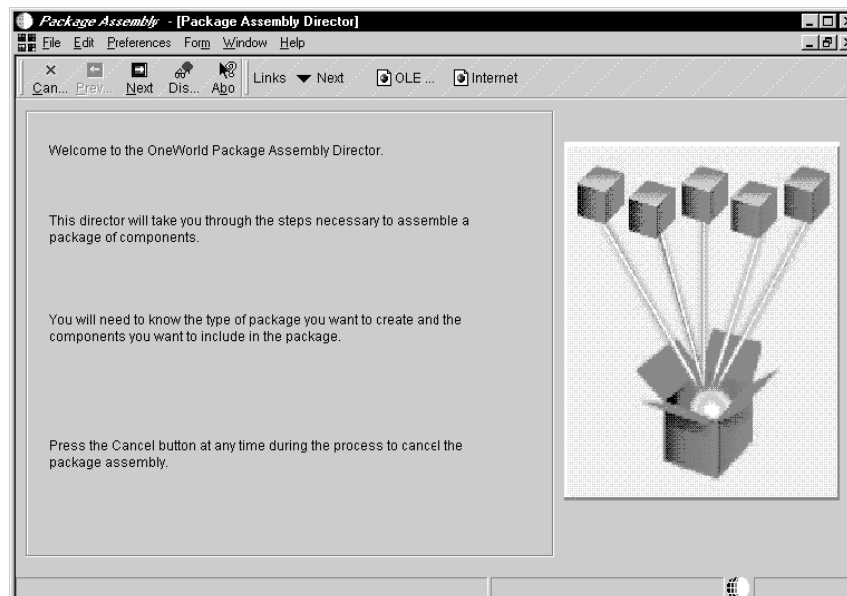
The Work with Packages form appears.



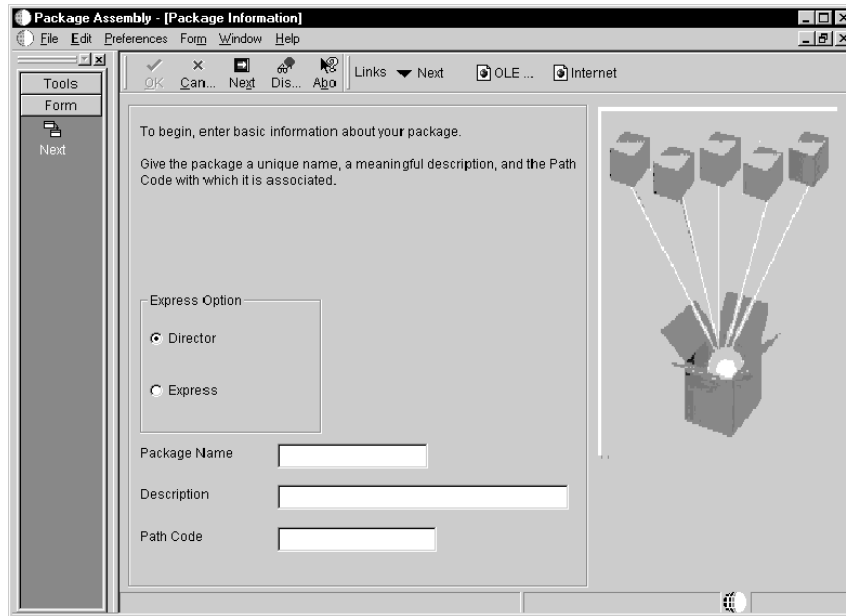
Any previously assembled packages appear on this form. As on any other parent/child form, you can expand or compress by clicking the plus (+) or minus (-) symbols to show more or less information.

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.

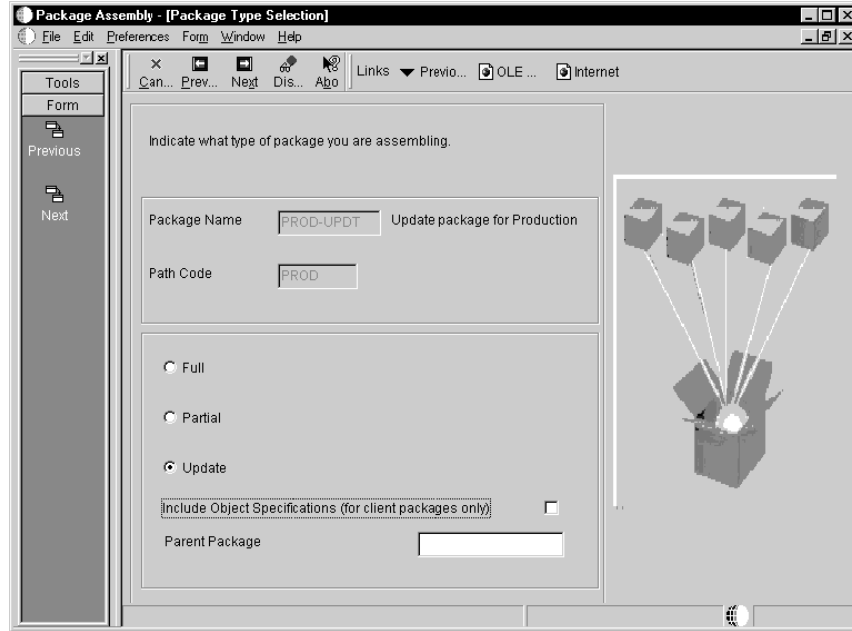
1. To assemble a new package, click Add.



2. From the Welcome screen of the Package Assembly Director, click Next.



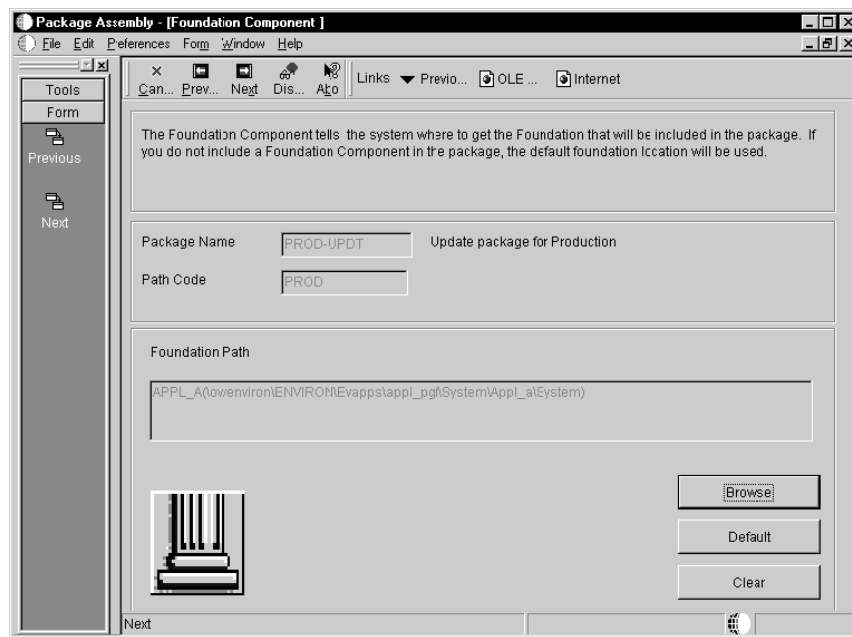
3. On Package Information, complete the following fields:
 - Package Name
 - Description
 - Path Code
4. Choose one of the following options:
 - Director
To customize the package assembly.
 - Express
To accept default values for the package assembly. Continue with the following task, Reviewing the Package Assembly Selections.
5. Click Next.



6. On Package Type Selection, complete the following fields:

- Full, Partial, or Update
- Parent Package
(for update packages only)
- Include Object Specifications (for client packages only)

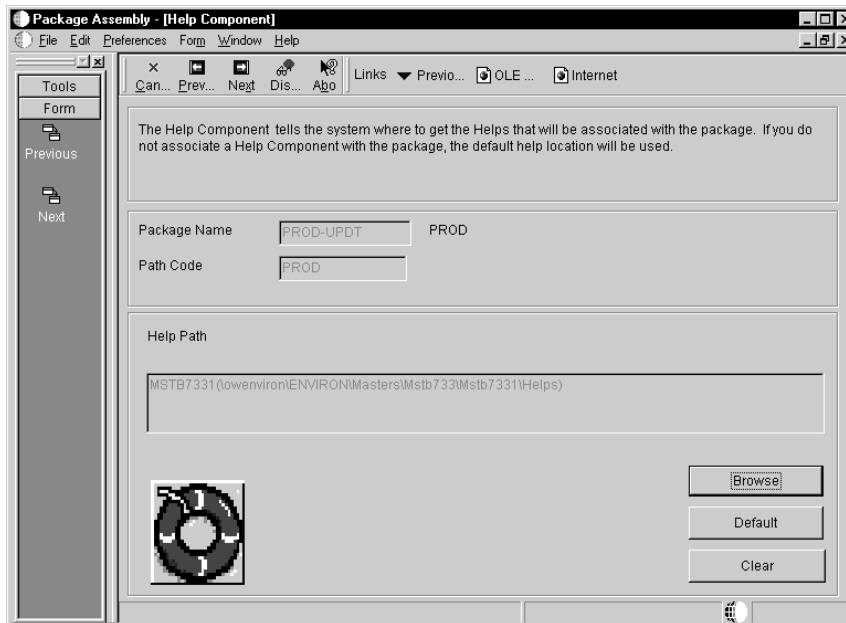
7. Click Next.



8. On Foundation Component, accept the default location by clicking Next, or click Browse to specify another foundation location. To clear the location, if any, click Clear. The Clear option applies only to update packages.

For more information about entering a foundation, see [Entering a Foundation Location](#).

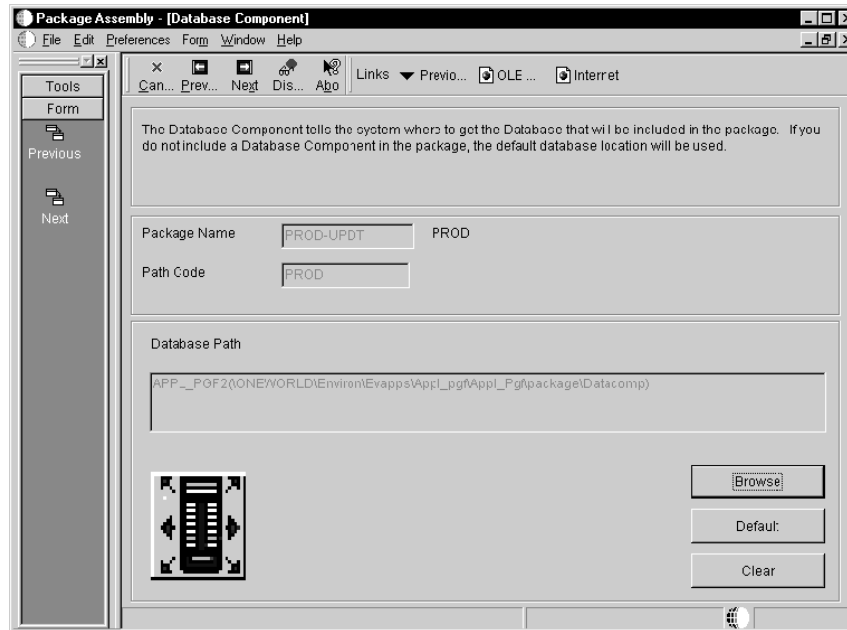
9. Click Next.



10. On Help Component, accept the displayed default location by clicking Next, or click Browse to specify another help location. To clear the currently displayed location, if any, click Clear. The Clear option applies only to update packages.

For more information about entering a help location, see [Entering a Help File Location](#).

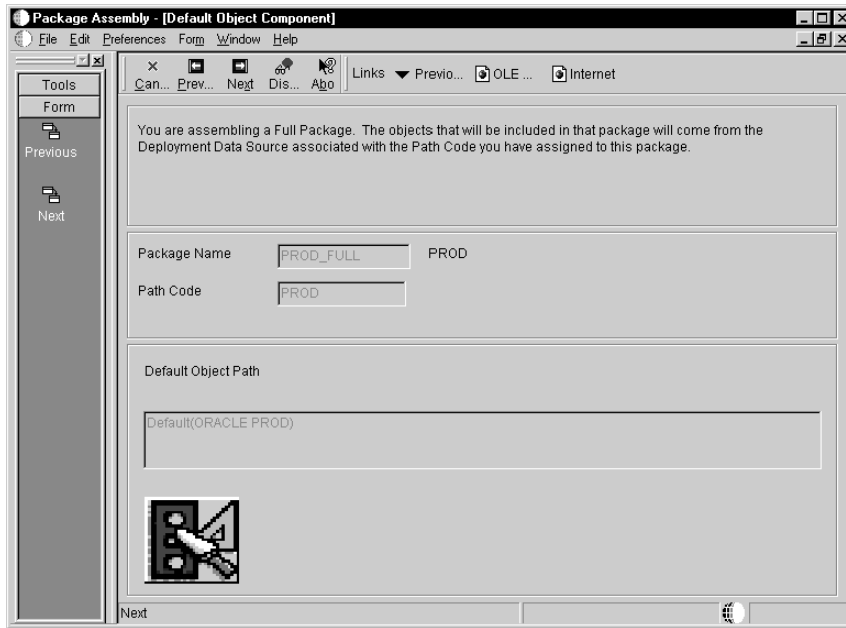
11. Click Next.



12. On Database Component, accept the default location by clicking Next, or click Browse to specify another database location. To clear the currently displayed location, if any, click Clear. The Clear option applies only to update packages.

For more information about entering a database location, see [Entering a Database Location](#).

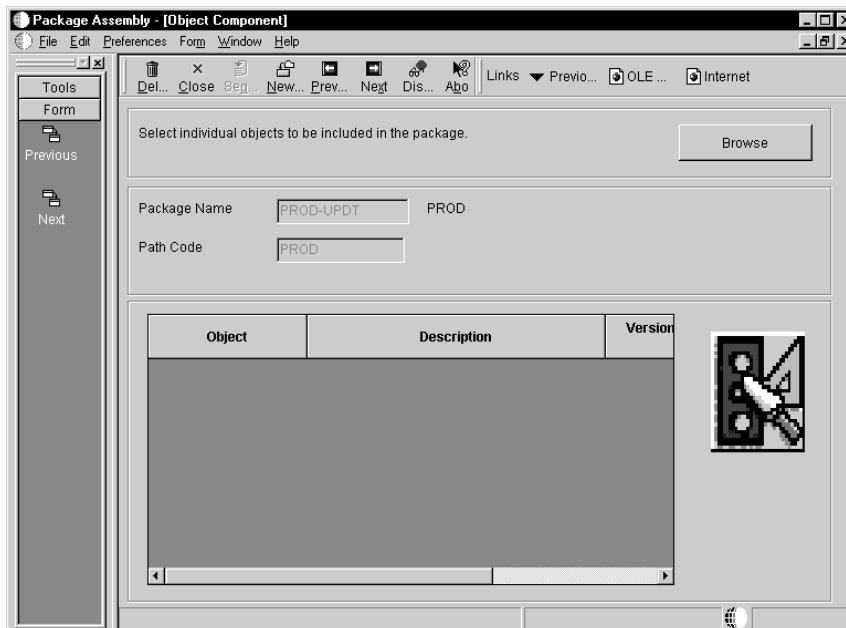
13. Complete one of the following:
- If you are assembling a full package, click Next.
 - If you are assembling a partial or update package, proceed to the next step.



On Default Object Component, OneWorld will build your package from the deployment data source associated with the displayed default object path. Verify that the correct location is displayed. Because you are assembling a full package, skip the following step.

14. On Database Component, click Next.

When you are assembling a partial or update package, the Object Component form appears. This form allows you to specify the individual objects you want to include in your package. When you revise a previously-assembled package, objects you added earlier are displayed.

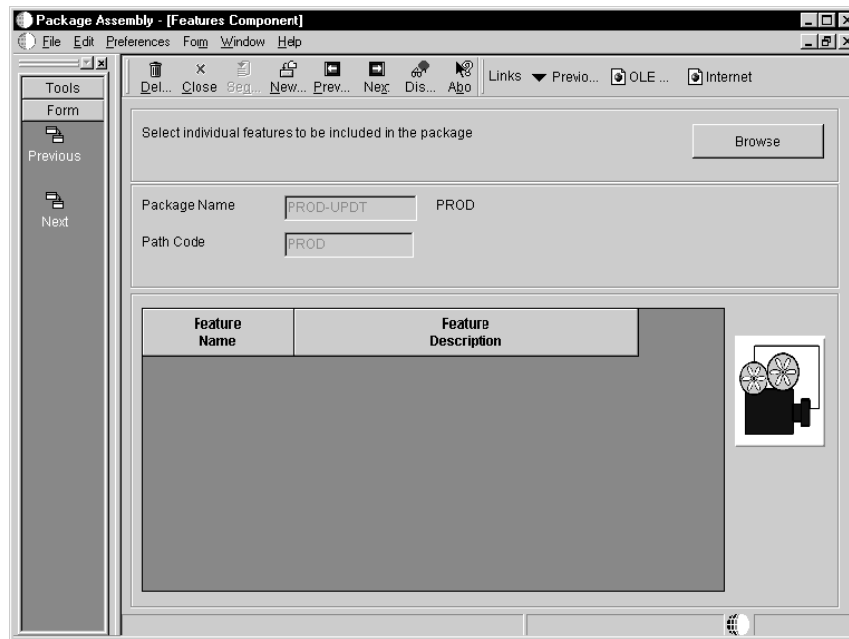


- To add an object, click Browse to display the Object Component Selection form. Use this form to locate and select the objects you want to include in your package. When you are finished adding objects, click Close to return to the Object Component form.

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

- Click Next.

On Features Component, you can specify the features you want to include in your package. When you revise a previously-assembled package, features you added earlier are displayed.



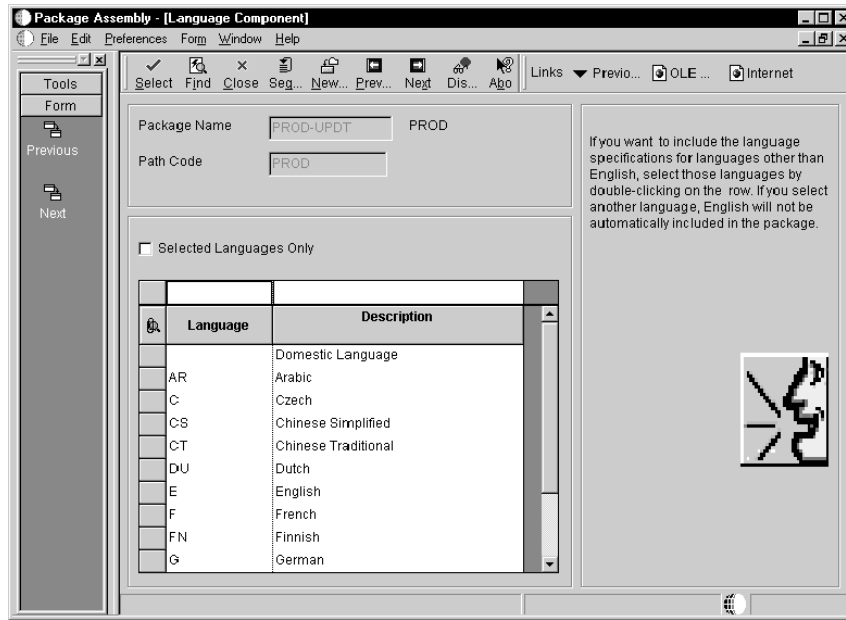
- To add a feature, click Browse to display the Feature Component Selection form.
- On the Feature Component Selection form, click find to display a list of features, then choose a feature and click the Select button to add the features you want to include in your package.

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

- Click Close.

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

- On Feature Component, click Next.



21. On Language Component, if you to display only the languages currently selected for the package, check the Selected Languages Only options. This applies to situations where you are revising a previously assembled package, not to when you are assembling a package for the first time.
22. Double-click the gray box next to the languages you want to add to your package language specifications.

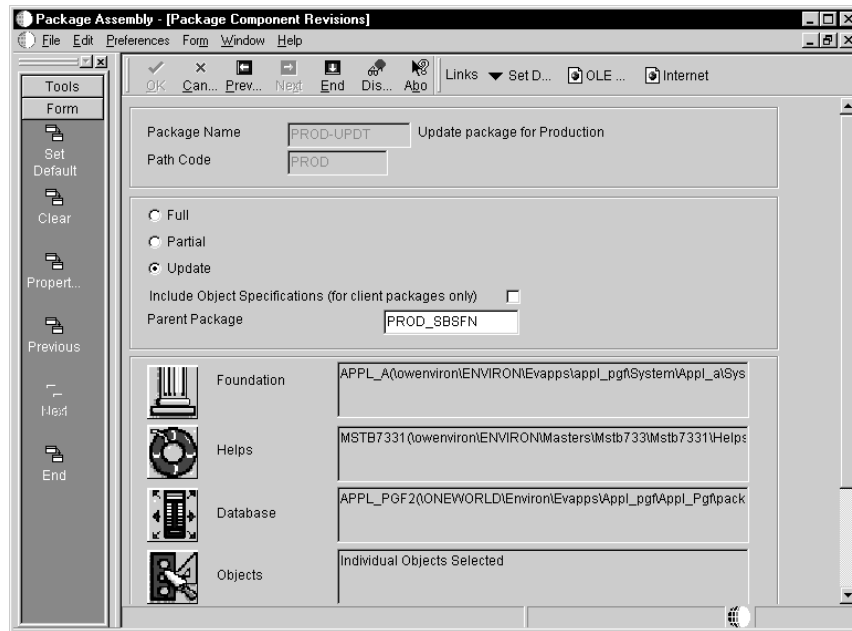
If you add a language to your package, only that language will be included. For example, if you add French, English will not be included even though it is the default language. If you want two languages (such as French and English) you must include them both in your package. For more information about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

23. 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 any language selection.

► To review the package assembly selections



1. On the Package Component Revisions, you can change any of the package components by clicking the icon for the component 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*, which immediately follows the tasks of customizing the main Package Assembly Director screens.

Field Descriptions

Field	Description
Description	A description of the package.
Path Code	The path code is a pointer to a set of OneWorld objects, and is used to keep track of sets of objects and their locations within OneWorld.
Include Object Specifications	When you include individual objects in an update package, you have the option of including a corresponding set of specifications for those objects. When you

Field	Description
	include specifications, a "snapshot" of the specifications will be included in the package after it is built. When the package is deployed, the package recipient receives those object specifications. If you do not include object specifications in the package, the old specifications for the objects in the package will be deleted from the workstation when the package is deployed. Then, the next time the package recipient attempts to use the object, a new set of specifications for it will be transferred to the workstation via just-in-time installation.
Parent Package	Since an update package includes only a subset of objects, you must indicate the parent package on which the update package is based or related to. This information is used by the system to determine how to build business functions.
Selected Languages Only	When this option is turned on, only languages that have been selected for inclusion in the package will appear in the detail area.
Language	A list of valid codes for a specific user defined code list.

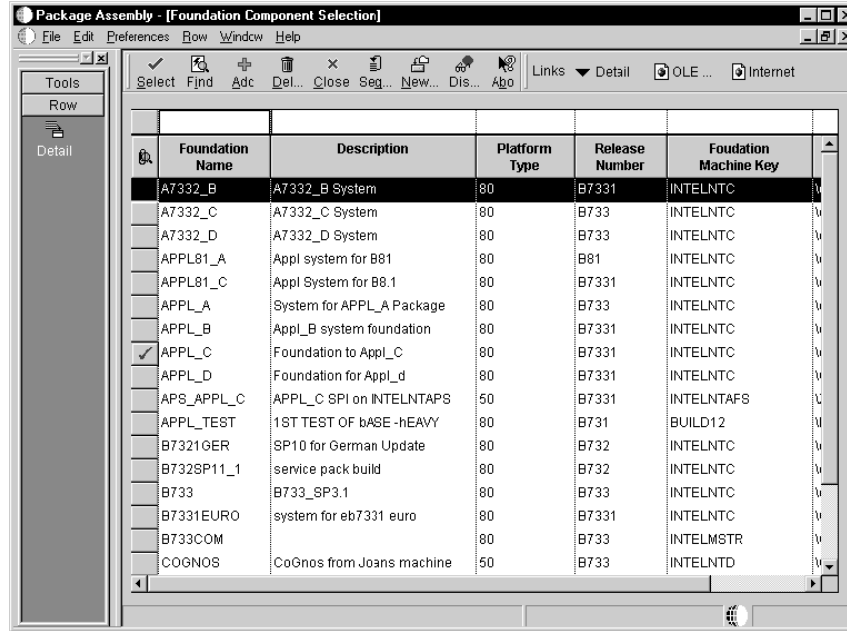
Entering a Foundation Location

The default foundation location is determined through the release associated with the package's path code. This is normally the system directory at the same directory level as your path code.

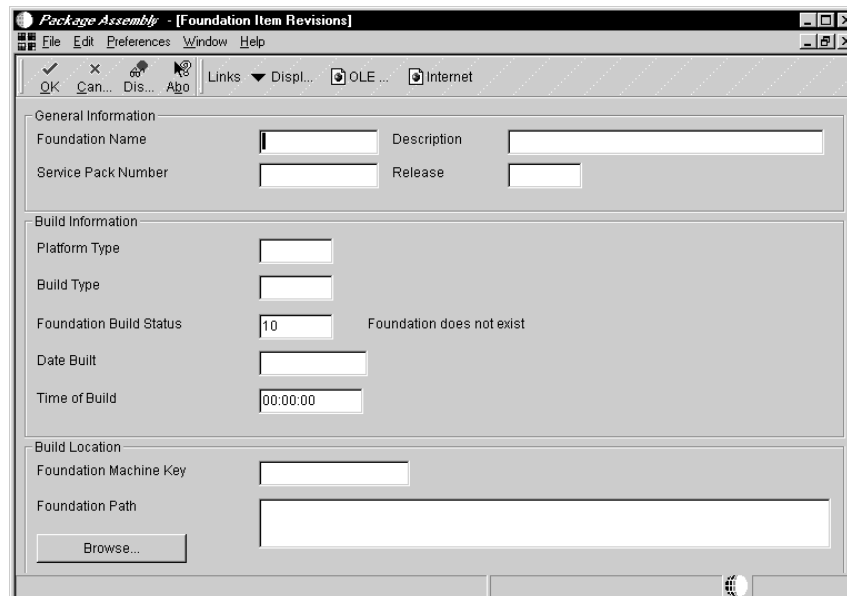
The following task describes how to change the foundation location from the default location displayed on the Foundation Component form.

► 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.



- Find and choose the foundation you want to use in your package. If you want to use a new foundation, see the following step.
- To add a new foundation, click Add. The Foundation Item Revisions form appears.



- 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

Field Descriptions

Field	Description
Foundation Name	A OneWorld foundation is the code required to run all OneWorld applications. A foundation ID is required for all Full and Partial packages. For full and partial packages, if you do not enter a foundation line, the default foundation will be used. The default foundation is determined through the release associated with the path code for the package. This is normally the system directory at the same directory level as your path code. The foundation must be compressed when built.
Service Pack Number	A service pack is an update to the foundation code that is delivered between major releases and cumulative releases of the OneWorld software.
Release	For World, the release number as defined in the Software Versions Repository. For OneWorld, the release number as defined in the Release Master. FORM SPECIFIC..... On this form, Release Number is the OneWorld major release with which this foundation is associated.

Field	Description
Platform Type	Host Machine Type.
Build Type	Indicate the compiler configuration to use for the software build.
Foundation Build Status	Describes the current status of the build process for OneWorld foundation.
Date Built	The date the software build completed.
Time of Build	The time at which the software build completed.
Foundation Machine Key	<p>For World, the Location indicates the machine (server or client). For OneWorld, the Location or Machine Key indicates the name of the machine on the network (server or workstation).</p> <p>.....FORM SPECIFIC.....</p> <p>On this form, the Machine Key is the name of the deployment server where your custom foundation resides.</p>
Foundation Path	<p>For World, the Server Share Path Field is used by the environment to determine the location of the current server. For OneWorld, this field indicates the shared directory for this Path Code. The objects that are stored on a file server will be found in this path.</p> <p>.....FORM SPECIFIC.....</p> <p>On this form, enter the exact path from which this item should be copied. All files in the last directory specified will be included in the package. Source Machine Key and Source are used together to define the item's location.</p>

Entering a Help File Location

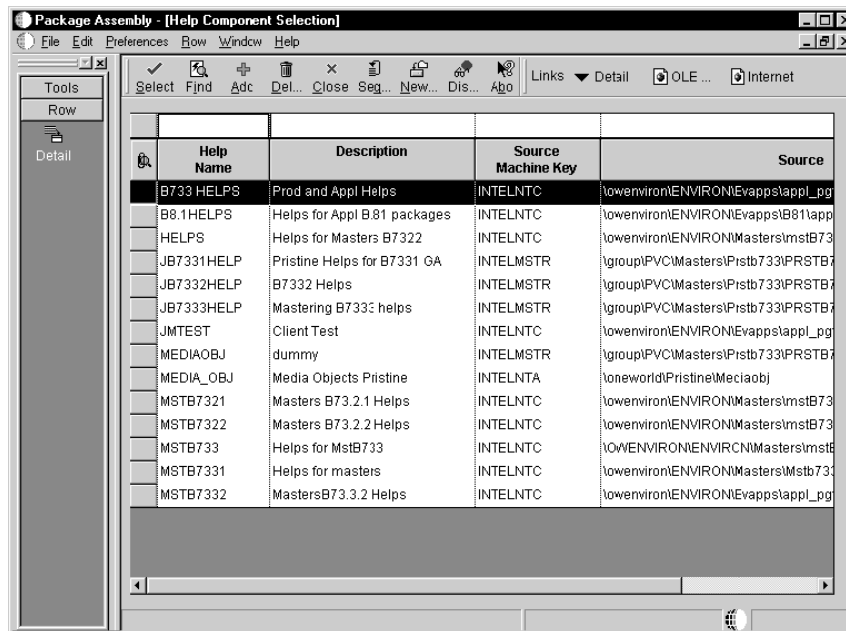
For all package types, if there is no help location, the package build creates an INF that has help mapped to the default location as defined in Release Master (one set of help per release). If a help file location is defined for a package, the package build copies the help file from the location defined in Help Item Revisions to the package directory.

The package INF will indicate that help files are located in the package itself, instead of the default location. Regardless, 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.

The following task describes how to change the help location from the default location displayed on the Help Component form.

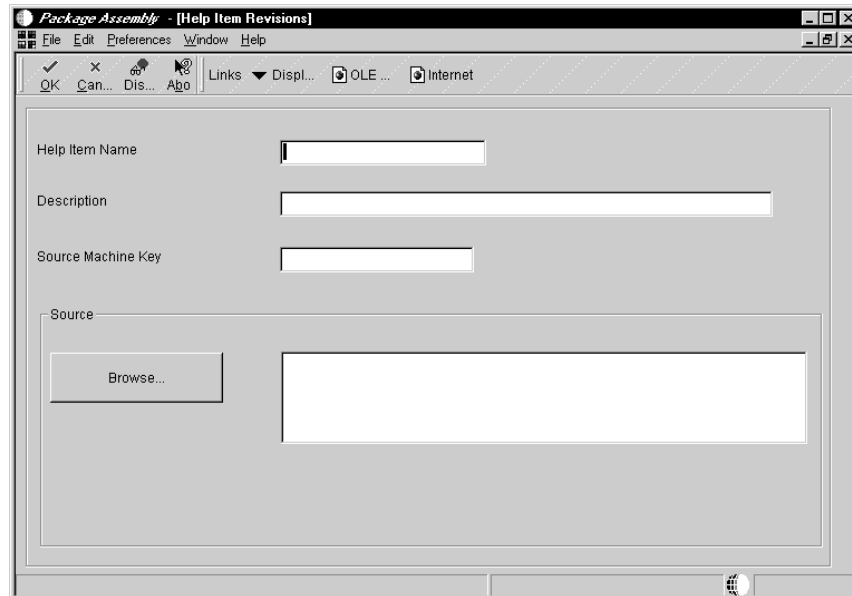
► To enter a help location

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. Find and choose the help location you want to use in your package. If you want to use a new location, see the following step.

3. To add a new help location, click Add.



4. On Help Item Revisions, complete the following fields.

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

5. Click OK:

Field Descriptions

Field	Description
Help Item Name	A OneWorld foundation is the code required to run all OneWorld applications. A foundation ID is required for all Full and Partial packages. For full and partial packages, if you do not enter a foundation line, the default foundation will be used. The default foundation is determined through the release associated with the path code for the package. This is normally the system directory at the same directory level as your path code. The foundation must be compressed when built.
Server Share Path	For World, the Server Share Path Field is used by the environment to determine the location of the current server.

Field	Description
	For OneWorld, this field indicates the shared directory for this Path Code. The objects that are stored on a file server will be found in this path.
Source	For World, the Server Share Path Field is used by the environment to determine the location of the current server. For OneWorld, this field indicates the shared directory for this Path Code. The objects that are stored on a file server will be found in this path.

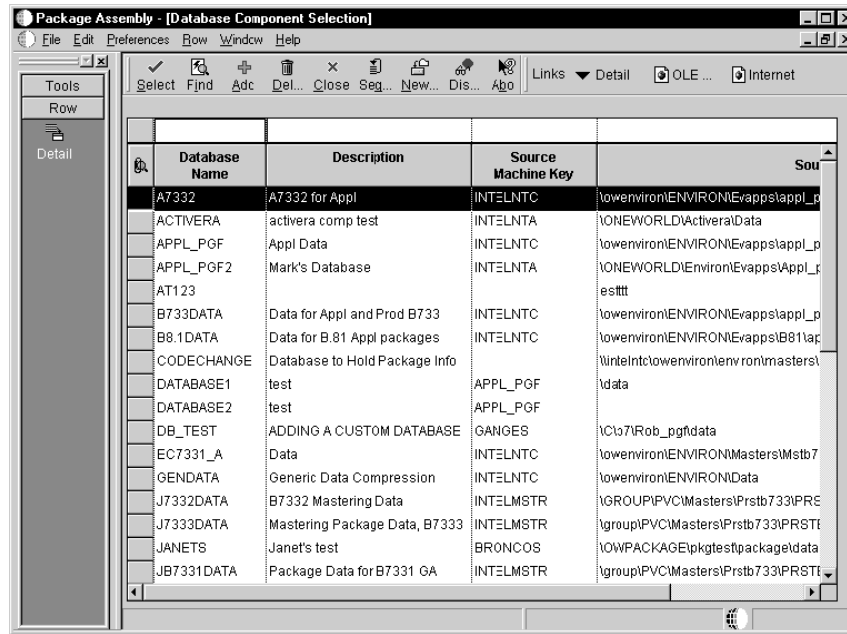
Entering a Database Location

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

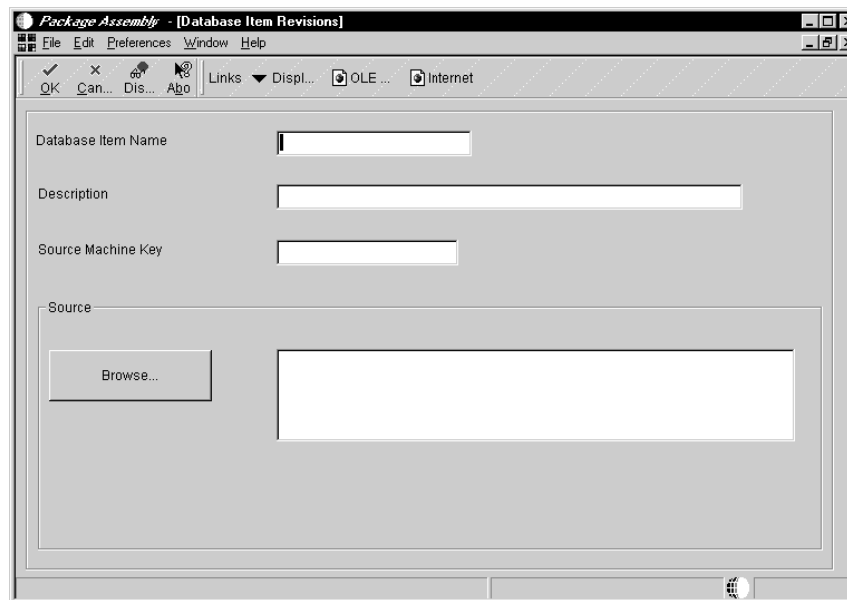
The following task describes how to change the database location from the default location displayed on the Database Component form.

► **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.



- Find and choose the database location you want to use in your package. If you want to use a new location, see the following step.
- To add a database, click Add.



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

- Source
5. Click OK:

Field Descriptions

Field	Description
Database Item Name	A OneWorld foundation is the code required to run all OneWorld applications. A foundation ID is required for all Full and Partial packages. For full and partial packages, if you do not enter a foundation line, the default foundation will be used. The default foundation is determined through the release associated with the path code for the package. This is normally the system directory at the same directory level as your path code. The foundation must be compressed when built.
Source Machine Key	For World, the Location indicates the machine (server or client). For OneWorld, the Location or Machine Key indicates the name of the machine on the network (server or workstation).
Server Share Path	For World, the Server Share Path Field is used by the environment to determine the location of the current server. For OneWorld, this field indicates the shared directory for this Path Code. The objects that are stored on a file server will be found in this path.

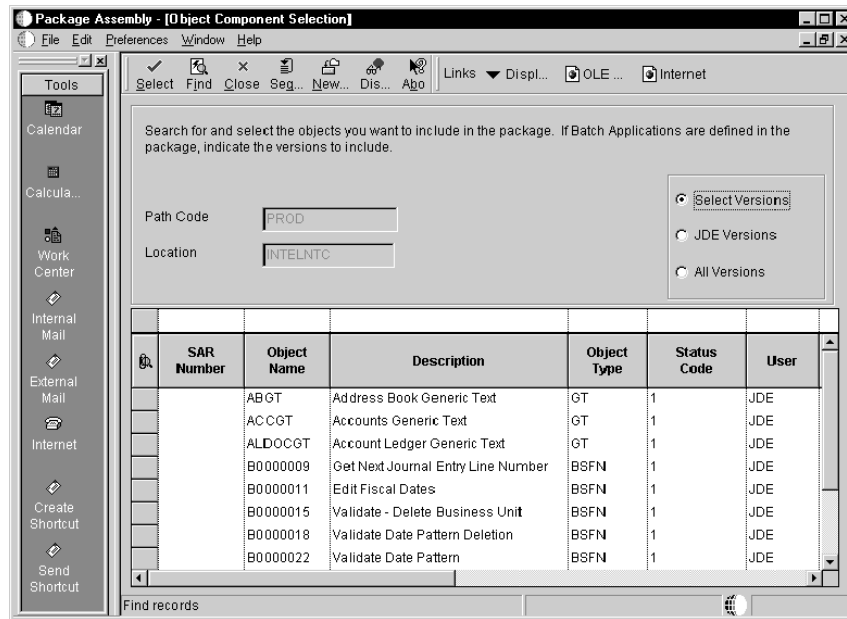
Adding Objects to a Package

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

The following 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.

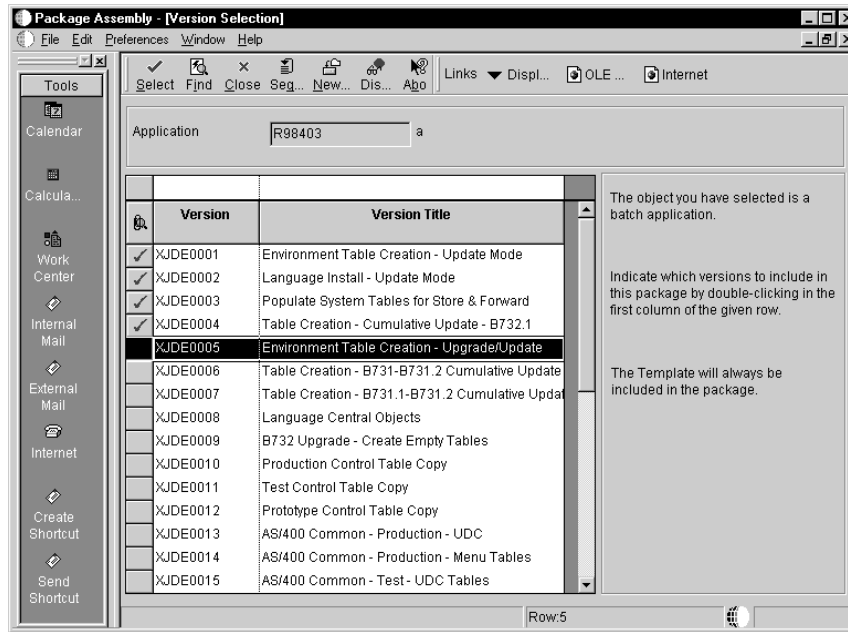
► To add objects

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. Find and choose the object you want to include in the package.
3. If you are adding a batch application, click one of the following options:
 - Select Versions- selects only specified versions of the object
 - JDE Versions - selects all JDE versions of the object
 - All Versions - selects all versions of the object
4. Click Select.

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



5. Double-click on the specific versions in the list that you want to include in the package. This action displays a check mark to the left of each version you select.

When you are finished 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. When you are finished, click Close to return to the form from which you accessed the Object Component Selection form.

Field Descriptions

Field	Description
SAR Number	An abbreviation for Software Action Request (SAR). This number is used by the J.D. Edwards project control system to keep track of all charges and activity by individual SAR number.
Object Name	The OneWorld architecture is object-based. This means that discrete software objects are the building blocks for all applications, and that developers can reuse the objects in multiple applications. Each object is tracked by the Object Librarian. Examples of OneWorld objects include: <ul style="list-style-type: none"> o Batch Applications (such as reports) o Interactive Applications o Business Views o Business Functions

Field	Description
	<ul style="list-style-type: none"> o Business Functions Data Structures o Event Rules o Media Object Data Structures
Description	The description of a record in the Software Versions Repository file. The member description is consistent with the base member description.
Object Type	The type of object with which you are working. For example, if you are working with tables the object type is TBLE, or business functions is BSFN.
Status Code	This code determines the status of the software in the development cycle.
User	For World, the IBM-defined user profile. For OneWorld, the identification code for a user profile.
Merge Option	The Merge Option denotes whether a customer's OneWorld object will be merged in with the J.D. Edwards OneWorld object. The Merge Option can be set at the path code level so that all objects checked into that path will carry the same Merge Option as the path code.
Date Modified	The date that the object, such as a DREAM Writer version, Software Versions Repository Record, and so on, was last modified.
Time Modified	The time the object was last checked in.
System Code	A user defined code (98/SY) that identifies a J.D. Edwards system.

Field	Description
Reporting System	A code that designates the system number for reporting and jargon purposes. See UDC 98/SY.
All Versions	The name given to identify a version of the software.
JDE Versions	The name given to identify a version of the software.
Version	A user-defined set of specifications that control how applications and reports run. You use versions to group and save a set of user-defined processing option values and data selection and sequencing options. Interactive versions are associated with applications (usually as a menu selection). Batch versions are associated with batch jobs or reports. To run a batch process, you must choose a version.
Version Title	A description of the version that appears next to the version number. The version title is different from the report title. This field should describe the use of a version. For example, an application for generating pick slips might have a version called Pick Slips - Accounting and another version called Pick Slips - Inventory Management.

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 OneWorld application or other OneWorld functionality. Like OneWorld objects, features are built into a package and deployed to the workstations and servers that require the feature components.

When you are finished adding features to the package, those features appear on the Package Assembly Director's Feature Component form, the Package Component Revision form, and the Work with Packages form.

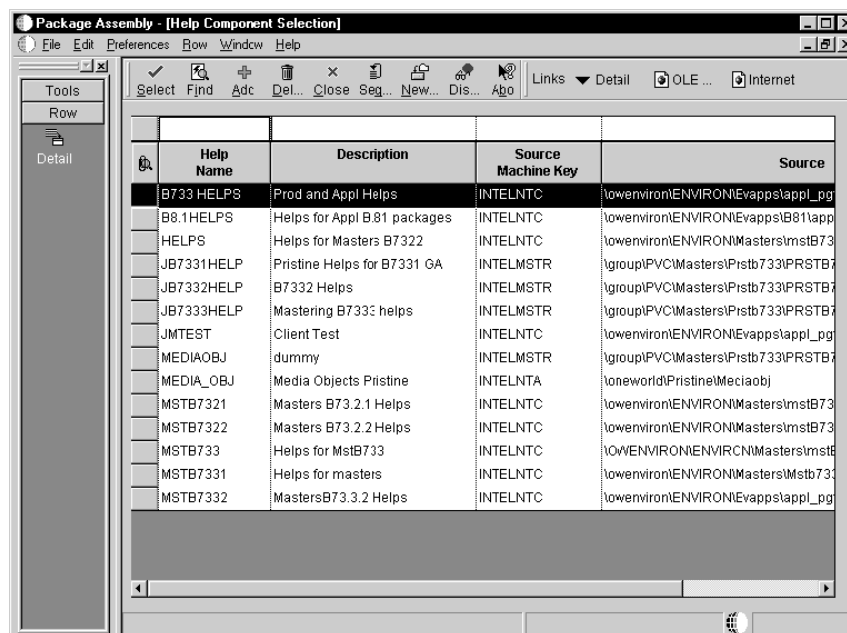
This task describes how to add features to a package. The steps are the same whether you are adding features for the first time or revising a previously assembled package.

See Also

Incorporating Features Into Packages for more information about features.

► To add features to a package

1. Access the Feature Component Selection form by doing 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 features 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 you want to include has not been defined, you can create the feature definition by clicking Add. The Feature Based Deployment Director launches. For more information about creating features, see *Defining Features*.
4. Repeat steps 2 and 3 until you have finished adding features to your package. When you are finished, click Close to return to the place from which you accessed the Feature Component Selection form.

Activating an Assembled Package

After you have assembled a package, the status remains at "Assembly." While defining the package, it is inactive until you specifically change it to "Assembly - Definition Complete," which activates the package. An assembled package cannot be built until the status has been changed to Definition Complete. Changing a package status to Complete 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 Complete, you can still 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 you want to activate. Packages that are currently In Definition are identified by an open box icon, while packages that have a Complete status have a closed box icon.
2. Choose Active/Inactive from the Row menu.

You can use this same process to change a Complete package back to In Definition.

When you are ready to define the package's build, follow the steps described in [Defining Package Builds](#).

Revising an Existing Package

Once you have assembled a package, you can easily revise any of that package's components by using the Package Component Revision form. You do not need to go through all of the Package Assembly Director's forms to revise a package.

Before You Begin

You cannot revise a package unless the package status is In Definition. If you try to revise a package whose definition is Complete, you will receive an error message. If necessary, before you choose the package to revise, choose

Active/Inactive from the Work with Packages form's Row menu to change the package status.

► **To revise an existing package**

From the Package and Deployment Tools menu (GH9083), choose Package Assembly (P9601). The Work with Packages form appears.

1. Find the package you want to revise, either by using the Find function or by expanding the Packages tree.
2. Choose the package you want to revise, and then choose Package Revisions from the Row menu.
3. On Package Component Revisions, you can choose a different parent package name or click the appropriate radio button to select a different package type (Full, Partial, or Update).
4. To change the foundation, helps, or database location, click on the appropriate icon to display the form where you can make your changes. Likewise, click the Objects icon to add objects to the package, the Features icon to add features, or the Language icon to specify a language other than English.
5. To change the description, choose Property Revisions from the Form menu.
6. When you are finished revising the package definition, click OK to return to the Work with Packages form.
7. If any build information exists for the package, OneWorld warns you that your changes will delete the existing build information. Select one of the following options:
 - Click 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 you made.
 - Click Cancel to delete your revisions and save the existing build information.

Copying a Package

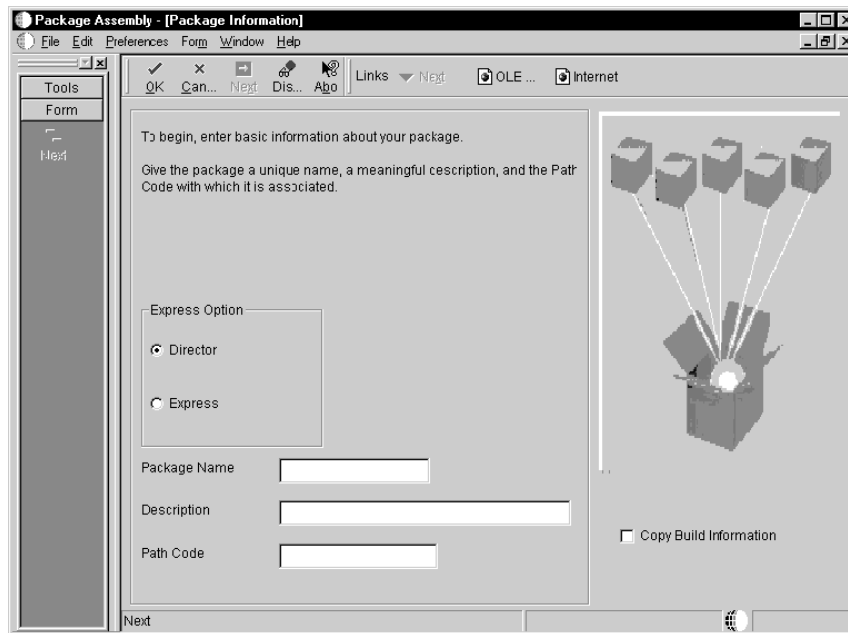
In some cases it may be easier to copy an assembled package rather than creating a new one entirely from scratch. Also, when you make a copy of a package that has been built (that is, has a Build Complete status) the original package definition

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 you want to copy, and then click Copy.



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

If the package 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 Package form, where you can choose Package Revisions from the Row menu to review and change any of the package components.

Field Descriptions

Field	Description
Copy Build Information	If you are copying a package, indicate whether you would like the package directories copied as well as the package definition. If you copy the package directories, the new package does not need to be built before it is deployed to users, because it is an exact copy of the original package.

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 all information for that package in the Package Header and Detail tables (F9603 and F9631) is deleted. In addition, if the package has been built, all specifications for that package will be deleted on all servers, as well as information in the Package Build Header and Detail tables (F96021 and F9622). Deployment records, if any, will also be deleted.

► To delete a package

From the Package and Deployment Tools menu (GH9083), choose Package Assembly (P9601). The Work with Packages form appears.

1. Find and choose the name of the package 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 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 B7333 (release name) directory structure looks similar to the following. Directories with an asterisk represent locations to which developers check in development objects.

```
PD7333 (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.

Topics

- Understanding the build process

- ❑ Understanding the Package Build Definition Director
- ❑ Defining a package build
- ❑ Revising build options for a package
- ❑ Compressing the parent package (after building the update)
- ❑ Copying a built package
- ❑ Viewing package build history and resubmitting builds

Understanding the Build Process

The following is a checklist of the processes you need to complete when you perform a build. This entire process can 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.

- ❑ Transfer objects.
Ensure that all the objects you want to include in the build have been transferred to the appropriate path code.
See [Understanding Object Movement](#).
- ❑ Ensure that the Microsoft Access database for the package has the most current replicated data.
*See [Data Replication](#) in the *System Administration Guide*.*
- ❑ Build a package.
Build a package over the path code to which objects were transferred. This process is described in this chapter.
- ❑ Perform a cross-reference build (optional).
Perform a cross-reference build to make sure 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 completed.
*See [Cross Reference Facility](#) in the *OneWorld Development Tools Guide*.*
- ❑ Deploy the software to the following:
 - Workstations and servers
 - Tiered deployment locations*See [Deployment](#).*

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 always have the option of going to the previous or next screen by clicking Previous or Next. Also, you can always cancel the build definition process by clicking Cancel.

The following summarizes each of the Package Build Definition Director forms and their function:

Welcome form	View this form for a description of the Package Build Definition Director.
Package Selection form	Use this form to choose the defined package you would like to build. The package's status must be [assembly] Definition Complete.
Package Build Location form	Use this form to indicate whether you would like 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. (Required when you build a package for a server.)
Build Specification Options form	Use this form to specify the specification tables you want to include in the package: all specification tables, 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 selected tables only 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 select options to compress directories (all or individual), data, helps, and foundation.
Individual Directory	Use this form to select the individual directories you want to compress.

Selection form**Build Features form**

Use this form to enter file set and compression information for features that were 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 a OneWorld application or other OneWorld functionality.

Package Build Revisions form

Use this form to review or change any of the options 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, a global build of business functions takes place during the package build process. When you build business functions as part of the package build, it is the same process as if you had manually executed the BusBuild program (selected Build from the Global Build menu) after the package was built.

Source and header information is gathered from the package (from the source and include directories), compiled, and put into the bin32, obj, and lib32 directories. Business functions are built in the package, not on the workstation. After business functions are built, they are compressed, provided that you turn on the Compress Package option.

OneWorld makes certain assumptions about path code, foundation, and destination for the business function build. In particular:

- When building business functions, you build from the path code you defined in the package.
- The foundation is either the same as the foundation included in the package, or, for an update package, the parent package's foundation.
- 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, you should always build business functions. Otherwise, the consolidated DLLs included in the package will not be current.

For update packages, a single build is performed for each business function defined in the package. After the individual business function is built, a global link is performed for that object and all other objects that are in the same consolidated DLL. The global link process uses the objects found in the check-in location for that package's path code.

Similarly, any business functions defined in a partial package are built individually and then linked with all objects in the check-in location. The only difference with a partial package is that all consolidated DLLs are delivered, not just the ones that may or may not be defined in the package itself.

Note:

Check-in location directories will not be updated with the build's output.

See Also

Business Functions section in the *OneWorld 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.

Topics

- Defining a Package Build
- Reviewing the Package Build Selections

Before You Begin

- ❑ Assemble your package, making sure to set the assembled package status to Definition Complete. For more information, see *Assembling Packages*.
- ❑ Verify Object Configuration Manager (OCM) mappings are correctly set for the the Package Build Report (R9621) and Server Package Build Report (R9622) that are generated as part of the package build process. For example, if you want the reports to run locally, you should make sure 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

There are two ways to access the Package Build Definition Director: from the Work With Packages form on the Package Assembly menu, or from the Work With Package Build Definition form on the Package Build menu.

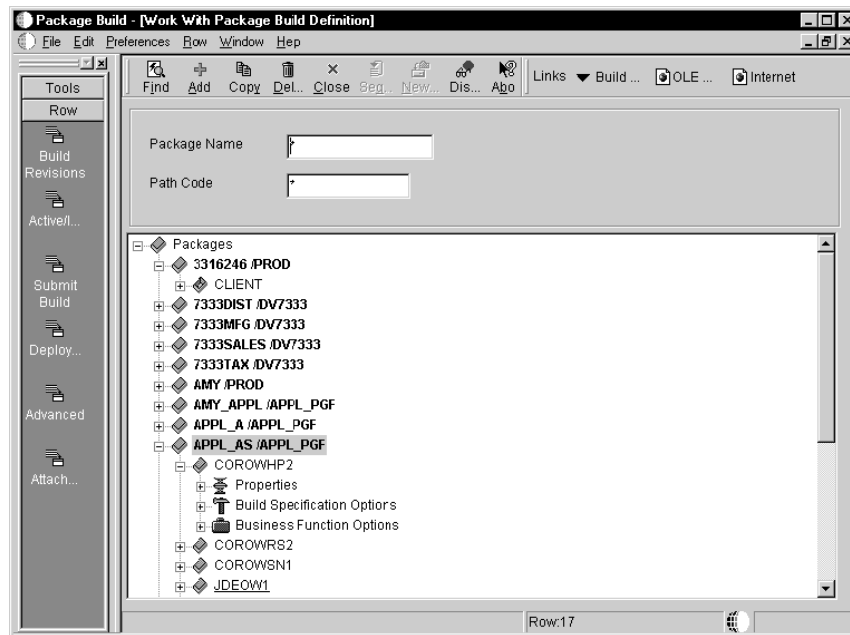
The advantage of accessing the director from the Work With Packages form is that the package name and other information is automatically entered for you. If you access the director from the Work With Package Build Definition form you will need to manually specify the name of the package you want to build.

Select one of the following options:

To access the Package Build Definition Director from the Work With Packages form, select a defined package that has a Definition Complete status, and then choose Build Director from the Row menu.

To access the director from the Package and Deployment Tools menu (GH9083), choose Package Build (P9621).

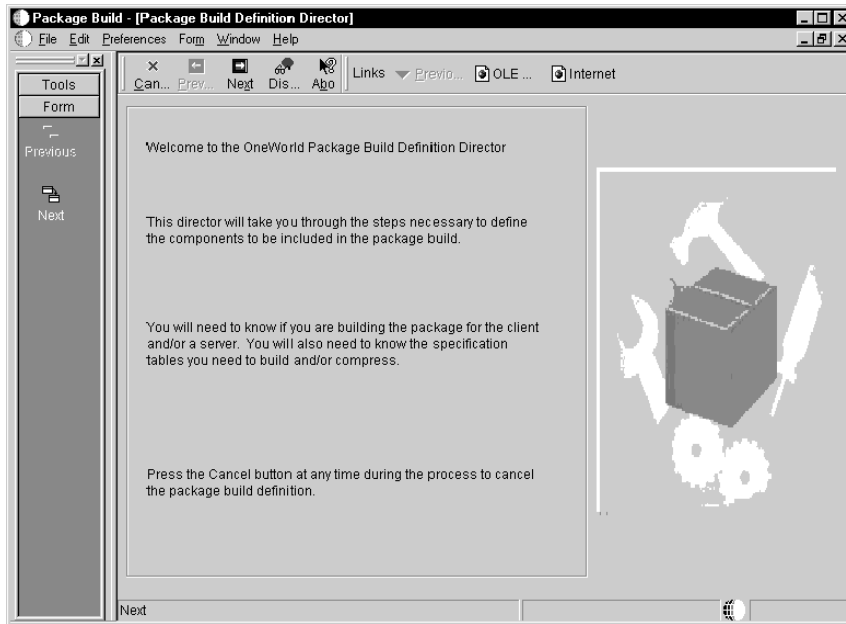
The Work With Package Build Definition form appears.



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

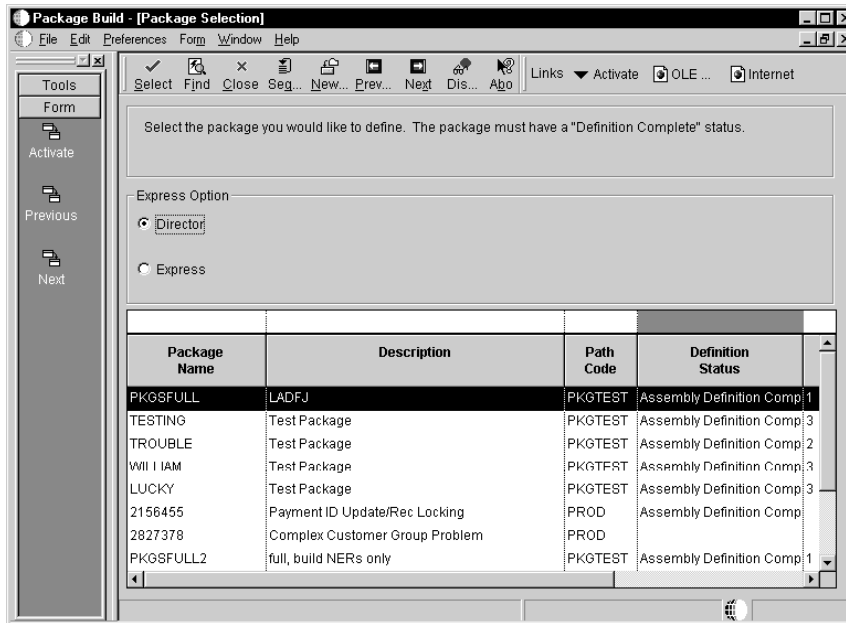
If you have any previously built packages, you can review all of the options and information you specified when you created the package, such as the package's properties, build options, business function options, and compression options.

1. Click Add to launch the Package Build Definition Director.



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

2. On Package Build Definition Director, click next.



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

If the package definition is still 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, choose one of the following options:

- Director

Choose this option if you want to customize your package build. Director allows you to navigate through the Package Build Definition screens.

- Express

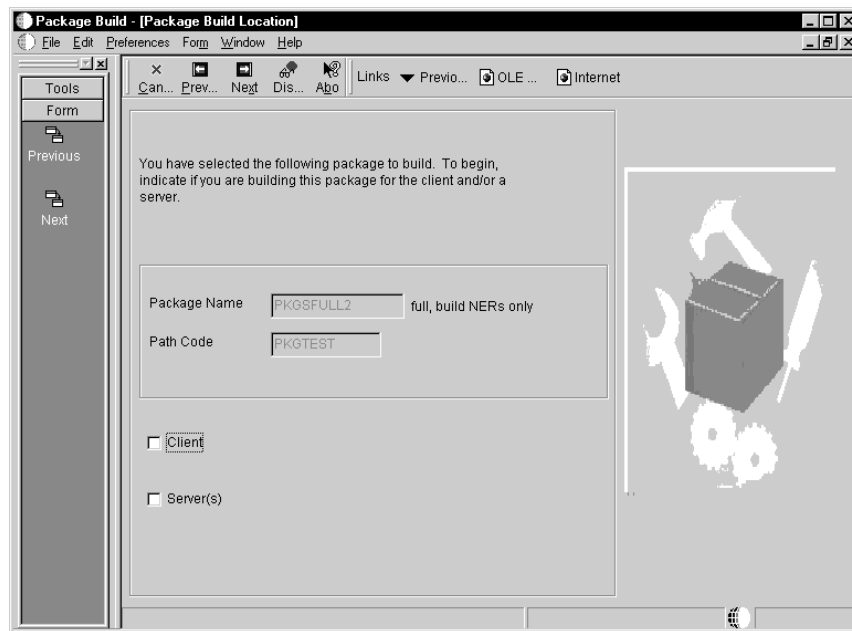
Choose this option if you want to accept the default build parameters. Express allows you accept the package build default options and skip the Package Build Definition screens.

If you Choose Express, click Next and continue with the task "To review the package build selections."

5. Click Next.

The Package Build Location form appears.

6. Specify whether you want to build a package for client workstations, one or more servers, or both clients and servers. If you are building a partial package, you will not be allowed to build the package for a server.

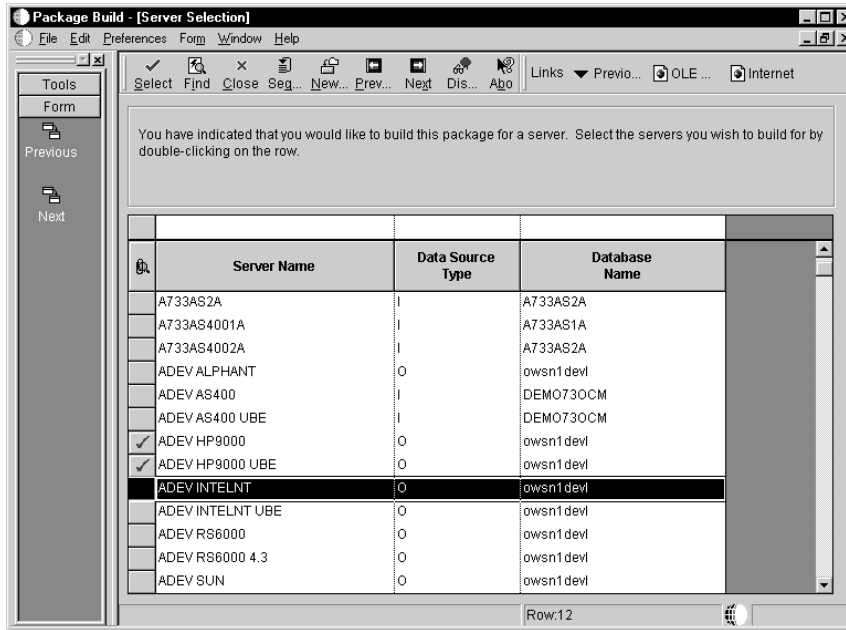


Be sure to click Server(s) if the package you are building goes to one or more servers.

If you are building a package for workstations only, skip the following step, which applies only to server package builds.

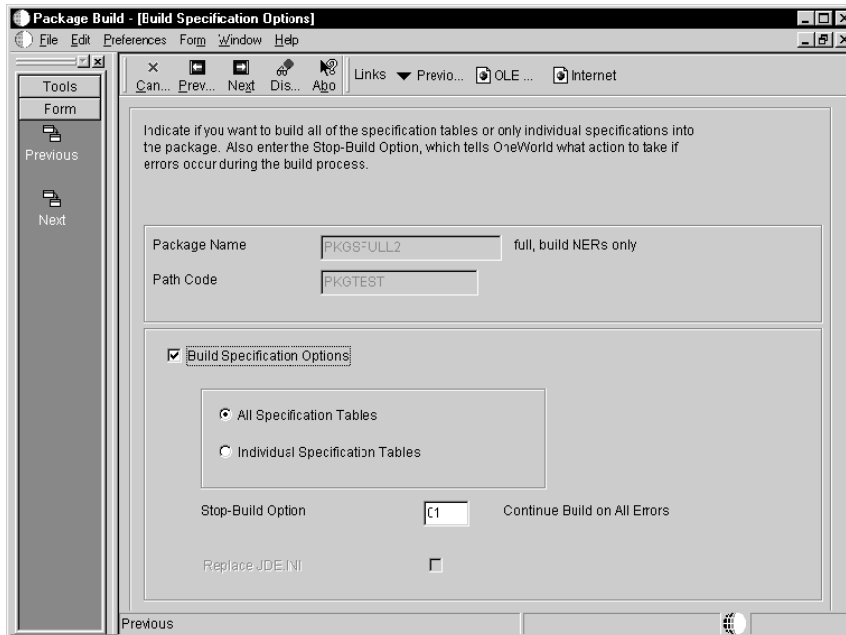
7. Click Next.

If you are building a package for a server and you clicked Server(s) 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. To choose a server, click Find, and then double-click in the column to the left of the server name. A checkmark indicates your selection. Repeat the process to choose multiple servers.

8. Click Next.



9. On Build Specification Options, complete the following fields:

- Build Specification Options
- All Specification Tables or Individual Specification Tables

If you turn on 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 (see the following page).

- Stop-Build Option

The Stop-Build option on this form allows you to indicate the point at which OneWorld should stop the build: Continue building on all errors; Stop building on specification errors; Stop building on business function errors; or Do not compress if there are errors.

- Replace JDE.INI (for update packages only)

Note:

If the package you are building goes to a server, make sure to choose Build Options and All Specification Tables.

10. Click Next.

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

Specification Table	Checked
ASVRDTL	Yes
ASVRHDR	Yes
BOBSPEC	Yes
DDCLMN	Yes
DDPKEYD	Yes
DDPKEYH	Yes
DDTABL	Yes
DSTMPL	Yes
GBRLINK	Yes
GBRSPEC	Yes
JDEBLC	Yes
POTEXT	Yes
RDASPEC	No
RDATEX	No
SMRTTMPL	No
FDASPEC	No
FDATEX	No

11. Uncheck each specification table you do not want to build. (Only the checked specification tables are built.)

12. Click Next.

If the package includes Business Functions or tables with Table Event Rules, the Business Function Options form appears. (If not, skip to instructions for the next screen.)

13. 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**
There are 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**
The Stop-Build Option field allows you to indicate the point at which OneWorld should stop building business functions: Continue building on all errors; Stop building on the first error; or Stop building on DLL failure.
- **Build BSFN Documentation**
The Build BSFN Documentation option allows you to create an online list of business function documentation. If your company does custom development you will want to be sure this option is selected to ensure that the documentation for your business

functions is generated. Business function documentation is created after you submit the build and is saved in the PrintQueue folder beneath the B7 folder that contains your other OneWorld files. To make the documentation available to other users, you should move or copy the documentation from the PrintQueue folder to a location on the server that is accessible to all users. In order to read the documentation files produced, users will need a Web browser or other application capable of displaying HTML files. For more information about building business function documentation, see the *OneWorld Development Tools Guide*.

- Clear Output Destination First

The Clear Output Destination First option allows you to clear the output directories (bin32, lib32 and obj) before building business functions so that old business functions are not included when the package is created. If you do not clear the output and a function fails to process, the global link process will use the old libraries and objects to create the consolidated DLL file. This can result in the creation of an inaccurate global package.

Note:

If the package you are building will be deployed to a server, be sure to choose Build Function Options.

14. Click Next.

If you are building an Update or Partial package that includes objects or features, the Compression Options form appears. (If not, skip to instructions for the next form.)

15. On Compress Options, complete the following options:

- Compress Options
- All Directories
- Individual Directories

When you build a server package, click on Compress Options and select All Directories. This selection will compress the directories on the enterprise server and copy them to your package on the deployment server in the following directory (where your server.ini file is located):

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

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

When you compress directories, the application objects included in the package are automatically compressed. OneWorld also creates an entry in the package INF file that indicates whether that the following items are compressed: foundation, helps, data, and application objects.

Caution

The Compress All Directories and Individual Directories options are the only options that modify the package INF. All other compression options affect only the actual compression.

- Compress Data

Compress Data compresses the Microsoft Access database associated with this package.

- Compress Helps

Compress Helps compresses the help files associated with your package.

- Compress Foundation

Compress Foundation compresses the foundation 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.

Note

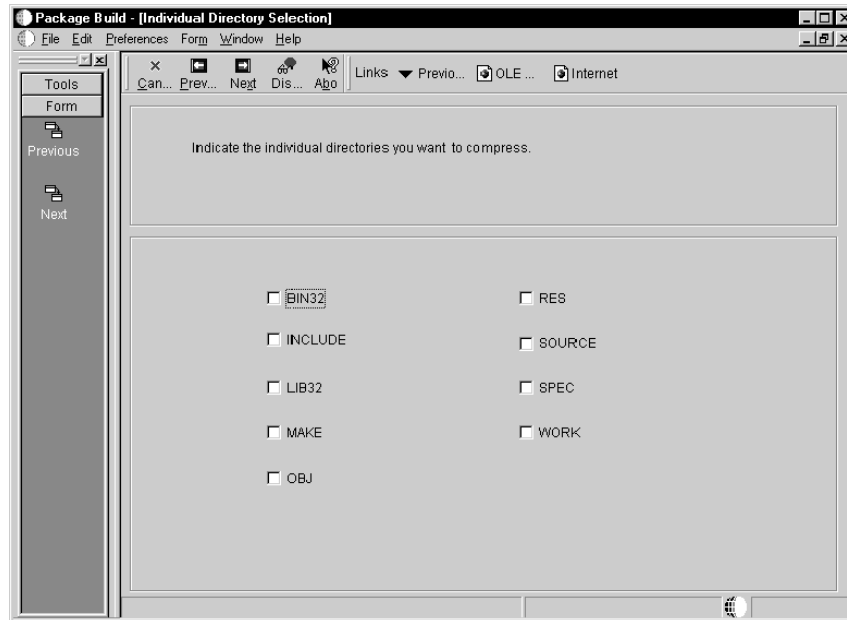
If the package you are building will be deployed to a server, there are only two situations when you would want to choose Compress Options:

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

Important

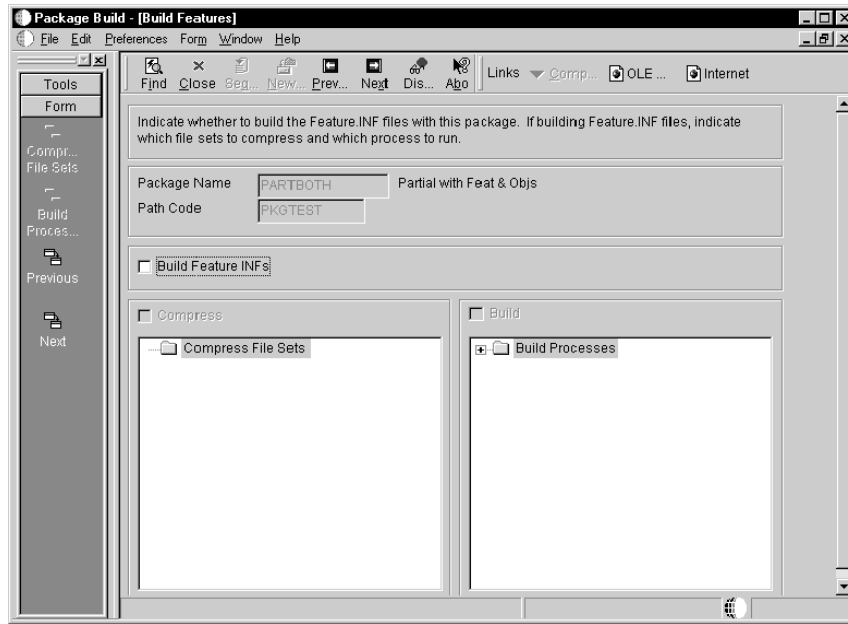
To enable creating compressed files on the server, check the `jde.ini` file on the workstation that is running the package build application. Under the BSFN BUILD section, verify that the `DoCompression` setting is 1 to enable compression. If this setting is incorrect, compressed files will not be created on the server. For more information, see *JDE.INI Settings for Server Package Builds* in the introductory section of this chapter.

16. Click Next.
17. If you elected to compress individual directories, the Individual Directory Selection form appears.



18. Click each directory you want to compress, and then click next.

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



19. On Build Features, if you want to build a feature.inf file with the package, choose Build Feature INFs.

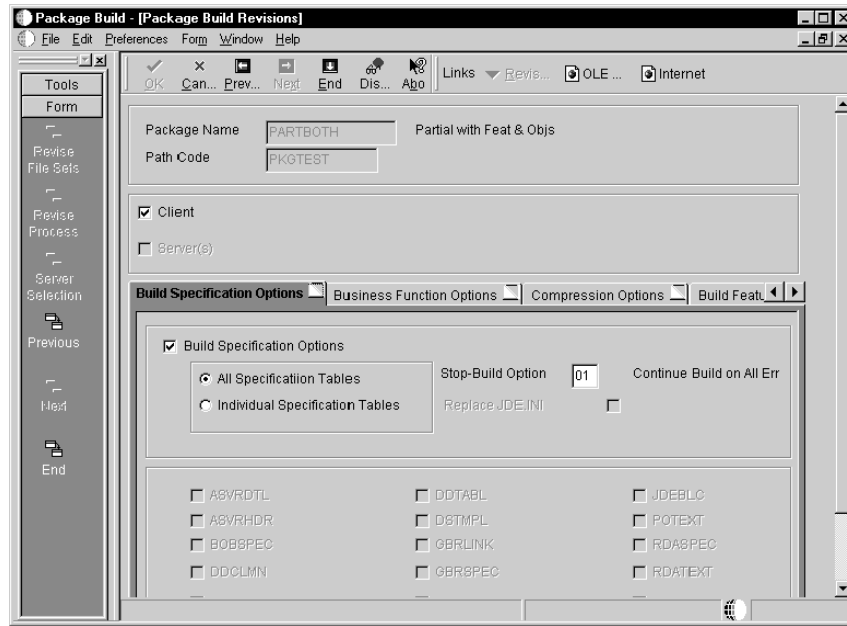
When you choose this option, the Compress and Build fields become available. For more information about the Compress and Build options, see *Configuring Features During Package Build Definition*.

20. 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 you have specified for the package. To change any of these options, click the tab for the type of option you want to change. (Only tabs for options you have selected will appear on this form.)



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

The form automatically closes and OneWorld 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 containing all applications. When the build is finished, the report will either display or print, depending on the destination you specified.

5. 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 completes successfully, you can schedule the package for deployment as described in [Deployment](#).

Field Descriptions

Field	Description
Server Name	The data source name.
Data Source Type	The type of database.
Database Name	The name assigned to the database during installation, such as HPDEVORAP or HP9000.
Build Specification Options	Select Build Options to take the package definition and copy and convert objects from the central data source to the replicated format used by workstations. Also, enter options pertaining to the build process.
All Specification Tables	Indicate whether you want to build all specification tables into the package or whether you would rather select individual tables to include in the package.
Stop-Build Option	This option enables you to specify what action OneWorld should take if an error occurs during the package building process.
Replace JDE.INI	Indicate if you want a new JDE.INI file delivered with the package. You should leave this unchecked unless your JDE.INI file has changed. For example, your JDE.INI may change when you perform upgrades or when you reconfigure in release master.
Generate NER	When resubmitting a client or server build, you can indicate whether NER should be rebuilt. If you already have a successful build of the NER, you can save substantial time by not rebuilding it.

Field	Description
Build Functions Options	
Build Mode	Indicate the compiler configuration to use for the software build.
Stop-Build Option	This option enables you to specify what action OneWorld should take if an error occurs during the business function building process.
Build BSFN Documentation	Indicate whether you want to build business function documentation during the package build process. For a full package, the documentation will be built for all business functions. For a partial or update package, documentation will be built for only those functions included in the package.
Clear Output Destination First	When building business functions, you should normally clear the output directories (bin32, lib32 and obj) so that old business functions are not included when the package is created. If you do not clear the output and a function fails to process, the global link process will use the old libraries and objects to create the consolidated DLL file. This can result in an inaccurate global package.
Compress Package	Select Compress Options to compress the applications included in the package, and to specify options for the compression process.
All Directories	Indicate whether you want to compress all directories in the package, or if you would rather select individual directories for compression.
Individual Directories	Indicate whether you want to compress all directories in the package, or if you would rather select individual directories for compression.
Compress Data	Indicate whether to compress the data in a package after the package is

Field	Description
	created.
Compress Foundation	Indicate whether to compress the foundation files in the package after the package is created.
Compress Helps	Indicate whether to compress the help files in a package after the package is created.

Processing Options for the Package Build Definition Director

The Package Build Definition Director (P9621) has a processing option that lets you specify whether to allow changes to the build definitions on individual servers.

When you enter 1 in this field, OneWorld allows you to change the build definition for individual servers. If you leave this field blank, any revisions you make to the build options and information you enter will be applied to all of the servers for which you want the package built.

For example, if you enter one, on the Work With Package Build Definition form you will be allowed to make changes to an individual server's properties, build options, business function options, and compression options. If you leave this field blank, you will not be able to make these changes to an individual server, but at the package level only. The changes will apply to all of the servers for that build.

Revising Build Options for a Package

Once you have entered a package's build options, you can easily revise any of those options by using the Build Component Revision form. You do not need to go through all of the Package Build Definition Director's forms to revise build options.

► To revise a package's build options

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

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 Packages plus sign (+) to expand the displayed tree.
2. Choose the package and click Active/Inactive from the Row menu if the package is active (as indicated by the "closed package" icon). The package must be inactive before you can make revisions.
3. Choose Build Revisions from the Row menu when the package you want to revise is selected. All of the build information for the selected package is displayed on the Package Build Revisions form.
4. Click on the appropriate tab to change the package's build options, business function options, or compression options. 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)

OneWorld 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 could copy the package definition and then modify the package record, header, or detail files as needed.

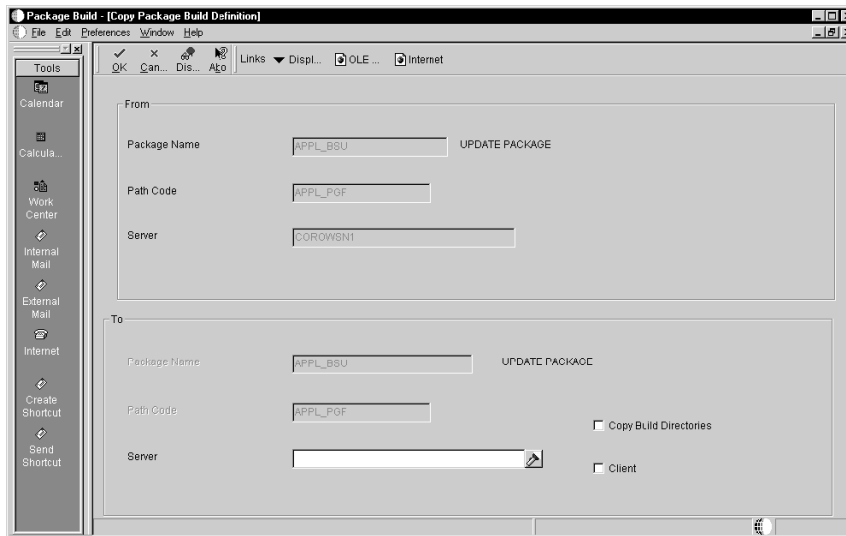
There are two ways to copy a package:

- By copying a *specific server's* build definition. You might use this method when you want to copy the same package from one server to another, or to a workstation.
- By copying the *entire package's* build definition. 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 (P9621).

1. On Work With Package Build Definition, Find the package you want to copy. Under that package, choose the server whose definition you want to copy.
2. Click Copy.



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

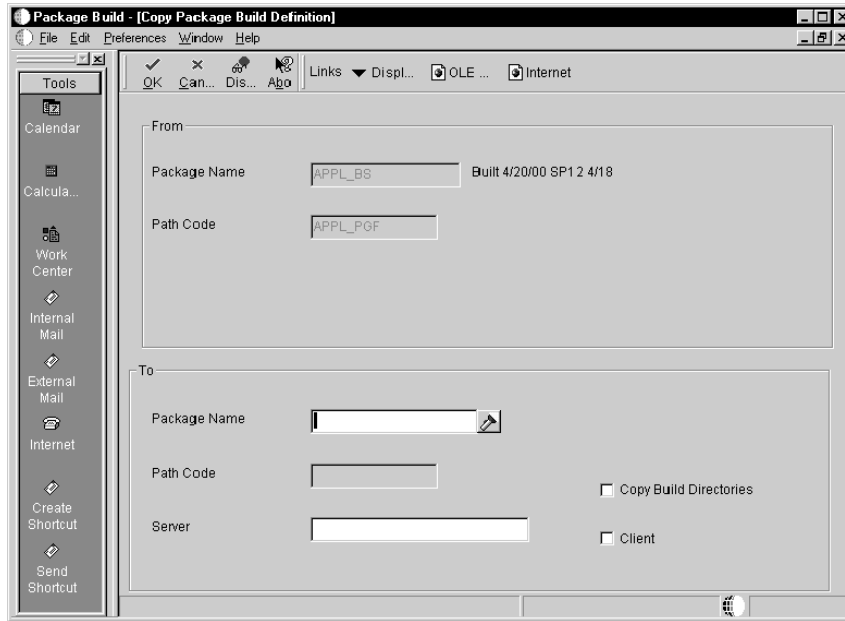
The package name, path code, and server name from which you are copying will be displayed, as well as the destination package name and path code (which are the same).

4. Complete the following optional fields:
 - Copy Build Directories.
Check this option if you want to copy the Package Build directories.
 - Client.
Check this option if you want 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 (P9621).

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



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

The package name, path code, and server name from which you are copying will be displayed.

4. Complete the following optional fields:
 - Copy Build Directories
Check this option if you want to copy the Package Build directories.
 - Client
Check this option if you want 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 by using the Package Build History (P9622) application. This application also enables you to view logs 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 specs and objects and build the package again.

See Also

[*Viewing Package Build History and Logs*](#) for information about 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 OneWorld application or other OneWorld functionality. Like OneWorld 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 you might want to include when you build a package:

- **ActiveX controls.** The OneWorld Application Design Aid tool supports including ActiveX controls in OneWorld application. If ActiveX controls are delivered with OneWorld, you will need a way to copy these controls to the workstation.
- **Open Data Access (ODA) data sources.** Open Data Access requires that additional ODBC data sources be created on any workstation or server that utilizes ODA.
- **Sales Force Automation databases.** The Sales Force Automation feature requires that a different Microsoft Access database be installed on the workstation so that it can be disconnected from the network during offline operation. A registry setting must also be written to indicate that the machine is used offline.
- **BMC Patrol, GenCorba, GenCom, and other third-party interfaces or products.** Each of these requires additional components on the workstation and server in order to function. As OneWorld functionality expands to support additional third-party products and interfaces, these products will each have their own set of supporting files.

Prior to OneWorld Xe, custom programming was required to accommodate adding feature components to the workstation and server. With OneWorld Xe, 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 deploy it using the Package Deployment Director or multitier deployment.

Because feature components are not OneWorld 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.

Topics

- ❑ Feature build and deployment process overview
- ❑ Understanding the Feature Based Deployment Director
- ❑ Creating a feature
- ❑ Defining the feature components
- ❑ Reviewing feature components
- ❑ Revising features
- ❑ Copying features
- ❑ Adding features during package assembly
- ❑ Configuring features during package build definition
- ❑ Installing packages containing features
- ❑ Understanding feature entries in the package.INF file

Feature Build and Deployment Process Overview

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

1. 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 will specify the feature name and type, enter a brief description, and specify installation parameters.

The Feature Based Deployment Director's other forms 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 *Defining Features*.

2. 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 indicate the features you want to include.

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

3. 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 files sets that will be compressed within the package, and to indicate 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*.

4. 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 for scheduling packages that do not include features.

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

5. 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 director's forms 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 always have the option of going 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 summarizes each of the Feature Based Deployment Director's forms and their function.

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. You will 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 will also need to select the 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 pick the file sets 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 if the feature you are defining requires an addition, modification, or deletion of an entry in the Windows registry. Registry information you enter in this form will be delivered in the package

containing the feature.

The Shortcut Definition Form

Use this form if you want to create shortcuts on the Windows desktop as part of the installation process. When you enter shortcut information, after the feature is installed a shortcut will be created automatically on the desktop.

The Additional Package Build Processes Form

Use this form to specify whether you want to execute a batch application or executable program either before or after the package containing the feature is installed.

The Additional Install Processes Form

Use this form if you need to install a third-party product as part of the feature installation.

The Initialization File (INI) Definition Form

Use this form if you need to make changes to an initialization file such as the JDE.ini file as part of the installation. For example, OneWorld developers often manually turn off the Debug log, but if a package is delivered only to developers, you could automatically make this change.

The ODBC Data Source Definition Form

Use this form if the feature requires that additional ODBC data sources be added when the package containing the feature is installed.

The Features Summary Form

Use this form to review the information you entered for the feature on the previous form. Key information for the feature is displayed in a tree structure on the left side of the form. Radio buttons on the right side of the form enable you to access any of the previous forms and, if necessary, make changes.

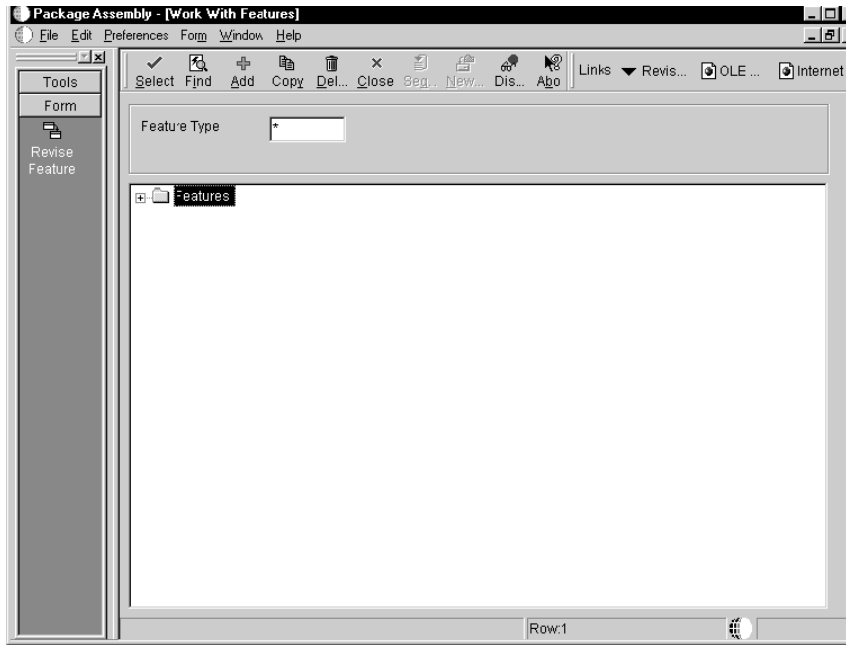
Creating a Feature

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

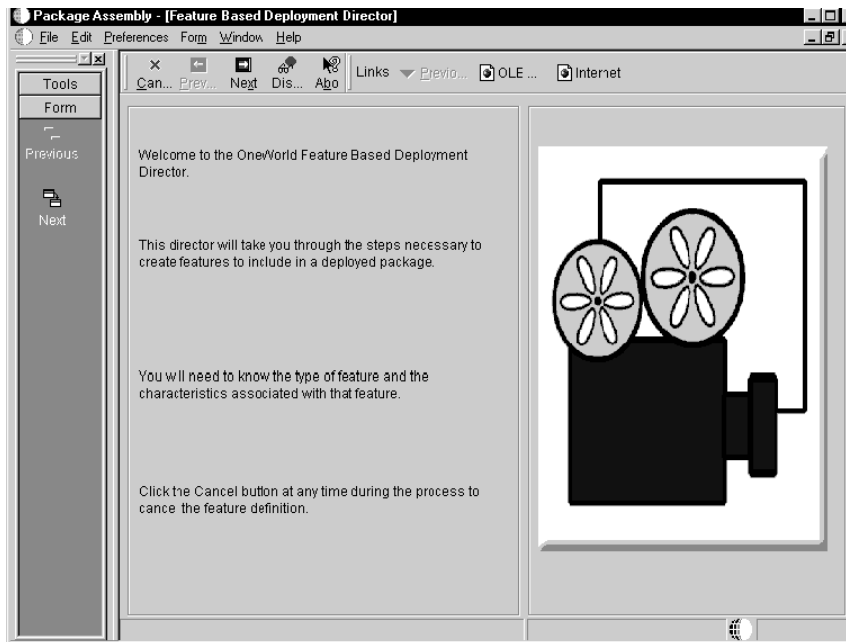
► **To create a feature**

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

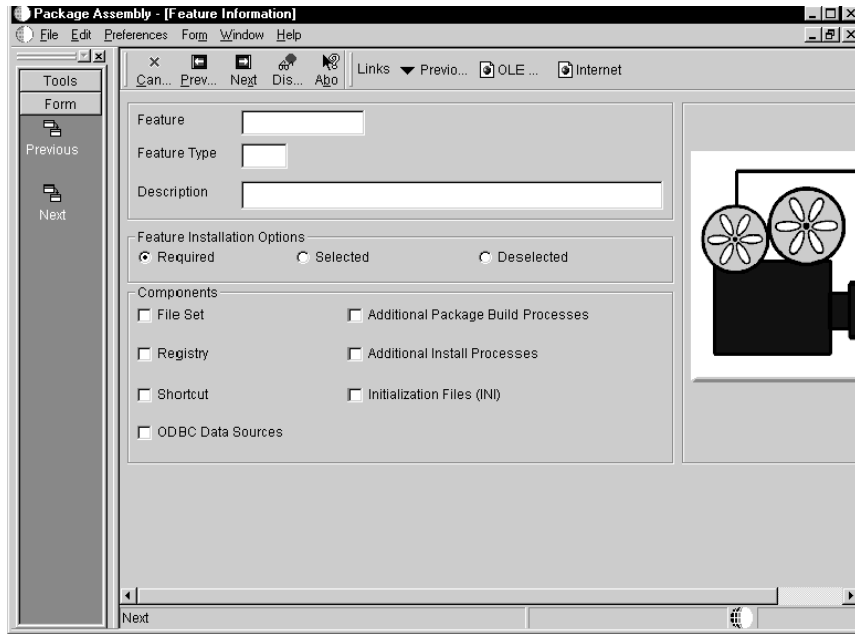
1. On Work with Packages, choose Features from the Form menu.



2. On Work With Features, click Add.



3. On Feature Based Deployment welcome, click Next.



4. On the Feature Information form, complete the following fields:

- Feature
- Feature Type
- Description
- Required
- Selected
- Deselected

Select as many of the following options and feature components as you like. The forms that subsequently display depend on the feature components you select.

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

5. Click Next.

Defining the Feature Components

The forms that appear next depend on the type of components you chose to include in your feature. Complete the relevant tasks below to define the components you are adding.

- Define a file set component
- Define a registry component
- Define a shortcut component
- Define additional package build processes
- Define additional install processes
- Define an initialization file component
- Define an ODBC data source component
- Reviewing feature components

Define a file set component

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

► To define a file set component

The screenshot shows the 'Package Assembly - [File Set Definition]' dialog box. The window title is 'Package Assembly - [File Set Definition]'. The menu bar includes 'File', 'Edit', 'Preferences', 'Form', 'Window', and 'Help'. The toolbar contains icons for 'Find', 'Del...', 'Close', 'Seg...', 'New...', 'Prev...', 'Next', 'Sav...', 'Dis...', and 'Abo'. On the left side, there is a 'Tools' pane with buttons for 'Form', 'Previous', 'Next', and 'SaveN...'. The main area of the dialog is divided into several sections. The top section has 'Feature' (text box with 'ALLCMP'), 'Feature Type' (text box with '1'), and 'Platform' (text box with '80'). Below this is a section for 'File Set' with a text box, 'Description' (text box), 'Source Path' (text box), 'Compress' (checkbox), and 'Target Path' (text box). On the right side, there is a pane with a tree view showing 'ALLCMP'.

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 identifying the source location of a file set.
- Compress
Option to compress the file.
- Target Path
A path identifying the target location of a file set.

The source path tells OneWorld where to find the file set to be copied into the package, and the target path indicates the location where 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.

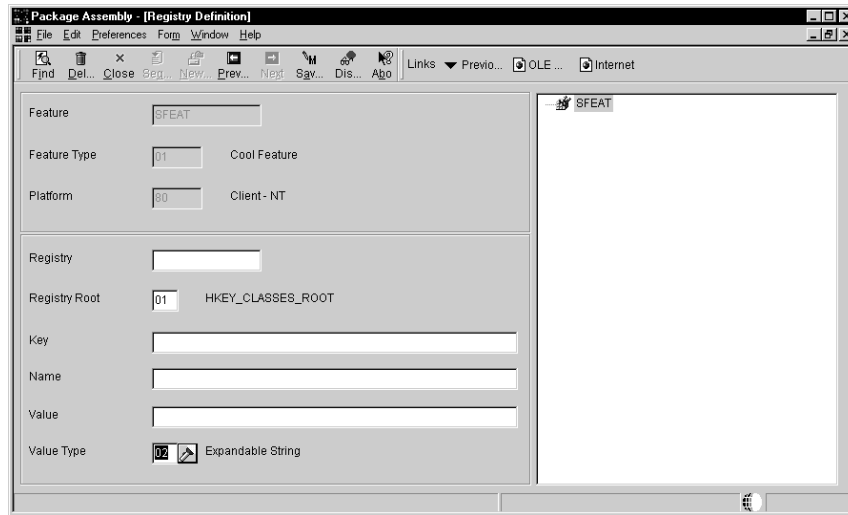
You can use this same form to modify or delete any previously defined file sets. Existing file sets are displayed in the tree structure on the right side of the form. To modify a file set, select the file set on the tree structure and modify any of the fields for the file set. To delete a file set, select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

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

Define 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 that the value will be stored as.

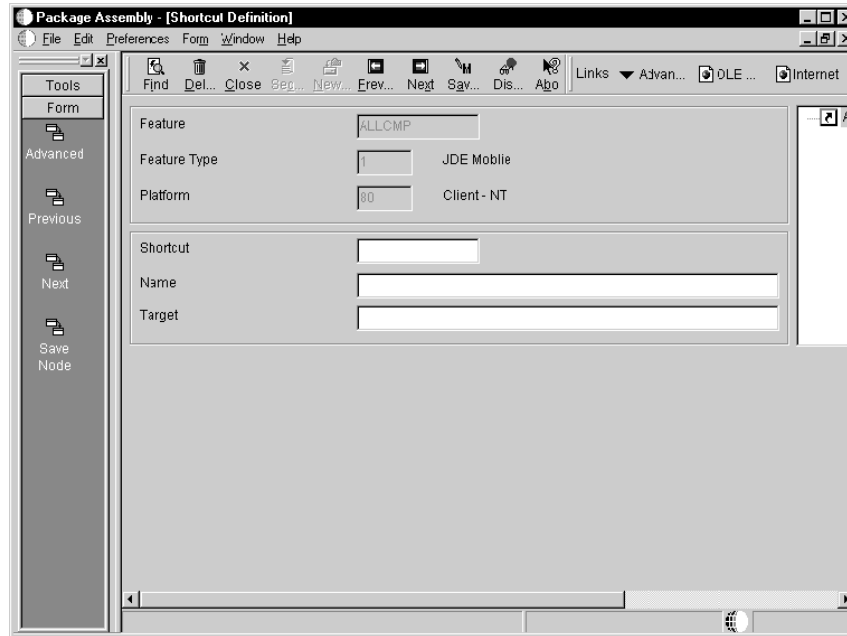
You can use this same [Registry] form to modify or delete any previously defined registry definitions. Existing registry definitions are displayed in the tree structure on the right side of the form. To modify a registry definition, select the item on the tree structure and modify any of the fields for the registry definition. To delete a registry definition, select the item and click Delete.

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

Define 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:

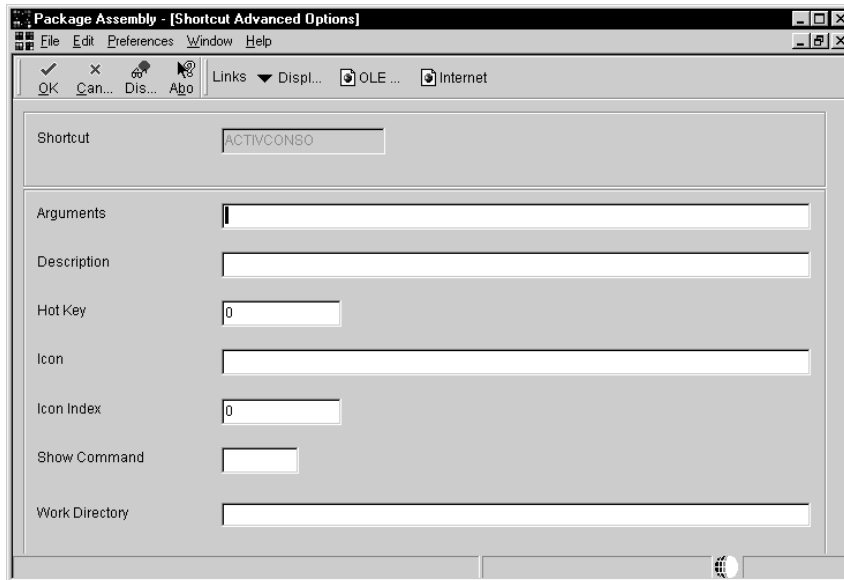
- **Shortcut**
A name that identifies a unique shortcut to a user's computer.
- **Name**
The name of the shortcut.
- **Target**
The path and file name of a target file.

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

You can use this same [shortcut] form to modify or delete any previously defined shortcut definitions. Existing definitions are displayed 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, select Advanced from the Form menu.



3. On Shortcut Advanced Options you can enter the following:

- Arguments
Parameters that are entered at the command line for the shortcut.
- Description
The description of the shortcut.
- Hot Key
A key sequence that will automatically launch the shortcut when pressed.
- 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
Specifies the window size after the shortcut is launched, such as window 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 click the Save Node from the Form menu.
6. When you have added all your shortcuts, click Next.

Define 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 identifying the order that the process will be run relative to the other processes running during the package build.
 - Synchronous Execution

A flag that indicates whether the package build job waits for the process to complete before continuing.

- Batch Application or Executable

Select whether the process is an application or an executable

If you selected Batch Application, complete the following fields:

- UBE Name

Name of the OneWorld UBE

- UBE Version

Version number of the UBE

- Machine Name

Name of the machine on which the UBE will run

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

- Executable Name

The name of the executable program that is launched to install the third-party software.

- Target Path

The path and file name of a target file.

- Parameters

The executable parameters passed to the setup program that will be launched to install third party.

You can use this form to modify or delete any previously defined processes. Existing processes are displayed 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 execute the process either before or after the feature is built.)
- 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 execute.
- Click Next.

Define additional install processes

If you selected 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

The screenshot shows a software dialog box titled "Package Assembly - [Additional Install Processes]". It has a menu bar with "File", "Edit", "Preferences", "Form", "Window", and "Help". Below the menu is a toolbar with icons for "Find", "Close", "New...", "Prev...", "Next", "Save", "Save...", "Del...", "Del...", "Dis...", and "Ab". The main area is divided into several sections:

- Feature:** Text box containing "ALLCMP".
- Feature Type:** Text box containing "1".
- Platform:** Text box containing "B0".
- Third Party:** Empty text box.
- Description:** Empty text box.
- Sequence:** Text box containing "1".
- Synchronous Execution:** A checkbox that is currently unchecked.
- Executable Name:** Empty text box.
- Target Path:** Empty text box.
- Parameters:** Empty text box.

 On the right side, there is a preview window showing a small icon and the text "Execute ALLCMP Before Inst". On the left side, there is a vertical toolbar with buttons for "Form", "Previous", "Next", "Save", "Save Ho...", "Delete", and "Delete Ho...".

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 that is to be run.
- **Sequence**
A number identifying the order that this process will run relative to the other additional install processes.
- **Synchronous Execution**
A flag that indicates whether the package install waits for the process to complete before continuing.
- **Executable Name**
The name of the program that launches the third-party software.
- **Target Path**
The path to the executable file (do not include the name of the file).

- Parameters

The executable parameters passed to the third-party program.

2. When you are finished adding third-party product information, choose Save from the Form menu.
3. To delete a third-party product definition, select the item on the tree display then choose Delete from the Form menu.
4. Click Next.

Define 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
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 identifying the action associated with the key in the initialization file.

You can use form to modify or delete any previously defined initialization file definitions. Existing definitions are displayed in the tree structure on the right side of the form.

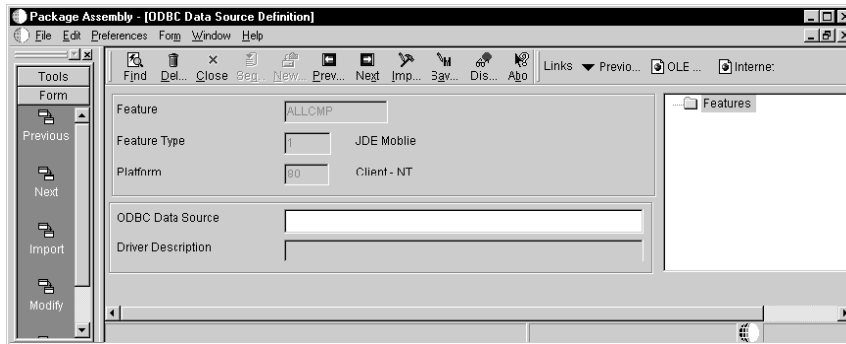
- To modify an initialization file definition, select the item on the tree structure and modify any of the fields for the definition
 - To delete an initialization file definition, select the item and click Delete.
2. When you are finished adding initialization information, click the Save Node from the Form menu.
 3. Click Next.

Define 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.

Select one of the following tasks to 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
Name of the ODBC Data Source

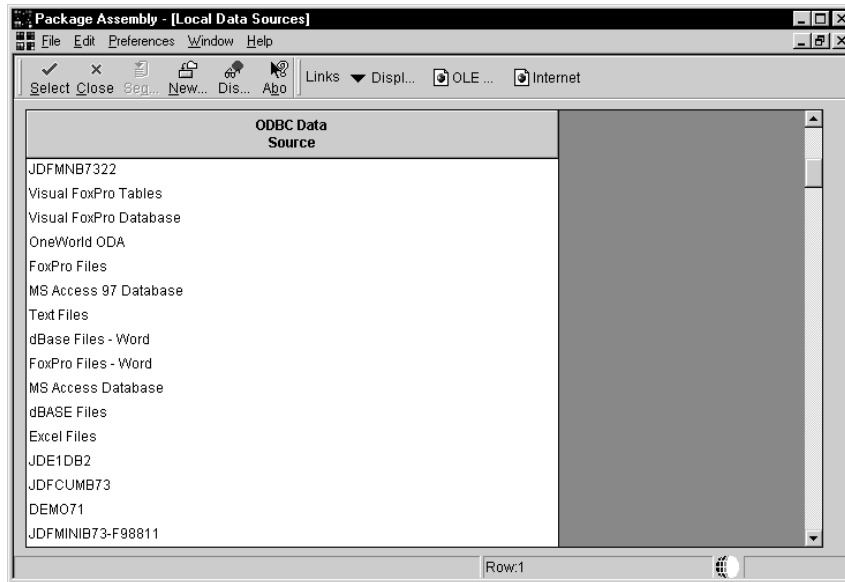
2. Click on Save Node from the Form menu.

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

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

► **To import existing ODBC data sources**

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



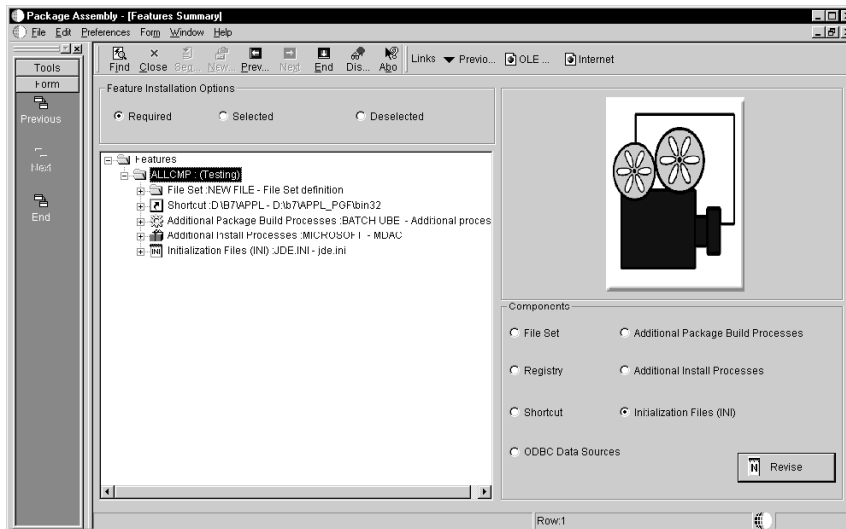
1. On Local Data Sources, choose one or several data sources (pressing the Control or Shift key) and click the Select button to add the data sources to the Feature. The program returns to the ODBC Data Source Definition form.
2. When you are finished adding data source information, click the Save Node from the Form menu.
3. Click Next.

To modify existing data sources, enter the data source name and then choosing Modify from the Form menu. The ODBC Data Source Revisions form appears. Use this form to make changes to the data source. 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 you entered on any of the Feature Based Deployments forms. For more information about revising feature components, see *Revising Features*.

► **To review the feature components**



1. On the Features Summary form, choose a component in the right-hand 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.

Revising Features

The process for revising previously defined features is very similar to adding a feature. Modification is done on the same forms you used when you first defined the feature.

► **To revise 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 you want to modify, and then choose **Revise Feature** from the **Form** menu.
3. To modify a feature component, on **Feature Summary**, choose one of the components listed on the right side of the form, and then click **Revise**. The selected form appears, and you can modify existing information or enter new information for the component.
4. Click the **Save** node from the **Form** menu to save the changes.
5. When you have finished modifying information, click **Close** to return to the **Feature Summary** form.
6. Repeat steps 4 and 5 to modify any other feature components.
7. When you are finished making feature modifications, click **Close** to return to the **Work With Features** form.

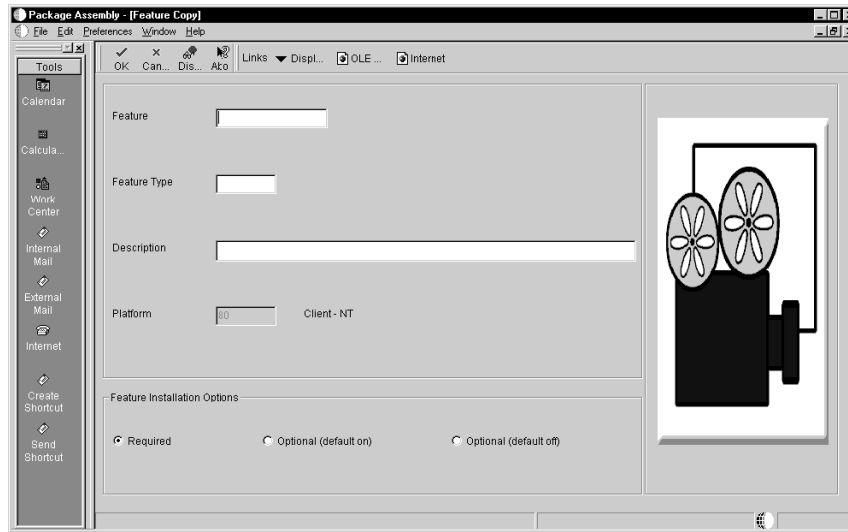
Copying Features

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 whose definition you want to copy, and then click **Copy**.



3. On Feature Copy, complete the following fields:
 - Feature
 - Feature Type
 - Description
 - Required, Optional (default on), or Optional (default off)
4. Click OK to return to the Work With Features form.
5. To revise the new feature definition, select the feature and choose Revise Feature from the Form menu.

See Also

Revising Features for more information about revising a feature.

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. Perform one of the tasks below:

- Adding features to a new package during package assembly
- Adding features to an existing assembled package

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*.

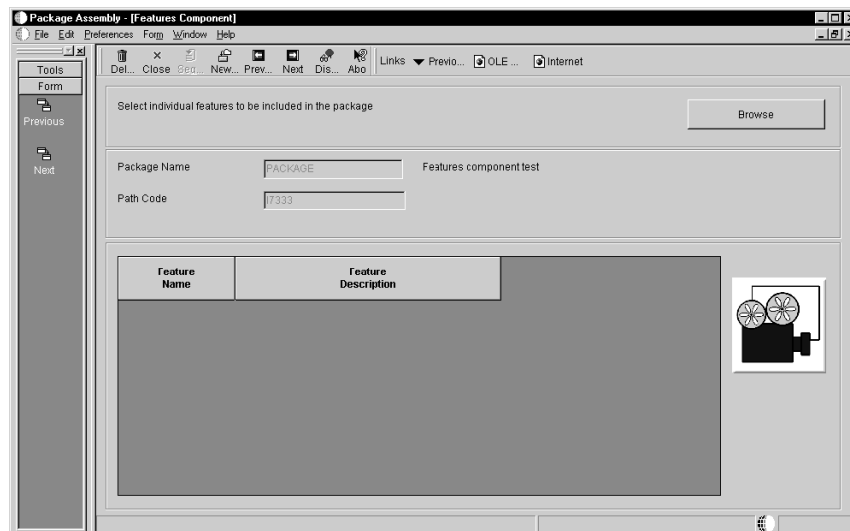
Note:

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

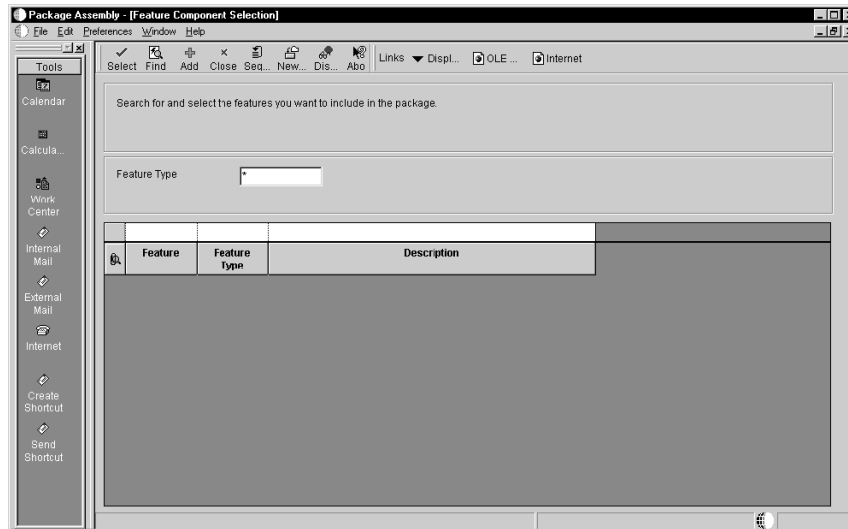
► **To add existing features to a new package during package assembly**

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 Package Assembly Director's forms and follow the steps in "Assembling a Package" until the Feature Component form is displayed.



3. To add a feature, click Browse.

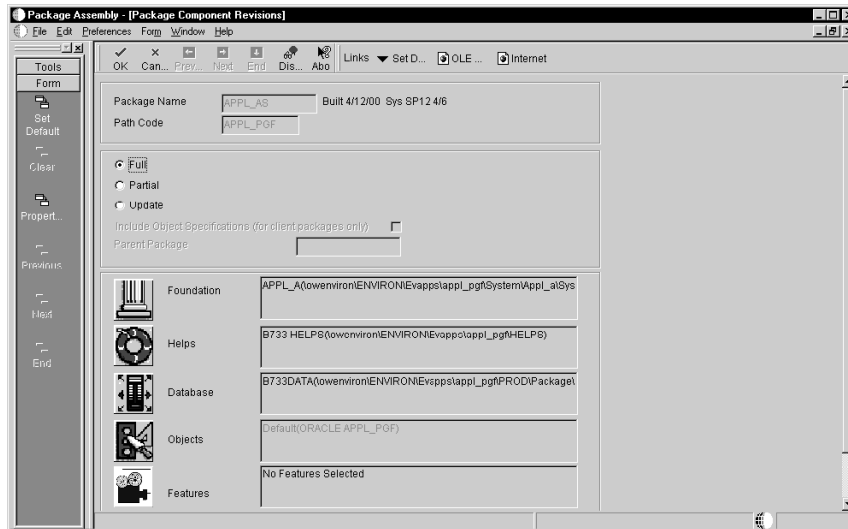


4. On Feature Component Selection, click Find to display the list of available features.
5. Use one of the following methods to select one or more features to include in your package:
 - Choose a feature and click the Select button. (Use the Control or Shift key to select multiple features)
 - Double click on each feature.
6. When you are finished adding features, click Close to return to the Features Component form. The selected features will appear.
7. 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). The Work with Packages form appears.

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



2. Click the Features button.
3. To delete a feature that was previously included in the package, on Features Component, select the feature and then click Delete. Continue with step 6 below.
4. To add a feature, click Browse. Continue with the sub steps below.
 - A. On Feature Component Selection, click Find to display defined features.
 - B. Use one of the following methods to select one or more features you want to add:
 - Choose a feature and click the Select button. (Use the Control or Shift key to select multiple features)
 - Double click on each feature.

A check mark appears in the left column of each feature you selected.
 - C. When you are finished adding features, click Close to return to the Features Component form.
5. Click Close to return to the Package Component Revisions form.
6. Click OK to return to the Work With Packages form.

Configuring Features During Package Build Definition

The Package Build Definition Director includes the Build Features form, which enables you to specify whether Feature INF files will be built 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*.

Topics

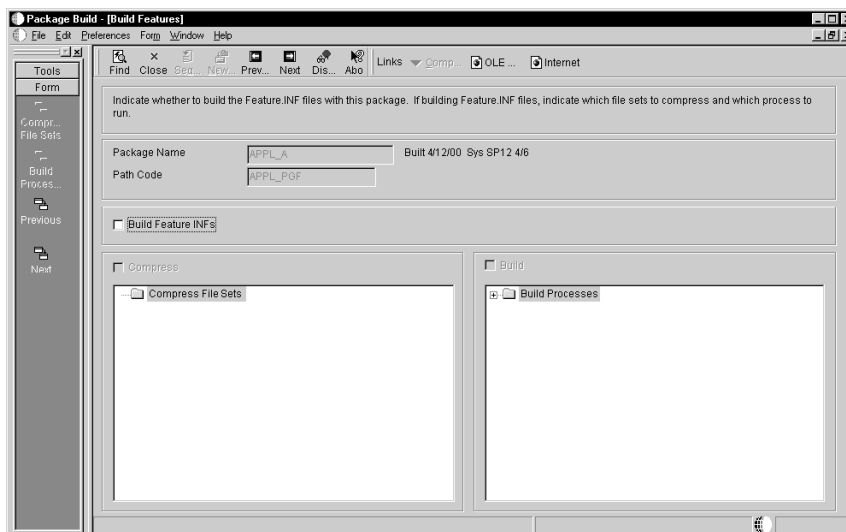
- Configuring features during package build definition
- Configuring features for an existing package build definition

The following tasks describe only the processes for entering feature information for a new or existing package build definition. For more detailed information about creating package build definitions, see *Defining Package Builds*.

► **To configure features for a new package build definition**

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

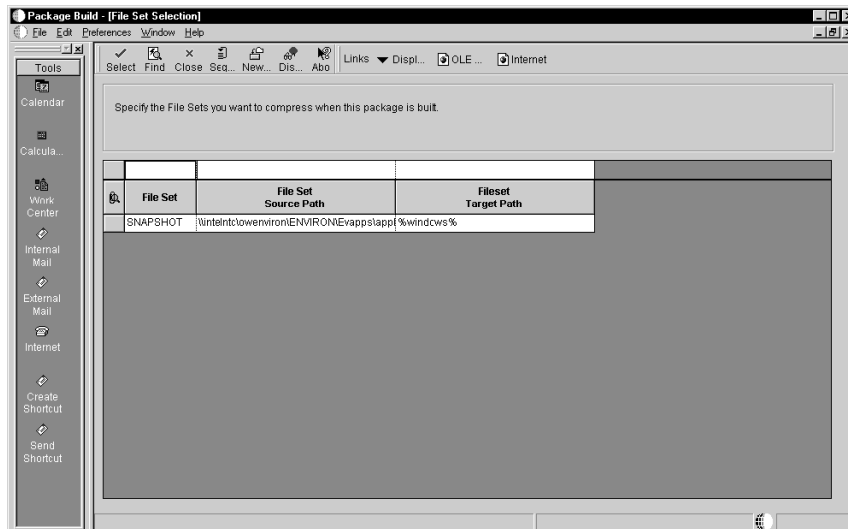
1. On Work with Package Build Definition, click Add to launch the Package Build Definition Director.
2. Click Next and complete the tasks in "Defining Package Builds" until you come to the Build Features form.



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:
 - Compress file sets
 - Build OneWorld processes

► **To compress file sets**

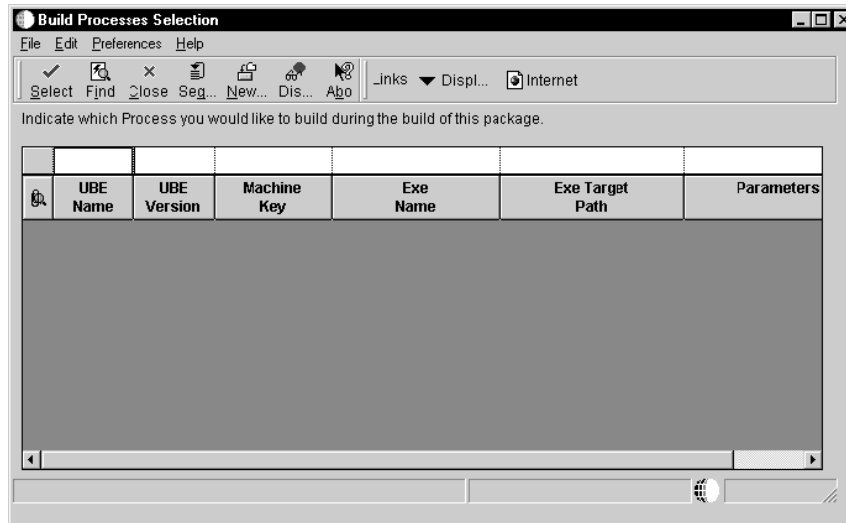
1. To compress file sets choose Compress, and then click Compress File Sets from the Form menu.



2. On File Set Selection, select each feature you want to include by double-clicking in the column next to the File Set column. (You can also select by choosing a file set and clicking Select.) When you are finished selecting file sets, click Close.
3. Continue with Build OneWorld processes, or click Next and complete the remaining forms to finish defining the package build.

► **To build OneWorld processes**

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

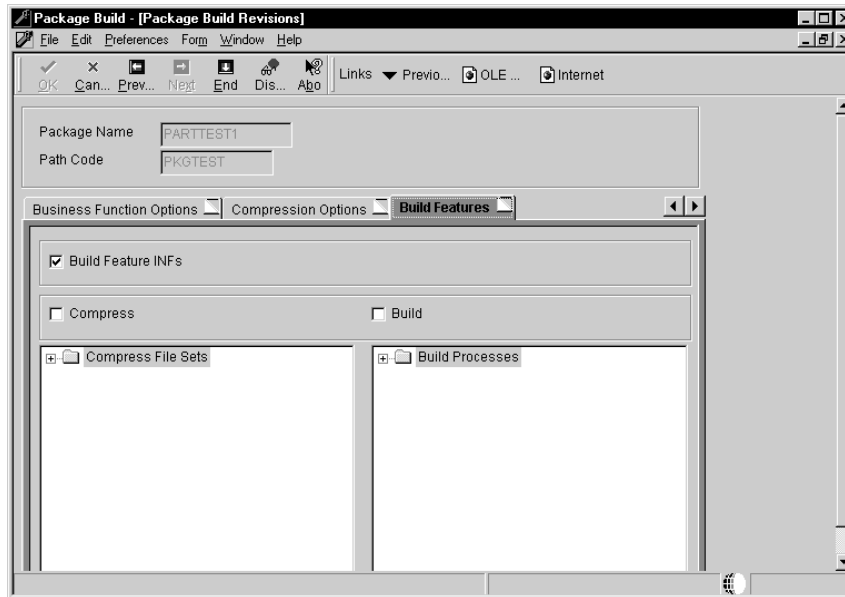


2. On Build Processes Selection, select each process you want to build by double-clicking in the column next to the UBE Name column. (You can also select by highlighting a process and clicking Select.)
3. When you are finished selecting processes to build, click Close.
4. From the Form menu, click on Build Processes and manually select each process to run during package build. If this is not done, nothing will run even though the process is 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 containing features.
2. Choose Build Revisions from the Row menu.
3. On Package Build Revisions, click the Build Features tab.



4. Modify or add to the existing build feature settings displayed:
 - Build Feature INFs
 - Compress
 - Build
5. If you select Compress, click Revise File Sets from the Form menu to modify file sets. The File Set Selection form appears. When you are finished modifying file sets, click Close to return to the Package Build Revisions form.
6. If you select Build, click Revise Processes to modify processes. The Build Processes Selection form appears. When you are finished modifying processes, click Close to return to the Package Build Revisions form.
7. If you select Build, from the Form menu, click on Build Processes and manually select each process to run during package build. If this is not done, nothing will run even though the process is included in the feature.
8. Click OK to complete the package build definition.

Installing Packages Containing Features

Packages containing features are installed on workstations and servers the same way as any other package: through the Workstation Installation and Deployment Server Installation applications.

When you launch either of these installations, by choosing the Custom option you can then select the features you want to install.

See Also

OneWorld Installation Guide for more information about the Workstation Installation and Deployment Server Installation applications.

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 is created 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*
- *Understanding Feature INF Files*

Viewing Package Build History and Logs

The Package Build History application allows you to view information pertaining to the build process, including the options and objects you specified when you created the build definition. This application 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 Form menu's View Logs option allows you to view four logs containing additional information about the build process. These logs are useful in the event that the build does not complete successfully and you need to know the errors that occurred during the build.

If a build does not complete successfully, you can use the Resubmit Build option to resume the build from the point where 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 complete, you may 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 where it failed, resetting the status enables you to start the build process from the beginning.

Topics

- Viewing the package build history
- Viewing logs
- Resubmitting a package build

- ❑ Changing the build status

Viewing the Package Build History

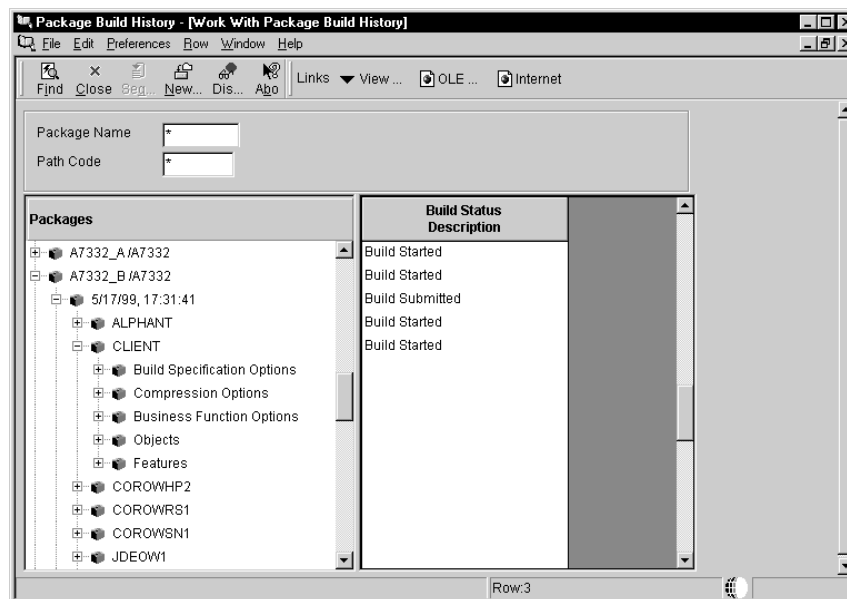
OneWorld maintains a history of the package build in the Build Detail History table (F96225). This table contains details about any package components that failed to build successfully.

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, OneWorld checks the F96225 table and rebuilds only the business functions or other package components that have a status of "Not Built" or "Error." OneWorld 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, directories are automatically compressed after building successfully.

► 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 machines. 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 the ones you specified when you created the package's build definition. For example, if you elected 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:

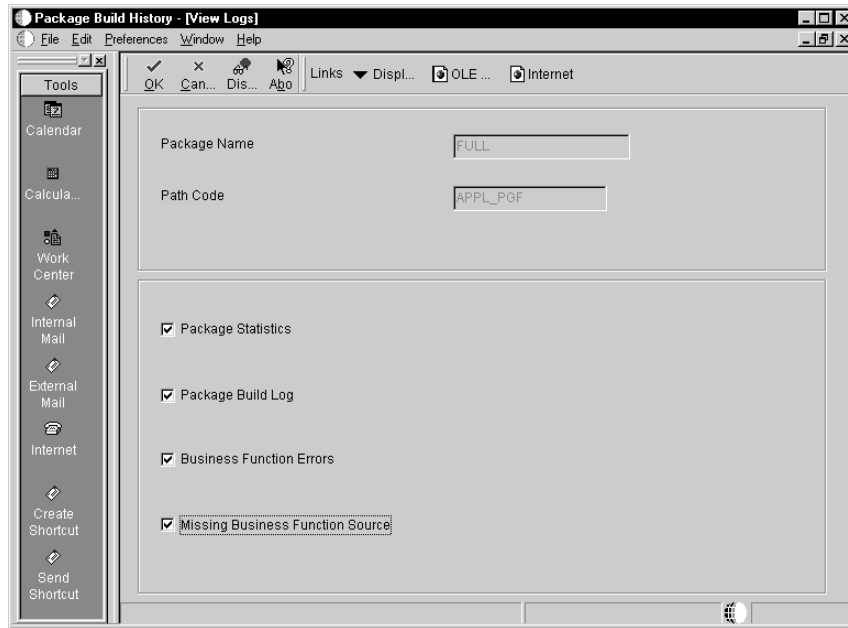
- Package Statistics Log
- Package Build Log
- Business Function Errors Log
- Missing Business Function Source Errors Log

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, click any of the following log options and click OK:

- Package Statistics
- Package Build Log
- Business Function Errors
- Missing Business Function Source

Each log you selected displays 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.

Field Descriptions

Field	Description
Package Statistics	This option allows you to view count and size statistics for the package directories that were built.
Package Build Log	This option allows you to view errors that may have occurred during a package build. These errors could have occurred while building the specification files or the objects for the package.

Field	Description
Business Function Errors	This option allows you to view the results of the business function build for this package. Both errors and warnings display in this report. A summary appears at the end of the report that indicates how many errors and warnings occurred for each dll. Use this information to determine if a rebuild is necessary.
Missing Business Function Source	This option lists all source members that were not available when the business function was created. The program attempted to find these members because each had a record in the Object Librarian table (F9860). However, a matching source could not be found in the source directory. To resolve these errors, either delete the Object Librarian record or provide a source member.

Where to Find the Error Logs

You can view error logs without using the Package Build History application by locating the desired log in the correct directory. Error logs are stored on the deployment server in directories beneath 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 package's work directory.

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

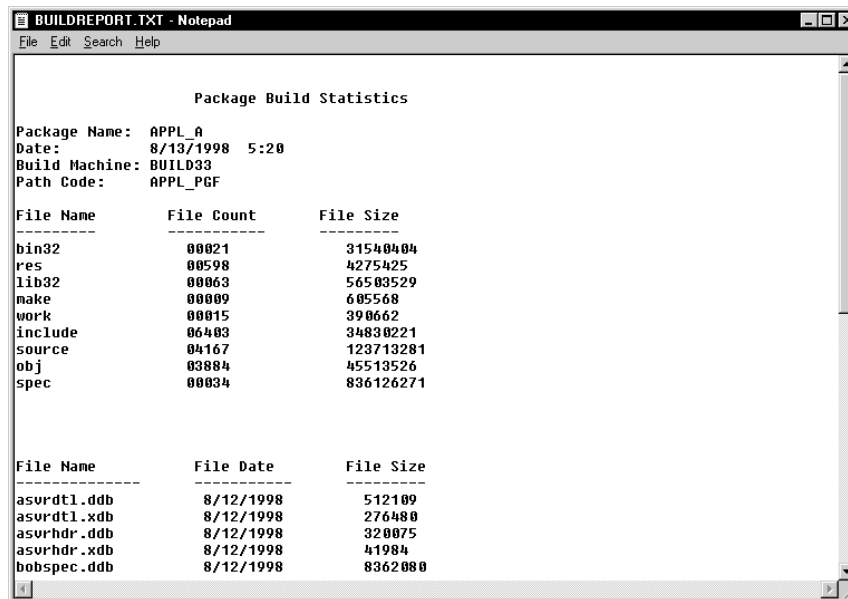
In the following examples, PD7333FA is used as the package name. To determine your actual directory, substitute your package name for "PD7333FA."

- Package Statistics: \PD7333FA\work\buildreport.log
- Package Build log: \PD7333FA\builderror.log
- Business Function Errors log: \PD7333FA\work\buildlog.txt
- Missing Business Function Source log: \PD7333FA\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 you can use to check 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. The log can be used to verify that the package was built successfully.

The Package Statistics log looks similar to this:



```

Package Build Statistics

Package Name:  APPL_0
Date:          8/13/1998  5:20
Build Machine: BUILD33
Path Code:    APPL_PGF

File Name      File Count      File Size
-----
bin32          00021           31540404
res            00598           4275425
lib32          00063           56503529
make           00009           605568
work           00015           390662
include        06403           34830221
source         04167           123713281
obj            03884           45513526
spec           00034           836126271

File Name      File Date       File Size
-----
asvrdtl.ddb    8/12/1998      512109
asvrdtl.xdb    8/12/1998      276480
asvrhdr.ddb    8/12/1998      320075
asvrhdr.xdb    8/12/1998      41984
bobspec.ddb    8/12/1998      8362080

```

The Package Build Log

The Package Build log can be found in the package name directory. This log lists the steps executed 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 this:

Full Package

```

BuildError.log - Notepad
File Edit Search Help
4/26/00 17:36:20 0
4/26/00 17:36:20 0
4/26/00 17:36:20 0
4/26/00 17:36:20 0 Package Build Log
4/26/00 17:36:20 0
4/26/00 17:36:20 0 Package Name: PRDD C
4/26/00 17:36:20 0 Date: 4/26/2000 17:36
4/26/00 17:36:20 0 Build Machine: BUILD36
4/26/00 17:36:20 0 Path Code: PRDD
4/26/00 17:36:20 0 RecordUsed: 4/26/2000 173619
4/26/00 17:36:20 0
4/26/00 17:36:20 0 Build Directories for Package Complete.
4/26/00 17:36:20 0 Build Package Definition List.
4/26/00 17:36:20 0 Create Package .inf file.
4/26/00 17:36:21 0 Copy bin32 and res directory for Package
4/26/00 17:38:21 0 Copy lib32,source,include,obj,make and work for Package.
4/26/00 18:12:18 0 Call BUSBUILD to build the business functions.
4/26/00 18:12:18 0 Spec file ASURDTL begun.
4/26/00 18:12:18 1 RDB record count in ASURDTL : 5335
4/26/00 18:12:45 0 Number of TAM records fetched: 5335.
4/26/00 18:12:45 0 Number of TAMadds Attempted: 5335.
4/26/00 18:12:45 0 Number of TAMadds succeeded: 5335 failed: 0.
4/26/00 18:12:45 0 Number of records TAM says it has: 5335.
4/26/00 18:12:45 0 Creating Indexes for table ASURDTL.
4/26/00 18:12:47 0 Finished creating Indexes for table ASURDTL.
4/26/00 18:12:47 0 Spec file ASURDTL finished.
4/26/00 18:12:47 0 ..... Other spec files .....
.
4/27/00 02:58:57 0
4/27/00 02:59:00 0 Spec file SMRTTHPL begun.
4/27/00 02:59:01 1 RDB record count in SMRTTHPL : 151
4/27/00 02:59:07 0 Number of TAM records fetched: 151.

```

Partial Package

```

BuildError.LOG - Notepad
File Edit Search Help
5/11/00 07:56:58 0
5/11/00 07:56:58 0
5/11/00 07:56:58 0
5/11/00 07:56:58 0 Package Build Log
5/11/00 07:56:58 0
5/11/00 07:56:58 0 Package Name: PRODPART A
5/11/00 07:56:58 0 Date: 5/11/2000 7:56
5/11/00 07:56:58 0 Build Machine: BUILD36
5/11/00 07:56:58 0 Path Code: PRDD
5/11/00 07:56:58 0 RecordUsed: 5/11/2000 75657
5/11/00 07:56:58 0
5/11/00 07:56:58 0 Build Directories for Package Complete.
5/11/00 07:56:58 0 Build Package Definition List.
5/11/00 07:56:58 0 Create Package .inf file.
5/11/00 07:56:59 0 Get TAM Table Names.
5/11/00 07:56:59 0 Create Empty TAM specs.
5/11/00 07:57:01 0 Copy bin32 and res directory for Package
5/11/00 07:59:53 0 Copy lib32, make and work for Package.
5/11/00 08:02:06 0 Spec file DDCLMN begun.
5/11/00 08:02:07 1 RDB record count in DDCLMN : 69837
5/11/00 08:03:04 0 Number of TAM records fetched: 69837.
5/11/00 08:03:04 0 Number of TAMadds Attempted: 69837.
5/11/00 08:03:04 0 Number of TAMadds succeeded: 69837 failed: 0.
5/11/00 08:03:04 0 Number of records TAM says it has: 69837.
5/11/00 08:03:04 0 Creating Indexes for table DDCLMN.
5/11/00 08:04:17 0 Finished creating Indexes for table DDCLMN.
5/11/00 08:04:17 0 Spec file DDCLMN finished.
5/11/00 08:04:17 0 .....Other spec files.....
5/11/00 08:11:34 0 Spec file SMRTTHPL begun.
5/11/00 08:11:34 1 RDB record count in SMRTTHPL : 151
5/11/00 08:11:37 0 Number of TAM records fetched: 151.
5/11/00 08:11:37 0 Number of TAMadds Attempted: 151.

```

Update Package

```

BuildError.LOG - Notepad
File Edit Search Help
3/29/00 14:10:18 0
3/29/00 14:10:18 0
3/29/00 14:10:18 0
3/29/00 14:10:18 0
3/29/00 14:10:18 0 Package Name: TTUPDATE
3/29/00 14:10:18 0 Date: 3/29/2000 14:10
3/29/00 14:10:18 0 Build Machine: BUILD36
3/29/00 14:10:18 0 Path Code: PROD
3/29/00 14:10:18 0 RecordUsed: 3/29/2000 141017
3/29/00 14:10:18 0
3/29/00 14:10:18 0 Build Directories for Package Complete.
3/29/00 14:10:18 0 Build Package Definition List.
3/29/00 14:10:18 0 Create Package .inf file.
3/29/00 14:10:19 0 Get TAM Table Names.
3/29/00 14:10:19 0 Create Empty TAM specs for Update.
3/29/00 14:10:19 0 Build 'object' types from definition.
3/29/00 14:10:26 0 Final size of spec files: count 34 size 179662.
3/29/00 14:10:26 0 Check for compression Records.
3/29/00 14:10:26 0 Successful Build of Package 'TTUPDATE', 3/29/2000 14:10.

The last 3 sections of the inf file

This gets added to the workstations program files. This gets added to the Start Menu
[START]
ProgramGroupName=OneWorld
Item1=SYSTEM\BIN32\activconsole.exe,JDEdwards OneWorld ActivEra Solution Explorer

This is the icons that get install to the workstation

```

Error and warning messages are preceded with the word "ERROR" and "WARNING" respectively. All other messages are informational messages that 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 executed before the failure.

For example, following are some steps from the log and some troubleshooting suggestions to follow if the build fails during or after the step:

- Build Directories for Package Complete

A failure at this point means that the directories in the package could not be built. If this occurs, make sure the server did not go down. Also, make sure you have authority to create directories in the package directory.
- Build Package Definition List

This step builds a list of all items defined in the package. If the build fails at this step, make sure the object is defined in the F9631 table.
- Create Empty TAM specs

This step creates the local workstation specifications (TAM tables) in the package directory under the specs directory. If you get a failure here, make sure the server didn't go down. Also, make sure you have permission to create directories in 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 you receive a failure at this point, make sure that each of these directories exist in the check-in location and under 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 you receive a failure at this point, make sure that each of these directories exist in the check-in location and under the package directory.

- Spec file DDCLMN begun
- Spec file DDCLMN finished

These two message go together. If you receive the "begun" message but not the "finished" message, this means the spec file was not built. After the package is finished you need to build just the one TAM spec file that failed. There will probably be an entry in the log indicating that there was an error 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 built with errors. Business functions in this report could be business functions that are still in development and have not yet been checked in. If they have never been checked in, they will not have source and therefore will be listed in the Missing Business Function Source Errors log.

The Business Functions Errors log looks similar to this:

```

buildlog.txt - Notepad
File Edit Search Help

BUILD LOG
Package: JB7332FUL
Date: Mon Sep 20 20:28:28 1999
Build Machine: SAMIAH
Path Code: PRSTB7332
Source Path: \\INTELMSTR\GROUP\PUC\Masters\Prstb733\PRSTB7332\package\JB7332FUL
Foundation Path: \\INTELMSTR\GROUP\PUC\Masters\Prstb733\System\JB7332_A\System
Destination Path: \\INTELMSTR\GROUP\PUC\Masters\Prstb733\PRSTB7332\package\JB7332FUL

Generating source for NER and TER...

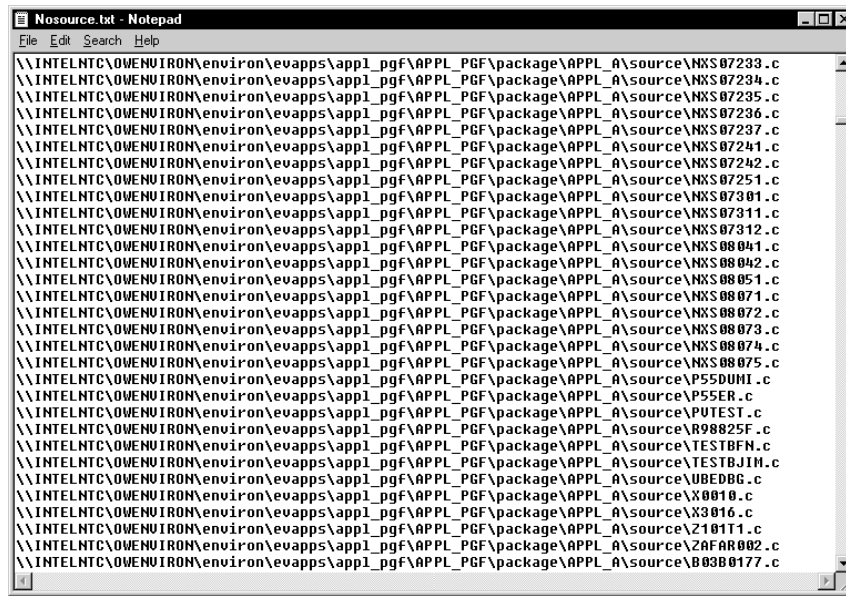
CWARE      : BSFN: B4600350...
CWRKFLOW   : BSFN: B988830...
JDBTRG1    : TBLE: F00092...
JDBTRG1    : TBLE: F0018...
JDBTRG1    : TBLE: F0018T...
JDBTRG1    : TBLE: F0022...
JDBTRG1    : TBLE: F0028...
JDBTRG1    : TBLE: F0029...
JDBTRG1    : TBLE: F0032...
JDBTRG1    : TBLE: F004201...
JDBTRG1    : TBLE: F00900...
JDBTRG1    : TBLE: F0101...
JDBTRG1    : TBLE: F01131...
JDBTRG1    : TBLE: F01133...
JDBTRG1    : TBLE: F0301...
JDBTRG1    : TBLE: F0312...
JDBTRG1    : TBLE: F03B11...
JDBTRG1    : TBLE: F03B112...
JDBTRG1    : TBLE: F03B1121...

```

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 there was no source.

The Missing Business Function Source Errors log looks similar to this:



```

\\INTELNTC\OEWENIROM\envirom\evapps\app1_pgF\APPL_PGF\package\APPL_A\source\NXS07233.c
\\INTELNTC\OEWENIROM\envirom\evapps\app1_pgF\APPL_PGF\package\APPL_A\source\NXS07234.c
\\INTELNTC\OEWENIROM\envirom\evapps\app1_pgF\APPL_PGF\package\APPL_A\source\NXS07235.c
\\INTELNTC\OEWENIROM\envirom\evapps\app1_pgF\APPL_PGF\package\APPL_A\source\NXS07236.c
\\INTELNTC\OEWENIROM\envirom\evapps\app1_pgF\APPL_PGF\package\APPL_A\source\NXS07237.c
\\INTELNTC\OEWENIROM\envirom\evapps\app1_pgF\APPL_PGF\package\APPL_A\source\NXS07241.c
\\INTELNTC\OEWENIROM\envirom\evapps\app1_pgF\APPL_PGF\package\APPL_A\source\NXS07242.c
\\INTELNTC\OEWENIROM\envirom\evapps\app1_pgF\APPL_PGF\package\APPL_A\source\NXS07251.c
\\INTELNTC\OEWENIROM\envirom\evapps\app1_pgF\APPL_PGF\package\APPL_A\source\NXS07301.c
\\INTELNTC\OEWENIROM\envirom\evapps\app1_pgF\APPL_PGF\package\APPL_A\source\NXS07311.c
\\INTELNTC\OEWENIROM\envirom\evapps\app1_pgF\APPL_PGF\package\APPL_A\source\NXS07312.c
\\INTELNTC\OEWENIROM\envirom\evapps\app1_pgF\APPL_PGF\package\APPL_A\source\NXS08041.c
\\INTELNTC\OEWENIROM\envirom\evapps\app1_pgF\APPL_PGF\package\APPL_A\source\NXS08042.c
\\INTELNTC\OEWENIROM\envirom\evapps\app1_pgF\APPL_PGF\package\APPL_A\source\NXS08051.c
\\INTELNTC\OEWENIROM\envirom\evapps\app1_pgF\APPL_PGF\package\APPL_A\source\NXS08071.c
\\INTELNTC\OEWENIROM\envirom\evapps\app1_pgF\APPL_PGF\package\APPL_A\source\NXS08072.c
\\INTELNTC\OEWENIROM\envirom\evapps\app1_pgF\APPL_PGF\package\APPL_A\source\NXS08073.c
\\INTELNTC\OEWENIROM\envirom\evapps\app1_pgF\APPL_PGF\package\APPL_A\source\NXS08074.c
\\INTELNTC\OEWENIROM\envirom\evapps\app1_pgF\APPL_PGF\package\APPL_A\source\NXS08075.c
\\INTELNTC\OEWENIROM\envirom\evapps\app1_pgF\APPL_PGF\package\APPL_A\source\P55DUMI.c
\\INTELNTC\OEWENIROM\envirom\evapps\app1_pgF\APPL_PGF\package\APPL_A\source\P55SER.c
\\INTELNTC\OEWENIROM\envirom\evapps\app1_pgF\APPL_PGF\package\APPL_A\source\PUTEST.c
\\INTELNTC\OEWENIROM\envirom\evapps\app1_pgF\APPL_PGF\package\APPL_A\source\R98825F.c
\\INTELNTC\OEWENIROM\envirom\evapps\app1_pgF\APPL_PGF\package\APPL_A\source\TESTBFH.c
\\INTELNTC\OEWENIROM\envirom\evapps\app1_pgF\APPL_PGF\package\APPL_A\source\TESTBJH.c
\\INTELNTC\OEWENIROM\envirom\evapps\app1_pgF\APPL_PGF\package\APPL_A\source\UBEDBG.c
\\INTELNTC\OEWENIROM\envirom\evapps\app1_pgF\APPL_PGF\package\APPL_A\source\X0010.c
\\INTELNTC\OEWENIROM\envirom\evapps\app1_pgF\APPL_PGF\package\APPL_A\source\X3010.c
\\INTELNTC\OEWENIROM\envirom\evapps\app1_pgF\APPL_PGF\package\APPL_A\source\Z10111.c
\\INTELNTC\OEWENIROM\envirom\evapps\app1_pgF\APPL_PGF\package\APPL_A\source\ZAFAR002.c
\\INTELNTC\OEWENIROM\envirom\evapps\app1_pgF\APPL_PGF\package\APPL_A\source\B03B0177.c

```

Server Logs

All compile logs for the enterprise server are located on the server itself in the source directory of the DLL that the object belongs to. For example, suppose you want to see the log for the Sales Order Entry master business function (B4200310) in the package PACKAGE1 on an HP 9000 whose BuildArea is /u02/jdedwardsoneworld/packages. There would be a file called /u02/jdedwardsoneworld/packages/PACKAGE1/source/CDIST/b4200310.log, since B4200310 is in the CDIST.DLL.

If there were a problem with the linking of the CDIST.DLL (shared library) on the HP 9000, there would be a file called /u02/jdedwardsoneworld/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 look at member B4200310 of the FAILED file in library PACKAGE1.

Resubmitting a Package Build

The following task describes how to 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, find and choose the package you want to resubmit. When doing this you can have two options:
 - Select a specific server to resubmitting only that server's builds
 - Select the CLIENT heading to resubmit only the workstation builds
2. Choose Resubmit Build from the Row menu.
3. If you generated NERs when you initially submitted the build, OneWorld displays a window asking if you want to re-generate the NERs. Click OK to re-generate NERs, or click Cancel to skip this process.

Note

If you do not want to regenerate NERs, you can change a processing option to prevent this window from displaying. To disable this window, right click on Package Build History, and choose Prompt For Values. Type '2' in the Generate NER field and click OK.

4. Choose a destination for the build report, and click OK:
 - On Screen
 - To Printer

The form automatically closes and OneWorld begins building the package. Build time varies, depending on the number and size of the items in your package. When the build is finished, the report will either display or print, depending on the destination you specified.

5. Review the report to make sure that all components in your package were generated successfully. If the report indicates any errors, review the error logs for more detail.

If the package build completes successfully, you can schedule the package for deployment as described in [Deployment](#).

Changing the Build Status

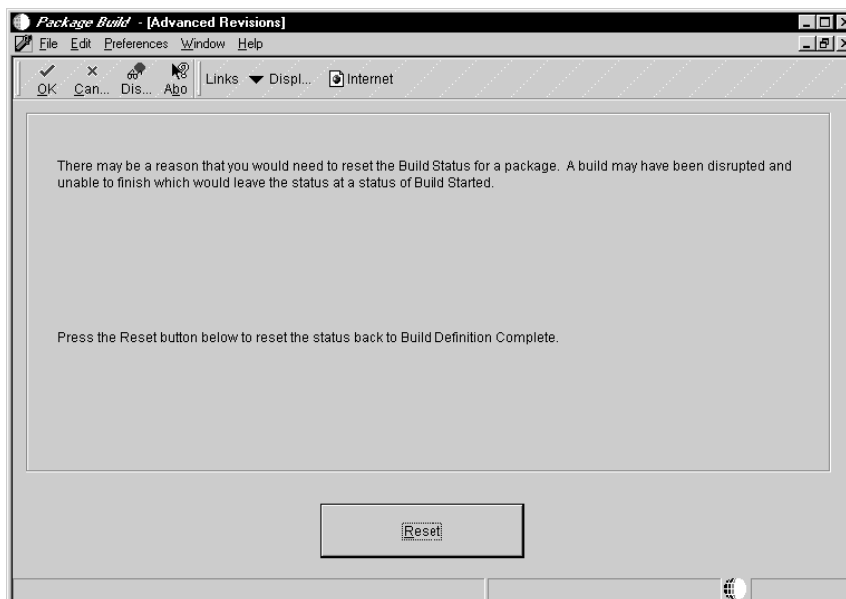
In some cases, you may need to attempt rebuilding the package rather than trying to resume the build from the point where the build failed. Before you can do so, you must first change the package's build status from Build Started to Build Definition Complete.

When you reset the package's build status, you have the option of resetting 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 whose status you want to reset. Underneath the package name, choose the server or servers and client workstation for which you want to build the package.
2. Choose Advanced from the Work with Package Build Definition form's Row menu.



3. On Advanced Revisions, click Reset to change the package's build status 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 Work with Package Build Definition form's Row menu.
6. The program will ask if you want to delete the current build or to continue without deletion.

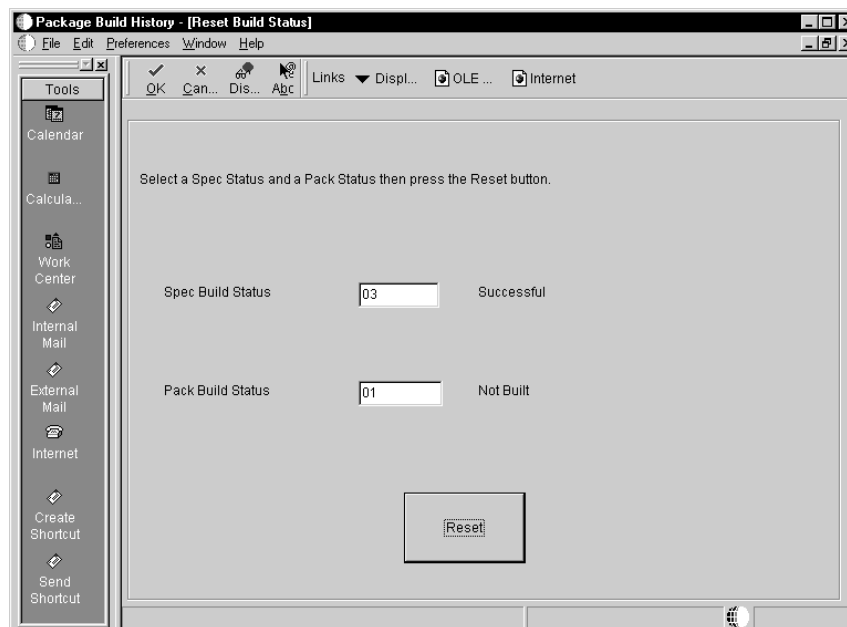
Changing the Spec Build and Pack Build Statuses

You can also reset a package's spec and pack statuses to the status you specify. This option is available through the Package Build History (P9622) application.

► 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 whose statuses you want to reset, 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 at 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 off of 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 PD7333FA, which is full package "A" for the PD7333 path code.

[Attributes]

This section contains information about the current OneWorld 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 the package is ready for installation.
PackageType=Full	Indicates the package type: full, partial, or update.
Release=B733	Indicates the current OneWorld 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 flag is always Y for full or partial packages. For update packages, this flag is Y only if there are objects in the package.
DelBlbTbl=Y	Indicates whether to delete global tables when installing. This flag will be set to Y for full or partial packages. For update packages, this flag is Y only if the objects include a table object.
ReplaceIni=Y	Determines whether to delete the existing jde.ini file when installing, and then create a new one. This flag 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 1998	Indicates the date and time the package was built. Full and partial packages always have this date. This is blank when there are no objects included in the package.
FoundationBuildDate=Wed Jun 03 15:08:34 1998	Indicates the date and time the foundation was built. For OneWorld Xe and later releases, this date is retrieved from owver.dll. Full and partial packages always have this date. This is blank when there are is no foundation location specified in the package.
DataBuildDate=Wed Jun 03 15:08:34 1998	Indicates the date and time the database file (jde7.mdb) was built. Full and partial packages always have this date. This is blank when there are is no database location specified in the package.
HelpBuildDate=Wed Jun 03 15:08:34 1998	Indicates the date and time the help (help.jz, the compressed help file,) was built. Full and partial packages always have this date. This is blank when there are is no help location specified in the package.
DeploymentServerName	Indicates the machine name of the deployment server.
Location	Indicates the location of the machine (deployment server).
DeploymentStatus=	Indicates the package deployment status.
PackageDescription=	Indicates the package description.
Icon Description	A description of the desktop icon.
Default Environment	Indicates the default environment.
Spec	Indicates the name of the spec file 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
<i>SPathcode</i> =\\ <i>MachineName</i> \ ONEWORLD\B7333\PACKAGE\ PD7333FA	Indicates the location of the package that the package build will compress for Workstation Installation. This path automatically defaults to the path code directory over which you built the package. You may change this setting if you want a different package pulled down.
<i>SSYS</i> =\\ <i>MachineName</i> \ ONEWORLD\B7333\SYSTEM	Indicates the location of the system directory that the package build will compress for Workstation Installation. This path automatically defaults to the system directory associated with the path code over which you built the package. Normally this would be under the release share name (B7333). This item appears only when included in the package.
<i>SPathcodeDATA</i> =\\ <i>MachineName</i> \ ONEWORLD\B7333\ PD7333\PACKAGE\DATA	Indicates the location of the Microsoft Access database that the package build will compress for Workstation Installation. This path automatically defaults to the data directory beneath the path code over which you built the package. This item appears only when included in the package.
<i>SHELP</i> =\\ <i>MachineName</i> \ ONEWORLD\B7333\HELPS	Indicates the location of the help files that the package build will compress for Workstation Installation. Users may not want to take up the disk space to have helps on their workstation. When you build a package this setting defaults to the help directory under the release share path (B7333). You may change this setting if you want to change the location in which helps will 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 machine configuration that is set up in user profiles (F00921) and Configuration Path Detail (F00051).

Item	Purpose
<i>DPathcode</i> =%INSTALL\ <i>path code</i>	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.
<i>DPathcode</i> DATA=%INSTALL\ <i>path code</i> \data	Indicates the destination directory for the database. This item appears only when included in the package.
DHELP=%INSTALL\helpers	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 under 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
<i>Pathcode</i> 1=Y, \$\$ <i>Spathcode</i> \bin32, \$D <i>pathcode</i> \bin32	Indicates the bin32 directory under the path code.
<i>Pathcode</i> 2=Y, \$\$ <i>Spathcode</i> \res, \$D <i>pathcode</i> \res	Indicates the res directory under the path code.
<i>Pathcode</i> 3=Y, \$\$ <i>Spathcode</i> \spec, \$D <i>pathcode</i> \spec	Indicates the spec directory under the path code.
<i>Pathcode</i> 4=Y, \$\$ <i>Spathcode</i> \include, \$D <i>pathcode</i> \include	Indicates the include directory under the path code.
<i>Pathcode</i> 5=Y, \$\$ <i>Spathcode</i> \lib, \$D <i>pathcode</i> \lib	Indicates the lib directory under the path code.
<i>Pathcode</i> 6=Y, \$\$ <i>Spathcode</i> \obj, \$D <i>pathcode</i> \obj	Indicates the obj directory under the path code.
<i>Pathcode</i> 7=Y, \$\$ <i>Spathcode</i> \source, \$D <i>pathcode</i> \source	Indicates the source directory under the path code.
<i>Pathcode</i> 8=Y, \$\$ <i>Spathcode</i> \work, \$D <i>pathcode</i> \work	Indicates the work directory under the path code.
<i>Pathcode</i> 9=Y, \$\$ <i>Spathcode</i> \make, \$D <i>pathcode</i> \make	Indicates the make directory under the path code.
<i>Pathcode</i> DATA=Y, \$\$ <i>Spathcode</i> DATA\data.CAB, \$D <i>pathcode</i> DATA	Indicates the compressed Microsoft Access database that the package build generates.

Item	Purpose
HELP=Y, \$SHELP\Helps.CAB, \$DHELP	Indicates the compressed helps file.

[Components]

These settings indicate the location of the foundation, production and development objects, the database, and 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 helps file location.

[Typical]

This section describes a typical development user's settings.

Item	Purpose
Name=Development	Indicates that the package is for a development user.
Description=Install the development objects	Indicates 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 a typical production user's settings.

Item	Purpose
Name=Production	Indicates that the package is for a production user.
Description=Install the production objects	Indicates 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.

[ODBC Data Sources]

This section describes the ODBC data source.

Item	Purpose
Access32=Microsoft Access Driver (*.mdb)	This line will be included in the ODBC data sources. This setting is determined by two registry settings on the build machine: LocalDS (the local data source name), and DataByPathCode (1=true, 2=false). If the DataByPathCode setting is set to 1 (true), the item name for this setting is <LocalDS> - <Pathcode>.

[Access32]

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. For example purposes, the local data source name used is Microsoft Access 32 database.

Item	Purpose
Driver=odbcjt32.dll	Indicates the name of the driver.
DefaultDir=C:\ACCESS	Indicates the default directory.
DriverId=25	Indicates the driver ID number.
FIL=MS Access	Required attribute setting for Microsoft Access.

JetIniPath=odbcddp.ini	Required attribute setting for Microsoft Access.
UID=admin	Required attribute setting for Microsoft Access.
Driver32=odbcjt32.dll	Indicates the ODBC driver for Microsoft Access.
DBQ=\$DAPPL_PGFDATA\jdeb7.mdb	Indicates the path code for the Microsoft Access database.
MaxBufferSize=2048	Indicates the maximum size of the buffer
Threads	Specifies the number of threads

[Features]

This section describes information for any features 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 what 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 whose sections contain specifications for each feature component.

[Header]

The Header section contains general information about the feature and determines the installation options for the feature.

Item	Purpose
	Feature. Name of the feature
Feature=	FeatureType. Type of feature.
FeatureType=	
Description=	Description. Text description of the feature.
Required=	Required. Determines if the installation of the feature is required.
InitialChoice=	InitialChoice. Determines 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 options settings (Required, Selected, Deselected) on the Feature Information form. When you select one of these three options, OneWorld 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 impacts the Windows registry.

Item	Purpose
Registry[no.]=Root[value],Key,[prefix]Name,[prefix]Value	Indicates the following specifications for the feature's Windows registry settings: 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 the name is created (if it doesn't already exist) when the feature is installed

- means the name is deleted with all subkeys when the feature is uninstalled

* means the name is created (if it doesn't already exist) when the feature is installed, and removed with all subkeys when the feature is uninstalled

Value. The registry value name's

value. Value prefixes are:

#x means the value is stored as a hexadecimal value

#% means the value is stored as an expandable string

means the value is stored as an integer

#\$ means the value is stored as a string

[INI]

This section contains information about the impact of the feature on the JDE.INI file.

Item	Purpose
Ini[no.]=FileName,Directory,Section,Key,Value,Action[value]	Indicates the following information pertaining 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. Indicates whether the fileset is compressed or not.
	Source Directory. The source location of the fileset.
	FileName. Name of the fileset's CAB file.
	Target Directory. The target location where the fileset is to be extracted.

[Shortcut]

This section contains information about shortcuts created 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	<p>Indicates the following specifications for the shortcut:</p> <p>Directory. The directory where the shortcut is created.</p> <p>Name. The name of the link file for the shortcut.</p> <p>Target. The shortcut's executable file name.</p> <p>Arguments. Any command line arguments for the shortcut.</p> <p>Description. A description of the shortcut.</p> <p>HotKey. A hot key that launches the shortcut.</p> <p>Icon. The shortcut icon and location.</p> <p>IconIndex. An index of the icon if the icon is inside an image list.</p> <p>ShowCmd. A command for the application window, with the following value options:</p> <ul style="list-style-type: none"> 0 means show the window normal sized 3 means show the window maximized 7 means show the window minimized; not active <p>WkDir. The shortcut's working directory.</p>

[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	<p>Indicates the following specifications for the third party applications:</p> <p>Source Directory. Source location of the executable</p>

for running the third party application.

Description. Description of the third party application.

Synchronous/Asynchronous. Indicates whether the installation of the third party application can occur in parallel (synchronous) or must install 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. The feature.inf contains one header section listing all data source components included in the feature. For each data source listed in the header there is a corresponding detail section. Only the header section is described below. Refer to the documentation that came with the selected ODBC Driver for information about the detail section.

Item	Purpose
DataSourceName=DataSourceDriver	Indicates the following specifications for the ODBC Data Sources: <p style="margin-left: 20px;">DataSource Name. Name of the ODBC data source.</p> <p style="margin-left: 20px;">DataSource Driver. The driver used for the data source.</p>

Deployment

After you have assembled, defined, and built a package, there are several ways to deploy the package to workstations and servers.

Topics

- ❑ Understanding package deployment
- ❑ Defining machines, locations, and deployment groups
- ❑ Working with the Package Deployment Director
- ❑ Deploying server packages
- ❑ Using Push Installation
- ❑ Installing OneWorld workstations from CD
- ❑ Deploying partial package templates

Understanding Package Deployment

After you build a package there are several methods available for deploying the package to workstations and servers throughout your enterprise. For workstations, the method you select depends on whether OneWorld is already installed on the workstation.

Topics

- ❑ Deployment to workstations without OneWorld
- ❑ Deployment to workstations with OneWorld already installed
- ❑ Deployment to servers
- ❑ Deployment to tiered deployment locations
- ❑ Deployment to workstations from CD

Deployment to Workstations without OneWorld

If a workstation does not currently have OneWorld installed, you can deploy the package through the Workstation Installation program. Workstation Installation is used to deploy only full and partial packages; you cannot use Workstation Installation to deploy an update package to a workstation that doesn't have OneWorld 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 describing from where to pull 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. For more information about the Workstation Installation application, see the *OneWorld Installation Guide*.

You can use Package Deployment to deploy the package if you use the push installation feature. Push installation enables the OneWorld administrator to initiate the installation of a package from the deployment server to workstations without requiring any user interaction. In order 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 application (P9654A).

Deployment to Workstations with OneWorld Already Installed

To reload a new package on workstations that already have OneWorld installed, use one of two methods:

- Workstation Installation (for full and partial packages)
- Package Deployment (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 scheduled to receive the package sign on to OneWorld, they are given the opportunity to load the package.

Unless you are using the Push Installation feature, Package Deployment requires that OneWorld 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.

There are advantages to both deployment methods. 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. On the other hand, 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 workstations.

Deployment to Servers

Servers receive the same package 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. Deployment is accomplished via the Package Deployment application, which uses the same scheduling mechanism used 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 for deploying 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 OneWorld 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. Doing this essentially defines the CD writer as a pseudo deployment server from which you can copy a OneWorld package onto a blank CD. You can then use this CD to install OneWorld on workstations by using the Workstation Installation program contained on the CD.

Defining Machines, Locations, and Deployment Groups

Before you deploy packages, you must define the workstations, servers, groups, or locations that will receive your package. Defining these ensures that when you are ready to schedule packages using the Package Deployment Director, the machines, groups, or locations you want to receive your package will be available as package recipients.

A deployment group is a group of workstations classified by a criterion such as job function, team, or any other grouping you specify. For example, you might have a software development group, a testing group, a production group, and so on. The Machine Group Identification application enables you to define or revise groups comprised of 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 have a location defined for each of your geographic locations. The Machine Identification application 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 OneWorld configuration. For example, a deployment server can be used as a data server. When you enter machine definitions, keep in mind 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.

Topics

- ❑ Defining machines
- ❑ Defining locations
- ❑ Defining package deployment groups
- ❑ Revising a package deployment group

Defining Machines

Before you use the Package Deployment Director to deploy a package to individual client workstations, make sure that each machine that will receive the package has a record in the Machine Master table (F9650).

There are two ways to populate this F9650 table:

- Manually. Use the Machine Identification application (P9650) to manually enter each new machine that has never signed on to OneWorld.
- Automatically. OneWorld automatically creates a record in the F9650 table when a user on a new machine signs on to OneWorld for the first time. (Existing records in the F9650 table are also updated each time a person signs on to the workstation.)

The simplest way to populate the F9650 machine master table is to have all users on new machines sign on to OneWorld. In cases where you need to deploy a package before this can happen, you must manually enter machine information as described in the following task. The Machine Identification application enables you to do just that.

In addition to defining workstations, you can also use the Machine Identification application to enter or revise definitions for the following machines:

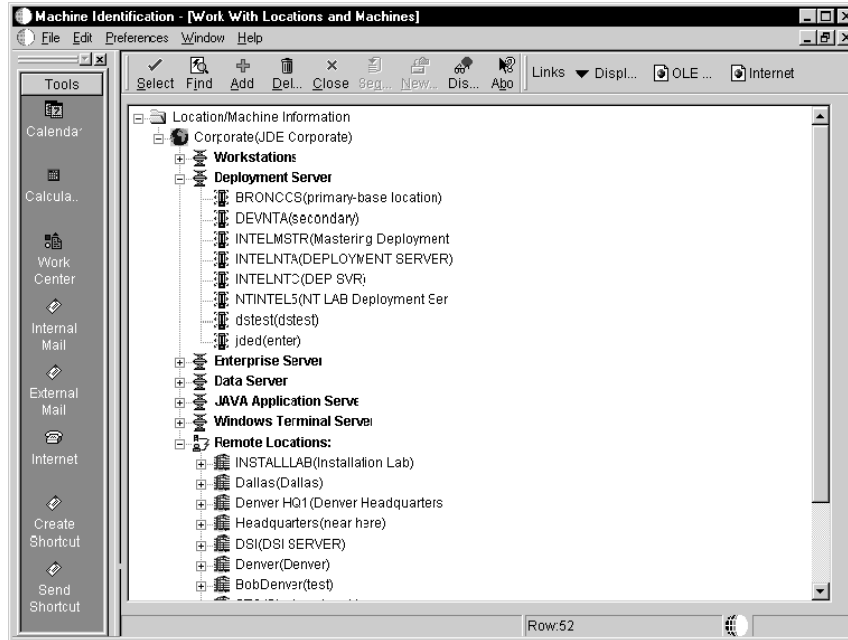
- Deployment servers
- Enterprise servers
- Data servers
- Java application servers
- Windows terminal servers

You can define a machine to be a workstation and a data server or as a data server only, but no other combinations are allowed. For example, you cannot define a machine as a workstation and a deployment server.

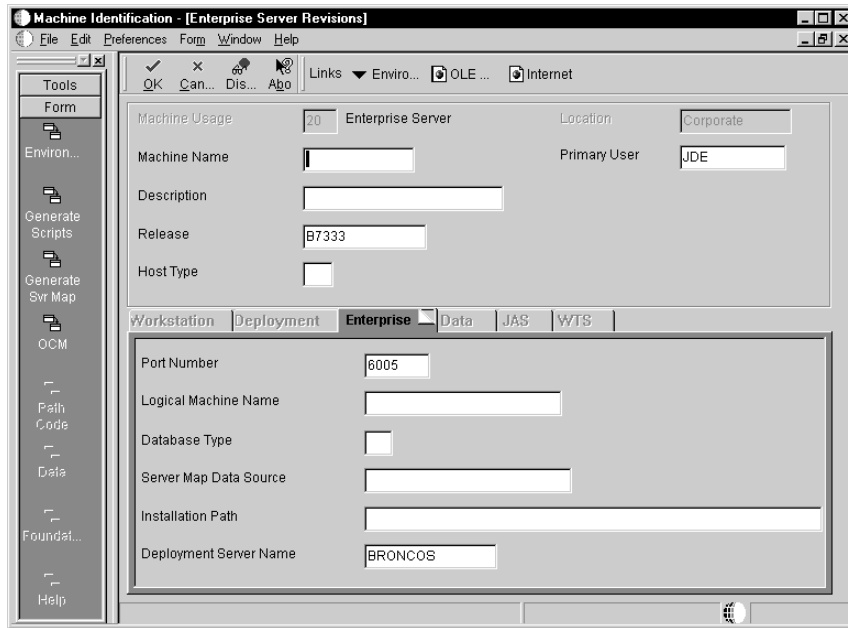
You can enter or revise definitions for these machines in multiple locations, including remote locations.

► **To define a machine**

From the Package and Deployment Tools menu (GH9083), choose Machine Identification (P9654A).



1. On Work With Locations and Machines, underneath the appropriate location, choose the type of machine you want to add:
 - Workstation
 - Deployment server
 - Enterprise server
 - Data server
 - Java application server
 - Windows terminal server
2. Click Add to add a new machine. The Machine Revisions form appears. This form displays different fields depending on the type of machine you are adding.



3. 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 (For example, AS/400, HP9000, Intel NT, and so on)
- Primary User (The name of the machine's primary user.)

4. The tab corresponding to the type of machine you are defining appears automatically. Enter the fields for the machine you are defining:

On the Workstation tab:

- Deployment Server Name (display-only field)

On the Deployment [Server] tab:

- Primary Deployment Server

If you have set up a primary deployment server, the Primary Deployment Server field will not be accessible when you define a new deployment server. This field can be modified only when you are revising the the primary deployment server definition, or when you change your primary deployment server to a secondary server. In this case you could specify a different server as your primary deployment server.

- Server Share Path

When you define a secondary deployment server, Form menu options enable you to select path codes, data items, foundation modules, and help items. (These options are not available for the primary deployment server.)

On the Enterprise [Server] tab:

- Port Number
- Logical Machine Name
- Database Type
- Server Map Data Source
- Installation Path
- Deployment Server Name

When you define an enterprise server, Form menu options 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.

Note that some of the fields may be already be defined and can not be changed.

On the Data [Server] tab:

- Data Source Type

On the JAS [Server] tab:

- Installation Path

On the WTS (Windows Terminal Server) tab:

- Installation Path

5. From the menu, choose Form - Environment.
6. On Machine Environment Revisions, click on the Environment column and select one or more environments to add to the new machine.
7. Click OK.
8. On Machine Revisions, click OK again to return to the Work with Locations and Machines form.
9. One 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 whose definition you want to revise and click

Select. The same revision form appears and allows you to change the machine's definition.

Field Descriptions

Field	Description
Machine Usage	The purpose for this machine. For example: database server, logic server, TSE server, and so on.
Machine Name	For World, the Location indicates the machine (server or client). For OneWorld, the Location or Machine Key indicates the name of the machine on the network (server or workstation).
Release	For World, the release number as defined in the Software Versions Repository. For OneWorld, the release number as defined in the Release Master.
Host Type	Host Machine Type.
Location	The name of the deployment location.
Primary User	Primary User for the listed Machine.
Server Map Data Source	The data source name.
Deployment Server Name	The name of the specific server that is being used for deployment.
Primary Deployment Server	This field specifies whether a deployment server is the primary deployment server for a specific location.

Field	Description
Database Type	The type of database.
Logical Machine Name	<p>The logical machine name assigned to this unique machine and port. A machine can be a workstation. Because you can have more than one instance of OneWorld running on a given machine, you must assign a logical machine name that identifies the unique physical machine name and port where this instance runs.</p> <p>J. D. Edwards recommends that the logical machine name represent the release and purpose of the machine, such as Financial Data Server-B73.3 or Distribution Logic Server-B73.3.</p>
Package Date / Time	The date and time a full package was deployed to a specific workstation.
Package Name	<p>A package describes where on the server to find the components you want to deploy to workstations. There are three package types:</p> <p>Full: Contains the full suite of OneWorld applications (all specifications).</p> <p>Partial: A minimum configuration of OneWorld. This package type allows users to load the desired applications at run-time rather than initially installing all applications.</p> <p>Update: OneWorld objects contained in this type of package are loaded after the workstation receives the package and the user signs on to OneWorld. If the update package includes objects without the corresponding specifications, old versions of the application are deleted from the workstation and replaced by the current version the next time the user accesses that application. Update packages are always deployed on the date and time specified by the OneWorld administrator.</p>
System Update Date / Time	The date and time the current system was deployed to the specified workstation.
Package Update Date / Time	This field represents the date and time of the most recent update package installation on the specified workstation.

Field	Description
Package Update Name	This field represents the name of the most recent update package that has been deployed to the specified workstation.
Port Number	Identifies the port for a given instance of OneWorld. Because the JDE.ini file controls the port to which a workstation will connect, for workstations this port number is for reference only.
Server Share Path	For World, the Server Share Path Field is used by the environment to determine the location of the current server. For OneWorld, this field indicates the shared directory for this Path Code. The objects that are stored on a file server will be found in this path.

Defining Locations

In some cases, an enterprise may span several buildings, cities, or even countries. In these situations it is often easier to 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 OneWorld terms, 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 could 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.

Once you have created a new location you can add workstations and servers for that location by defining the machine names associated with that location.

The topmost location that is displayed when you launch the Machine Identification application is the base location. You cannot change or remove this base location, but you can create or revise locations beneath it.

When you create a location beneath another location, the original location is known as the parent location, and the location beneath is known as the child location. For example, if you have a location called "Seattle" and then create a location called "Redmond" under Seattle, Seattle is the parent location and Redmond is the child location.

See Also

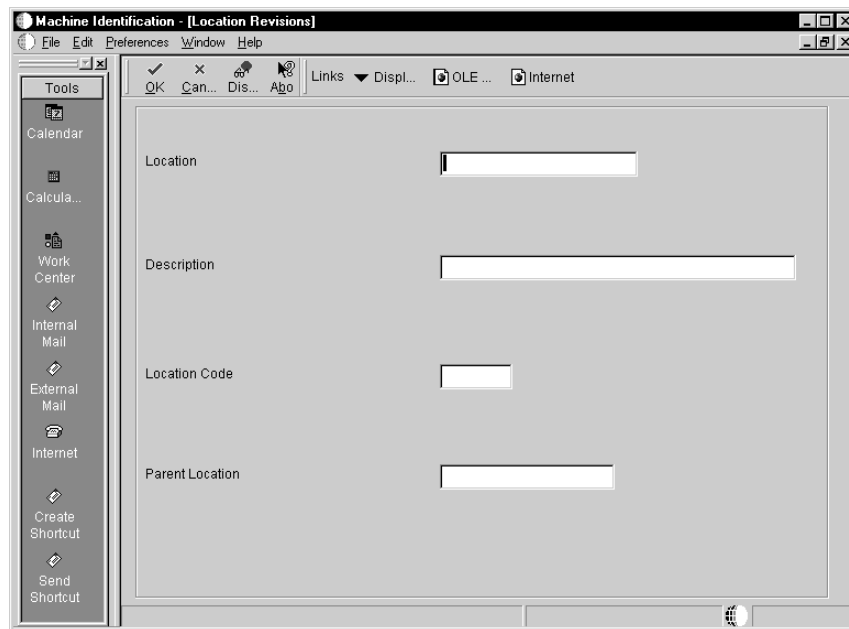
- [Multitier Deployment](#).

► To define a location

From the Package and Deployment Tools menu (GH9083), choose Machine Identification (P9654A).

On Work With Locations and Machines, select the current location under which you want the new location to reside. For example, if you are adding a new remote location, choose the Remote Locations icon.

1. Click Add to add a new location.



2. On Location Revisions form, enter 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 you want to revise and click Select. The same revision form appears and allows you to change the location.

Field Descriptions

Field	Description
Location	The name of the deployment location.
Location Code	This field represents the current location for OneWorld deployment.
Parent Location	The name of the parent 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.

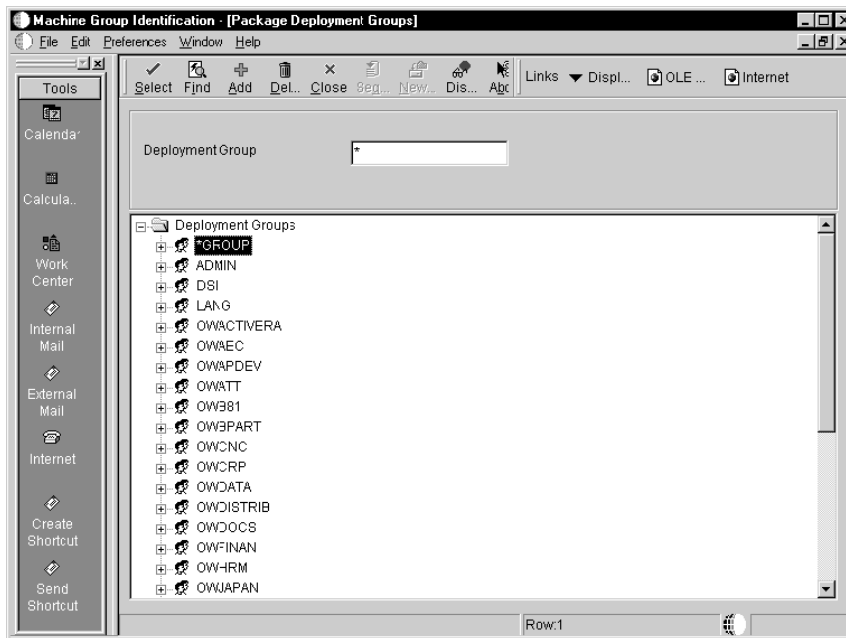
Package deployment groups are particularly useful in large enterprises where it would be very time-consuming to schedule a package for deployment to several individual workstations. In these environments it is much faster to create deployment groups.

A group can contain a subgroup (or group within a group). For example, you might have a group called "Quality Assurance" within a larger "Development" group.

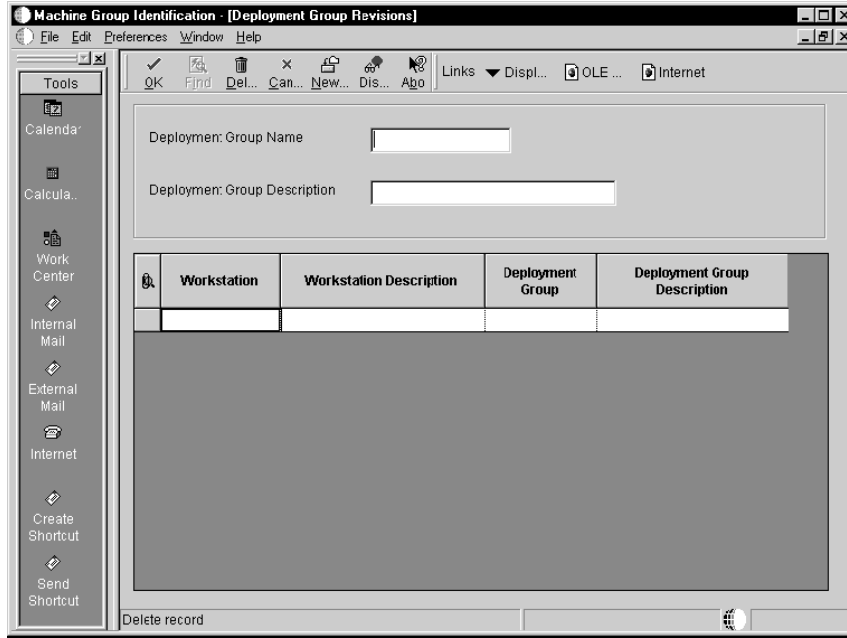
It will be helpful to the person who builds and schedules packages if deployment groups have easily identifiable names. For example, if you have a group composed of quality assurance specialists who are responsible for testing, it is better to name this deployment group "Testing" rather than "Green Team."

► **To define a package deployment group**

From the Package and Deployment Tools menu (GH9083), choose Machine Group Identification (P9652A).



1. On Package Deployment Groups, click Add to add a new deployment group.



2. Enter the deployment group name and description.
3. Enter the the workstations or deployment groups you want to include in the group. You can do this by manually typing the workstation name or group name in the Workstation field or Deployment Group field, or by using the visual assist feature.

When you use the visual assist at the Workstation field, the Machine Select form appears. When you use the Deployment Group field's visual assist, the Package Deployment Group Search form appears.

4. Click OK when you are finished adding workstations or deployment groups. You return to the Package Deployment Groups form.
5. Click Close to exit from the Package Deployment Groups form.

Field Descriptions

Field	Description
Deployment Group Name	<p>A profile used to classify users into groups for security purposes. Some rules for creating a User Class/Group are as follows:</p> <ul style="list-style-type: none"> o The 'Class/Group' profile must begin with * so that it does not conflict with any System profiles. o The 'User Class/Group' field must be blank when entering a new group profile.

Field	Description
Deployment Group Description	The description for the selected deployment group.
Workstation	For World, the Location indicates the machine (server or client). For OneWorld, the Location or Machine Key indicates the name of the machine on the network (server or workstation).
Workstation Description	A user defined name or remark.
Deployment Group	This field represents a group that is defined to be part of a parent group.

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 (P9652A).

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. You can do this by manually typing the name in the Workstation or Deployment Group field, or by using the visual assist at those fields.

When you use the visual assist at the Workstation field, the Machine Select form appears. When you use the Deployment Group field's visual assist, the Deployment Group Search form appears.

3. Click OK when you are finished adding or revising workstations or deployment groups. You return to the Package Deployment Groups form.

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 Package Deployment 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 are given the opportunity to load the package when they sign on to OneWorld.

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.

Topics

- Understanding the Package Deployment Director
- Using the Package Deployment Director
- Activating the scheduled package
- Installing a scheduled pacakge

Understanding the Package Deployment Director

The Package Deployment Director is designed to simplify and expedite the process of scheduling built packages to workstations and servers. The director displays a series of forms that allow you to specify the package you want to deploy, the deployment destinations, and the deployment time.

After specifying the package 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 method or the other.

You can make the package mandatory, which means users cannot access OneWorld until they have installed the package. If the package is optional, users will be given the option of installing the package every time they sign on to OneWorld until they either install or decline the package.

In addition, you can specify a "push installation," which means the package can be pushed from the deployment server to the workstations you specify, without requiring any interaction from the user. For more information on push installation, see [Using Push Installation](#).

The Package Deployment Director requires that OneWorld be already 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 Location Definitions (F9654)
- Package Deployment Information (F98825)
- Package Deployment on Servers (F98826)
- Software Package Header (F9603)

The following summarizes each of the Package Deployment Director's forms and their function:

Welcome form	View this form for a description of the Package Deployment Director.
Package Selection form	Use this form to find and select the package 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 you want the package deployed. 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. For more information about Push Installation, see Using Push Installation.
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 your package.
Deployment Groups Selection form	Use this form to specify the deployment groups whose members will receive your package.
Build Selection form	Use this form to specify the server or client package for multi-tier deployment 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 you entered through the Package Deployment Director.

Using the Package Deployment Director

Once 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 always have the option of going to the previous or next form by clicking Previous or Next. Also, regardless of where you are in the process, you can always cancel the deployment process by clicking Cancel.

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. Depending on your selection, different forms appear so you can enter information pertaining to 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 to only 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 according to one of the following four different ways: machines, deployment groups, locations, or packages.

Depending on your display selection, the tree displays different information as you expand the tree. The following describes the information displayed 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

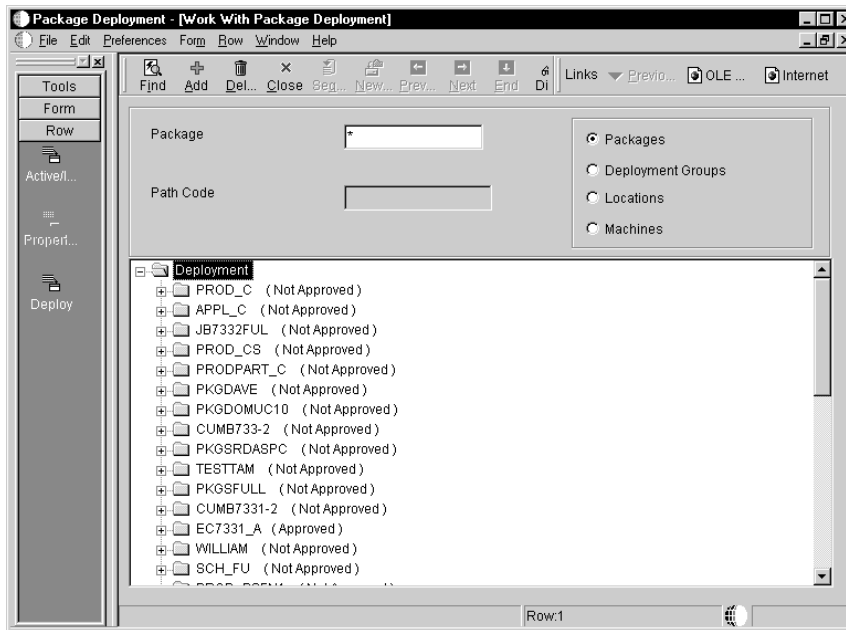
Level Two: Client Workstation, Deployment Server, and Enterprise Server headings

Level Three: Package deployment dates and times for each heading

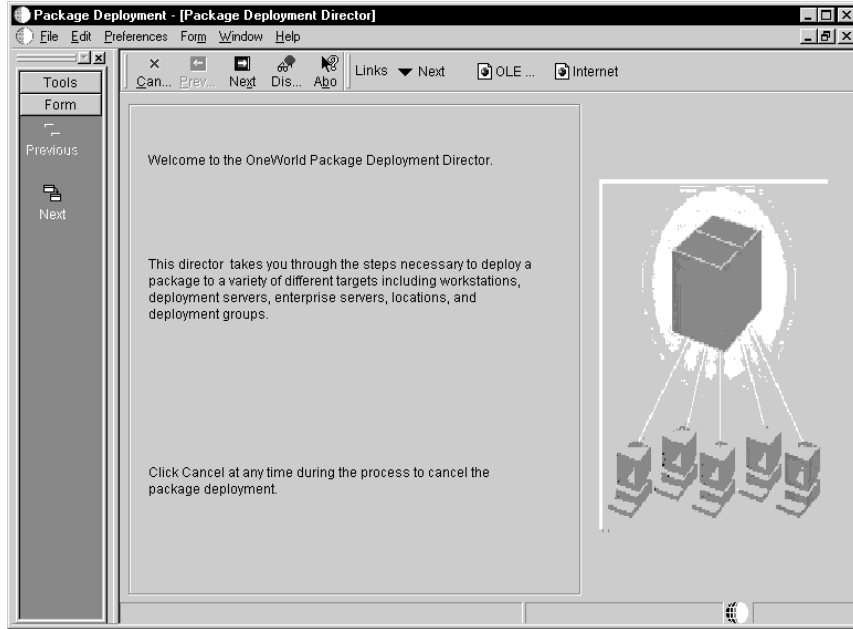
Level Four: Specific machines that have deployed that package for that date and time

▶ **To schedule a package for deployment to a client workstation or server**

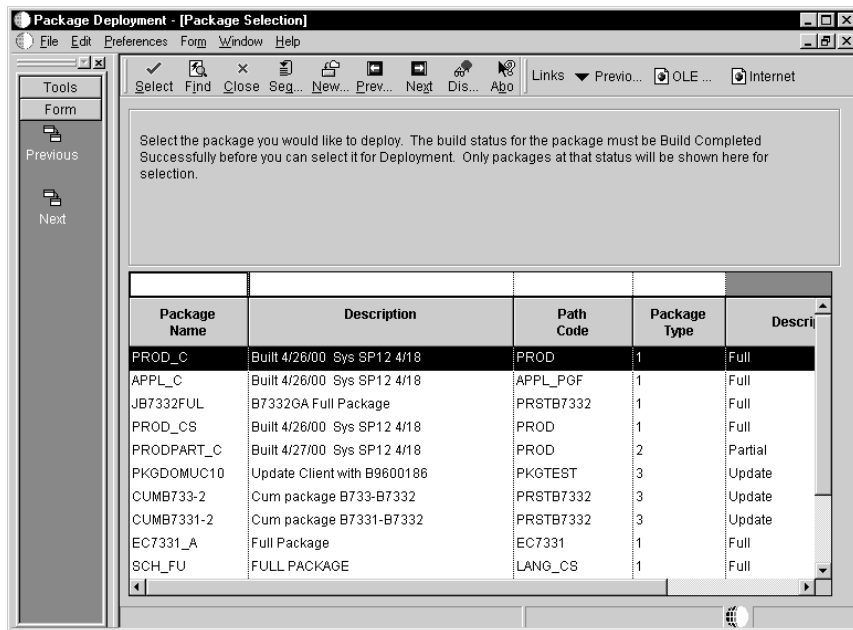
From the Package and Deployment Tools menu (GH9083), choose Package Deployment (P9631).



1. On Work with Package Deployment, click Add to launch the Package Deployment Director.

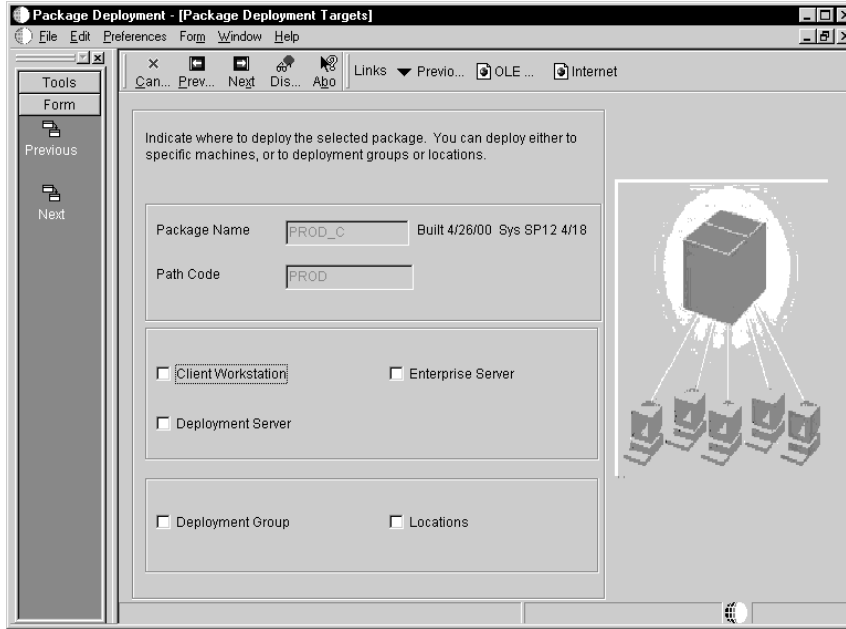


2. On Package Deployment Director, click Next.



3. On Package Selection, choose the package you want to deploy.

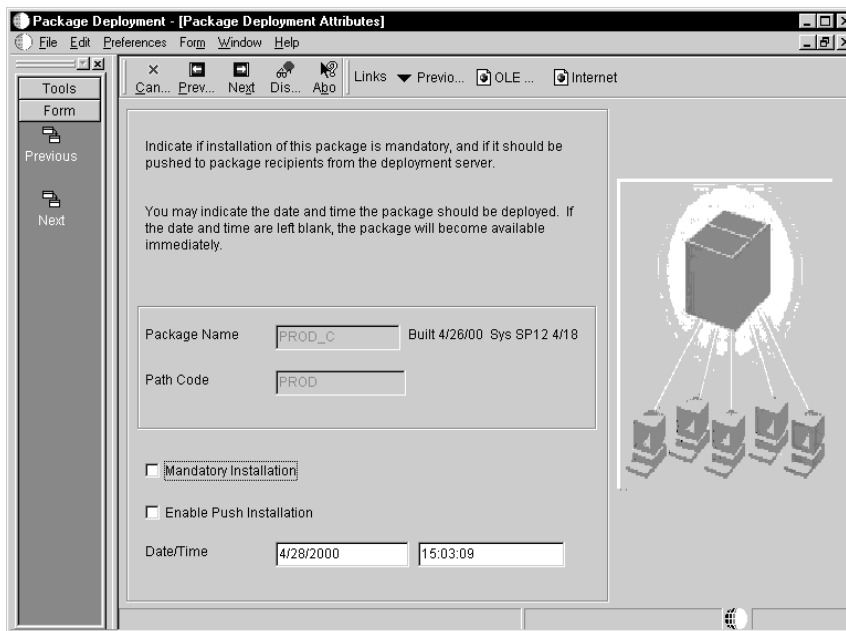
4. Click Next.



5. On Package Deployment Targets, indicate the type of machines to which you want to deploy your package:

- Client Workstation
- Deployment Server
- Enterprise Server

6. Click Next.

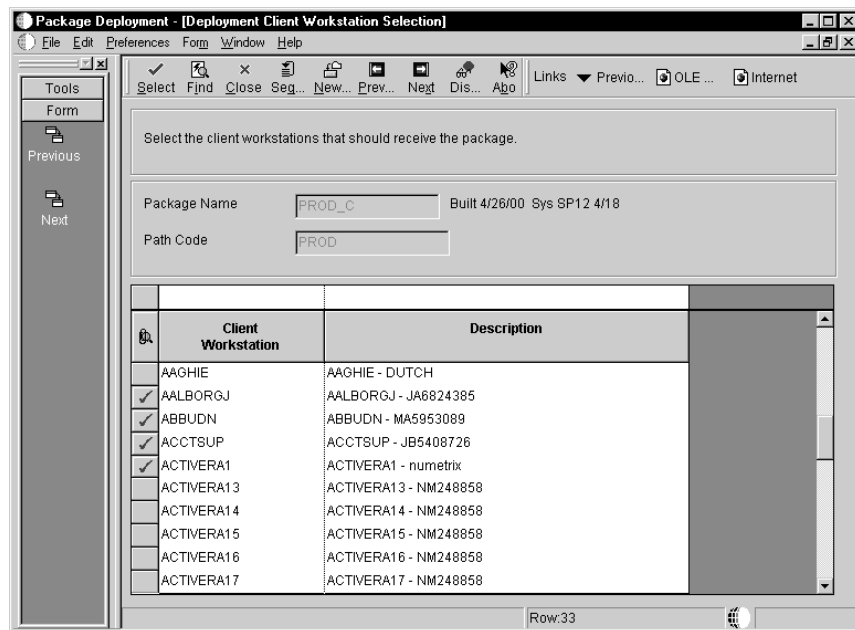


7. On Package Deployment Attributes, complete the following fields:

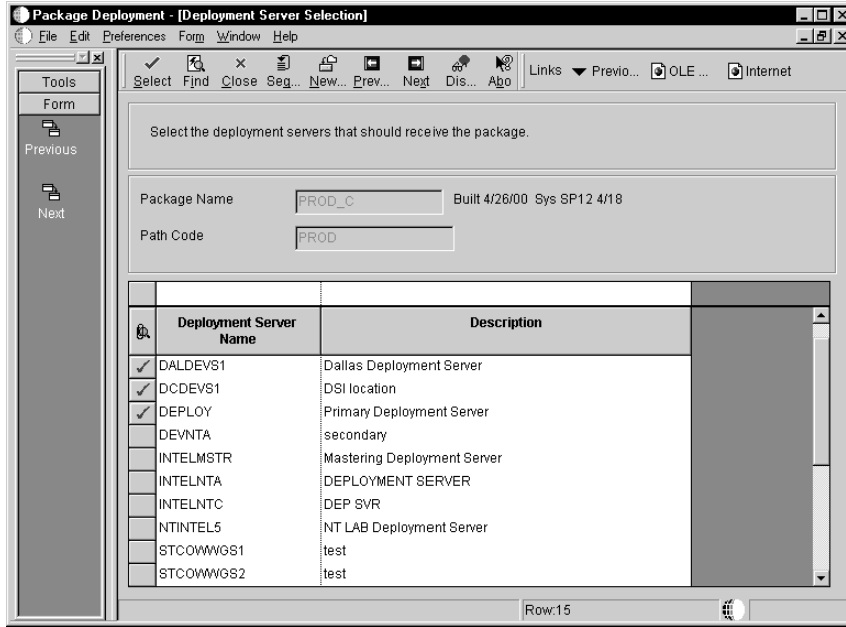
- Mandatory Installation
- Enable Push Installation
- Date/Time

Click the Enable Push Installation option if you want to deploy the package using push installation, which "pushes" the package to workstations from the deployment server. For more information about push installation, see [Using Push Installation](#).

8. Click Next. If you are deploying to workstations, the Deployment Client Workstation Selection form appears. If you are not deploying to workstations, skip the following step.

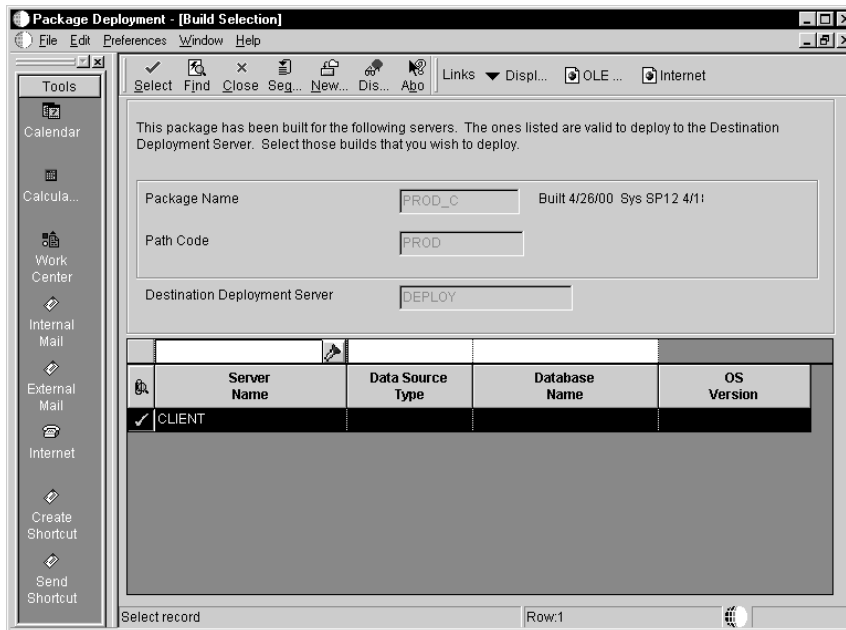


9. Find and choose the workstations to which you want to deploy the package. Choose a workstation by double-clicking in the column to the left of the Client Workstation column. A checkmark appears next to each workstation you choose. Click Next.
10. If you are deploying to a deployment server, the Deployment Server Selection form appears. If you are not deploying to a deployment server, skip this step.



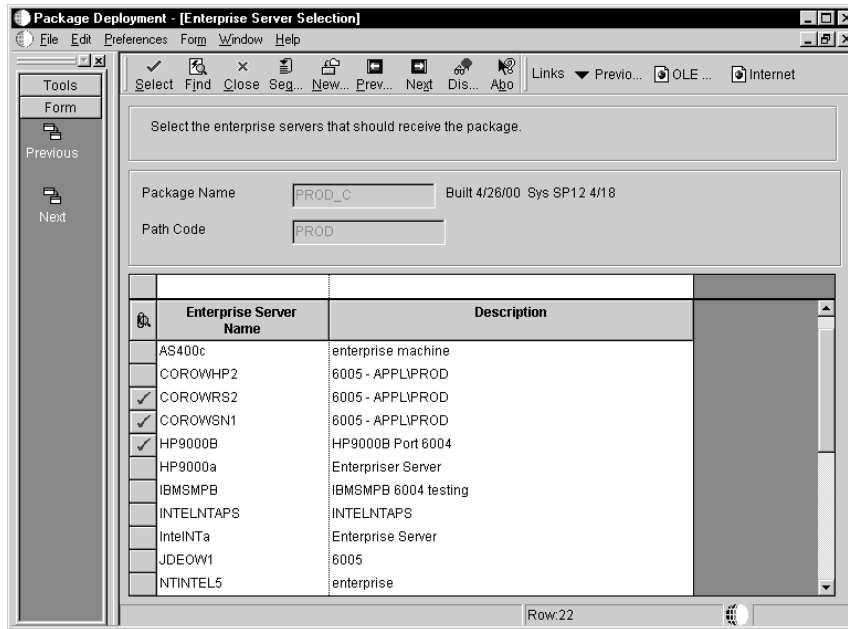
Find a the deployment server to which you want to deploy the package. Select a server by double-clicking the column to the left of the Deployment Server Name column. A checkmark appears next to each server you select. Click Next.

11. On Build Selection, select the server package build you want to deploy to the destination deployment server.



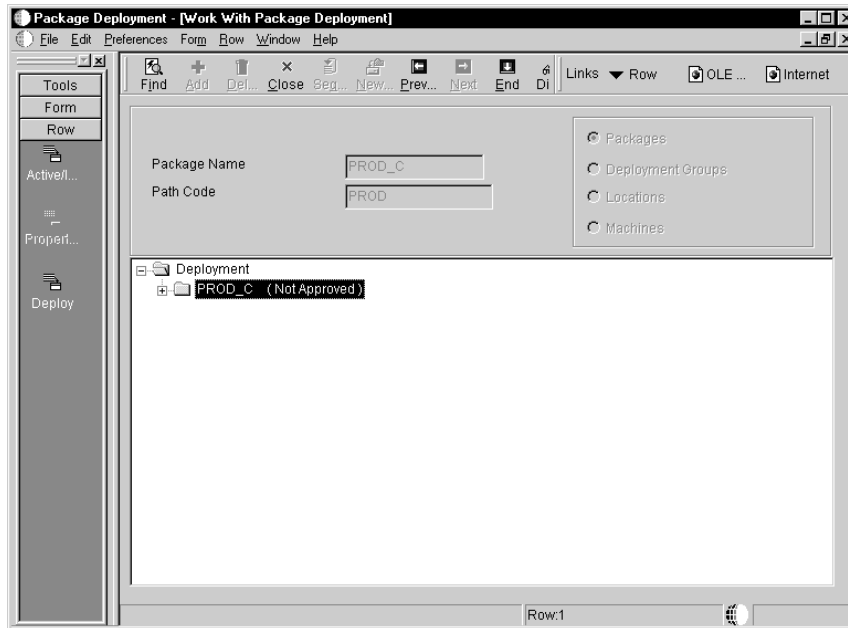
After you have selected a server package, click Close.

- 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 to the next step.



Find and choose the enterprise server to which you want to deploy the package. Choose a server by double-clicking in the column to the left of the Enterprise Server Name column, then click Next.

- On Work with Package Deployment, review your deployment selections. To change any of your selections, click Prev to return to the appropriate previous form.



14. When you are finished viewing or changing your deployment selections, click End.
15. If you are deploying a server package, you must find and choose the server package on the Work with Package Deployment form, and then choose Deploy from the Row menu. For more information about this step, see *Scheduling a Server Package for Push Installation*.

After you schedule your package for deployment, at the specified time on the date you specified, the package will be deployed to workstations. This package will be made available when the user signs onto OneWorld.

If you are using Push Installation, the package will be automatically installed at the time you specify through the Scheduler application. For more information, see [Using Push Installation](#).

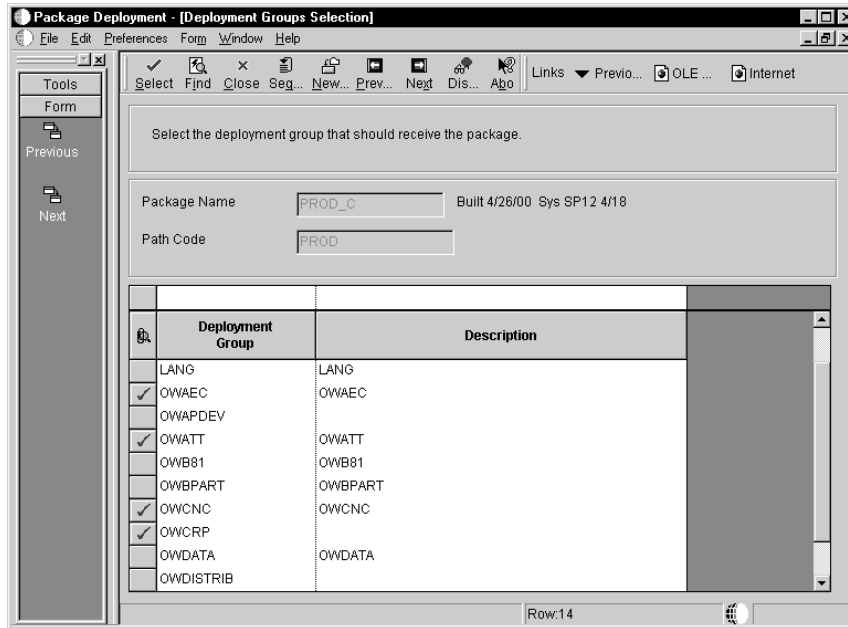
► **To schedule a package for deployment to a deployment group or location**

From the Package and Deployment Tools menu (GH9083), choose Package Deployment (P9631).

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 you want to deploy.
4. Click Next.
5. On Package Deployment Targets, choose either Deployment Group or Locations.
6. Click Next.
7. On Package Deployment Attributes, complete the following fields:
 - Mandatory Installation
 - Enable Push Installation
 - Date/Time

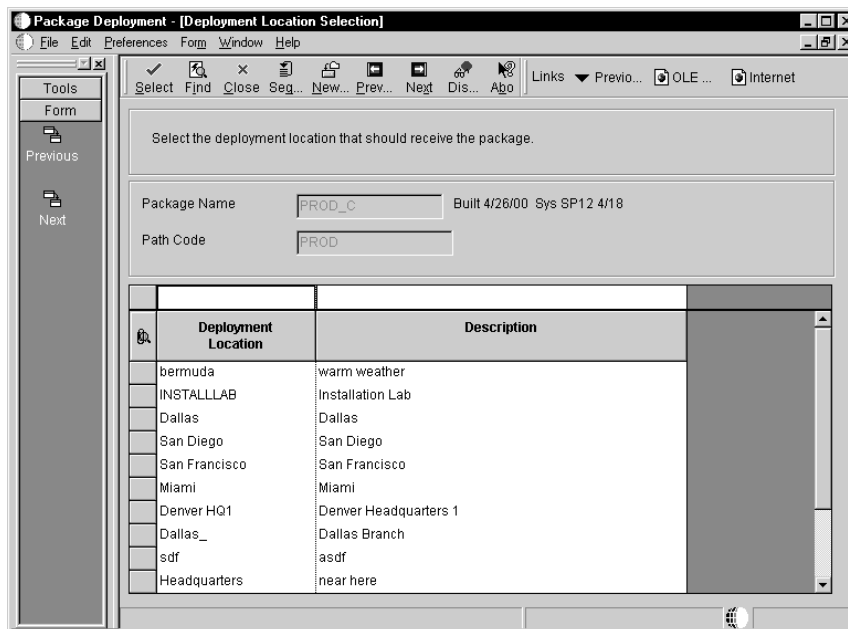
Click the Enable Push Installation option if you want to deploy the package using push installation, which "pushes" the package to workstations from the deployment server. For more information about push installation, see [Using Push Installation](#) in this guide.

- Click Next. If you are deploying to a location, skip this step.



If you are deploying to a deployment group, the Deployment Groups Selection form appears. Find and choose the deployment group you want to receive the package. Choose a group by double-clicking in the column to the left of the Deployment Group column. Skip the next step, which applies only to location deployment. Click Next.

- If you are deploying to a location, the Deployment Location Selection form appears.



Find and choose the deployment location you want to receive the package. Choose a location by double-clicking in the column to the left of the Deployment Location column.

10. Click Next.

On Work with Package Deployment, review your deployment selections. To change any of your selections, click Prev to return to the appropriate previous form.

11. When you are finished viewing or changing your deployment selections, click End.

12. If you are deploying a server package, you must find and choose the server package on the Work with Package Deployment form, and then choose Deploy from the Row menu. For more information about this step, see *Scheduling a Server Package for Deployment*.

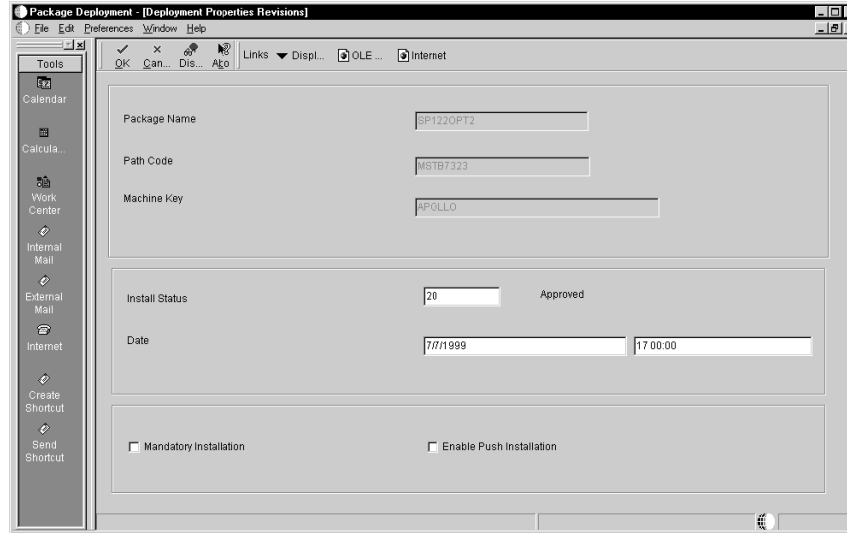
After you schedule your package for deployment, at the specified time on the date you specified, the package will be deployed to workstations. This package will be made available when the user signs onto OneWorld.

If you are using Push Installation, the package will be automatically installed at the time you specify through the Scheduler application. For more information, see [Using Push Installation](#).

► To revise deployment options

From the Package and Deployment Tools menu (GH9083), choose Package Deployment (P9631). The Work with Package Deployment form appears.

1. Choose Machines and click Find to display information according to machine name.
2. Find and choose the deployed package whose options you want to modify, and then choose Properties from the Row menu. The Deployment Properties Revisions form appears.



3. Revise 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.

Field Descriptions

Field	Description
Package Name	<p>For World, name identifying an install package.</p> <p>For OneWorld, a package describes the location on the server where components you want to deploy to workstations reside. There are three package types:</p> <p>Full: Contains the full suite of OneWorld applications (all specifications).</p> <p>Partial: A minimum configuration of OneWorld. This package type allows users to load the desired applications at run-time rather than initially installing all applications.</p> <p>Update: OneWorld objects contained in this type of package are loaded after the workstation receives the package and the user signs on to OneWorld. If the update package includes just-in-time applications, old versions of the application are deleted from the workstation and replaced by the current version the first time the user accesses that application. Update packages are always deployed on the date and time specified by the OneWorld administrator.</p> <p>With the exception of just-in-time applications included in an Update</p>

Field	Description
	package, all packages are a "snap shot" at a point in time of the central objects for a particular path code. Just-in-time applications are dynamic, not built.
Path Code	The path code is a pointer to a set of OneWorld objects, and is used to keep track of sets of objects and their locations within OneWorld.
Mandatory Installation	Indicates whether the package is mandatory or optional. Valid choices are: Y The deployment is mandatory. The user must install the package. N The deployment is optional to the user.
Enable Push Installation	A 1 in the field allows the package to be installed through Push Installation.
Date	Date to deploy updated objects to the listed machine.

Activating the Scheduled Package

After a package deployment is successfully defined, you must activate the package so 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.

There may also be situations where you 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 it is not available for installation through the Workstation Installation program. This package could then be scheduled for deployment to the members of the testing group through Package Deployment.

► **To activate a package**

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 the Packages button at the upper right of the form, and then click Find.
2. Choose from the list the packages you want to activate or inactivate. Alternatively, you can enter the package name at the Package field.
3. Choose Active/Inactive from the Row menu.

Installing a Scheduled Package

When users receive a package, they will be given the opportunity to install the package when they sign onto OneWorld on or after the scheduled deployment date.

If the package is mandatory, users will be unable to access OneWorld until they load the package.

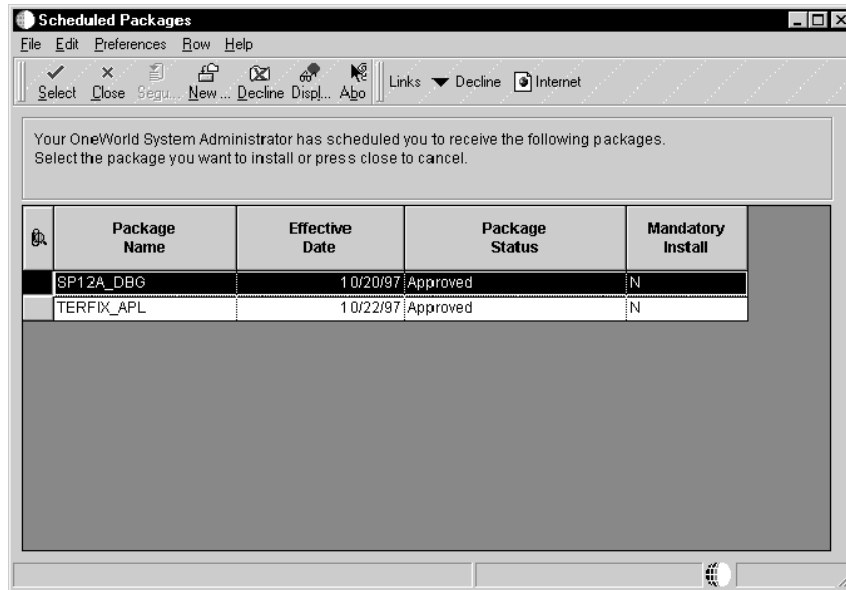
If the package is optional, users can decline the package or decide whether to load the package now or later. If they decide to load later, OneWorld Explorer launches, and they will be given the opportunity to install the package the next time they sign on.

If a package scheduled for push installation fails to load for some reason (such as if the scheduled user neglected to leave the workstation switched on during the time the package was scheduled to deploy), that package will be included in the list of available packages when the user signs onto OneWorld.

► **To install a scheduled package**

1. Sign on to OneWorld the same way you normally do.

When you are scheduled to receive a package, the Just-In-Time Installation program launches and the Scheduled Packages form.



2. Select one of the following options:
 - To load the package immediately, skip to the next step.
 - If you never want to load the package, click Decline from the row menu.
 - To list all items in the package, click 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 OneWorld 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, OneWorld 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, Package Build, and Package Deployment applications.

After you schedule a server package for deployment, you will need to perform an additional step to launch the batch application that enables you to deploy to servers. You must perform this task whenever you deploy a package to an enterprise server or deployment server. This task is described in [Scheduling a Server Package for Deployment](#).

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, OneWorld queues any jobs 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, make sure that the OneWorld services will be up and 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 not allowing developers to move their own objects through server packages. Instead, require that an administrator perform this function.

Server packages are deployed by choosing the Deploy function from the Work with Package Deployment form's Row menu. This is the same function used to deploy packages to deployment servers during multitier deployment.

OneWorld determines which batch applications 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, choosing Deploy launches the Multitier Deployment batch application (R98825C).

- If the Deployment Server folder is selected, choosing Deploy launches the Multitier Deployment batch application for every deployment server that has a package scheduled.
- If a specific enterprise server is selected, choosing Deploy launches the Enterprise Server Deployment batch application (R98825D).
- If the Enterprise Server folder is selected, choosing Deploy launches the Enterprise Server Deployment batch application for every enterprise server that has a package scheduled.
- If a specific workstation or the Workstations folder is selected, the Deploy option is unavailable.
- If a specific package is selected, choosing Deploy launches the Multitier Deployment batch application, and then the Enterprise Server Deployment batch application for the selected package.
- If you sort by packages and the Deployment folder selected, choosing Deploy launches both the Multitier Deployment batch application and Enterprise Server Deployment batch applications for all packages.

When a batch application launches for all servers or all packages, deployment does not occur unless the package has been previously scheduled for a specific server.

See Also

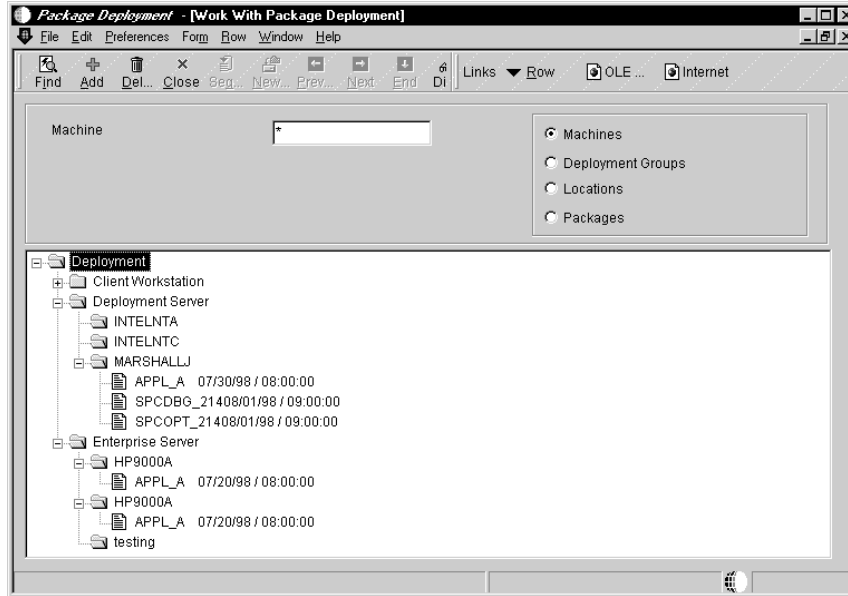
- [Assembling Packages](#)
- [Defining Package Builds](#)
- [Deploying Packages](#)

Before You Begin

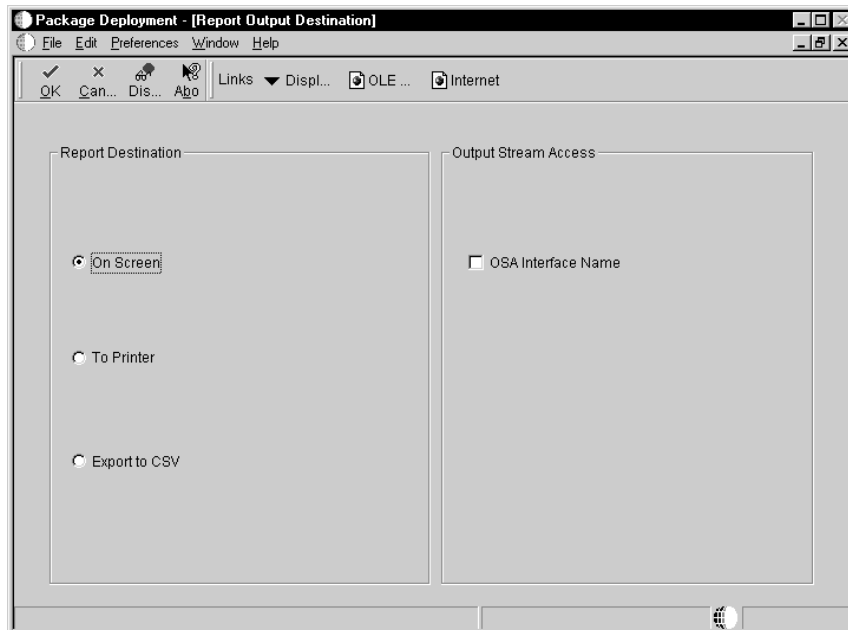
- ❑ Assemble the server package
- ❑ Define the server package
- ❑ Build the server package
- ❑ Schedule the package for deployment to the appropriate server

► To deploy a server package

From the Package and Deployment Tools menu (GH9083), choose Package Deployment (P9631).



1. On Work with Package Deployment, find and choose the server package you want to deploy. Alternatively, choose the enterprise server (or the Enterprise Servers folder if the package is scheduled to more than one server).
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 the OneWorld 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 a particular group for deployment after hours. When members of that group report to work the following morning, that package will be available for immediate use.

Push installation is particularly useful in situations where 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 OneWorld on a workstation for the first time. This ability can greatly minimize downtime and provide maximum deployment flexibility.

Topics

- ❑ Understanding push installation
- ❑ Preparing the enterprise server for push installation
- ❑ Preparing workstations for push installation
- ❑ Scheduling a package for push installation
- ❑ Scheduling the push installation batch application
- ❑ Running the Push Package Installation Results report

Understanding Push Installation

Push installation is so named because 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 set up to use Package Deployment for scheduled packages that are not "push enabled" also pull packages.

In terms of deployment, the end result 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 switched on during the time when the package is scheduled to deploy.

For a partial package or an update package that contains application 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:

5. Install a push installation "Listener" application on each workstation that will receive pushed packages. This Listener monitors the progress of the Push Installation batch application (R98825) running on the server and performs functions such as monitoring installation status. The Listener can run as either a local service or a network service.
6. Schedule the package using Package Deployment (P9631). The Push Installation batch application reads the scheduling table and sends a message to the Listener on all workstations for which a package has been scheduled.
7. Use the Scheduler application (P91300) to launch the batch application on the enterprise server. This application enables you to specify the job name, version, start date and time, and recurrence.
8. At the specified start time, the Scheduler launches the Push Installation batch application, which initiates the package installation process from the deployment server. The Listener and the batch application interact during the process until installation is complete. Codes are passed from the Listener to the batch application to indicate the installation status (for example, failed, successful, in progress, and so on).
9. When the installation is finished, OneWorld sends an e-mail message to the workstation's primary user. This message informs the user whether the installation was successful. E-mail notification works only if the package recipient is listed in the F9650 table and has an e-mail address in the profile.
10. If the push installation fails for some reason (for example, if the package recipient neglected to leave the workstation powered on), the installation status will change 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 OneWorld the standard scheduling screen appears. At this point the user can either accept the mandatory package or exit OneWorld.

Preparing the Enterprise Server for Push Installation

In order 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 don't already have a domain name service set up, you can install Microsoft DNS by selecting the Network icon in the Control Panel, selecting the Services tab, and then adding Microsoft DNS Server.

After you have added Microsoft DNS, you will need to 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 that is not running Windows NT Server 4.0 cannot resolve workstation addresses because the Windows NT server dynamically assigns them.

In order to enable name resolution you need to configure your servers to resolve their IP address lookup through a Windows NT DNS server, which must in turn be configured to look at 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:

4. From the host server on which the Push Installation batch application runs, a business function attempts to retrieve the machine host address by looking at the DNS server.
5. Because the DNS server does not contain IP addresses, it looks to the WINS server for the address.
6. The WINS server returns the address to the DNS server.
7. The DNS server returns the address to the host server.
8. The host server finds the Listener on the client workstation and sends workstation installation information.
9. 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 the batch application running 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 that already has OneWorld installed or if you are installing OneWorld 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 can be monitored by administrators 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 your enterprise's workstations, 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 logged 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 logged 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 Listener's parameters would apply to every user on the network, push installation users would be required to install OneWorld to and from the same locations. For example, one user could not have OneWorld Xe on drive C and another user have the same release on drive D. Also, every time users change their signon password the system administrator would have to 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 installing the Listener locally on each workstation.

Whether you run the Listener as a local service or network service, the workstation must be switched on to receive a scheduled package.

There are four main ways to install the Listener on a workstation:

- Using a third-party software distribution system such as the Tivoli Management Environment (TME10) Software Distribution System or the Microsoft System Management Server (SMS) software
- Distributing an executable installation program (a setup.exe file) and the accompanying ancillary files via an intranet Web site or the Worldwide Web
- Using Windows NT logon scripts (a .bat file) to call a C program
- Installing from the Worldwide Web

Caution

If the Listener is not installed and running on the workstation (or if the workstation is switched off) the push installation cannot occur. After you schedule a package, be sure to 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 logged on.

Before You Begin

Before you install the Listener on the workstation, J.D. Edwards recommends that you do the following:

- ❑ Close any applications currently open.
- ❑ Verify that the destination directory where you will be installing the Listener has sufficient disk space. You will need approximately 2MB of free disk space to install all of the Listener files and components.

► To install the Listener on workstations

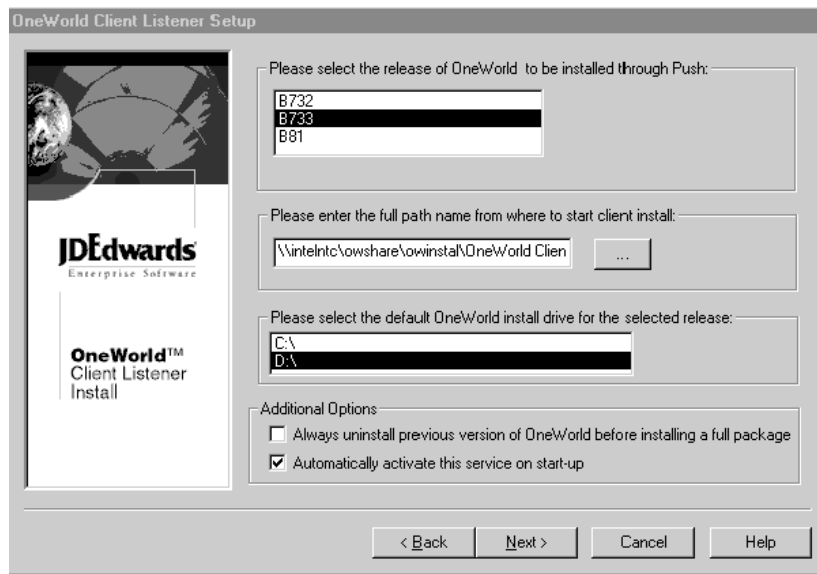
The following task describes how to install the Listener by launching an installation program (that is, a "setup.exe" program). You can distribute this program to users on your enterprise by sending them either the program or a shortcut using e-mail, or by pointing them to the program location on your 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 on the setup.exe icon.



2. On the Welcome form, click Next.

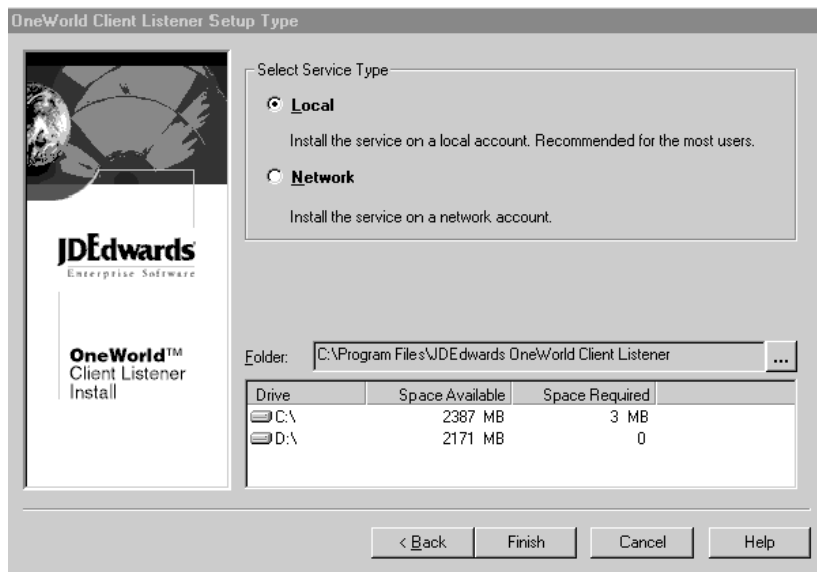


3. Complete the following fields:

- OneWorld release. Choose the OneWorld release you want to install through push installation
- Path name. For the OneWorld release you selected, enter the full path name on the deployment server from which to initiate the OneWorld installation
- OneWorld installation drive. Specify the drive where you want to install the specified release of OneWorld

- Uninstall previous releases. Turn on the uninstall OneWorld option to uninstall existing versions of OneWorld 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. Complete the following fields:

- Local or Network. Unless your system administrator tells you to install the Listener on the network, click Local to install the Listener on your local workstation.
- Folder. Specify the destination drive and folder in which the Listener files will reside.

6. 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 as explained in [Changing Default Parameter Settings](#).

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 OneWorld administrator enters configuration settings for all workstations and distributes a batch file that

automatically initiates the Listener installation the next time 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 each user is not required to enter configuration information or step through the installation process.

The following task describes how to install the Listener via silent installation. This task would typically be performed by the OneWorld administrator.

► **To Install the Listener using silent installation**

1. Edit the following settings in the `listen_silent_setup.inf` file that came 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 OneWorld base release. For example, B733. Do not enter a cumulative update release, such as B733.1.
 - **InstallPath**
Enter the location on the workstation where OneWorld is installed. For example, `D:\b7`.
 - **LaunchPath**
Enter the deployment server name and the location from which the Client Workstation Installation program runs. For example, `\\server name\b733\OneWorld Client Install\setup.exe`.
 - **AutoStart**
Enter 1 to automatically start the Listener service when the workstation starts up. Enter 0 if you don't want Autostart enabled.
 - **UninstallPackage**
Enter 1 if you want to automatically uninstall previous versions of OneWorld before installing a new full package. Enter 0 if you don't want automatic uninstall enabled.

2. Create or modify a batch file to include the silent installation parameter `"/s"` (without quotation marks) 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\b733\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 workstation to run the batch file and load the Listener via silent installation.

After workstation users have successfully installed the Listener, the Listener icon displays on the Windows taskbar. Users can click this icon to start and stop the Listener or change Listener settings as described in [Changing Default Parameter Settings](#).

Caution

You cannot change the name of the Listener silent installation file shipped with OneWorld. The filename must be `listen_silent_setup.inf`.

Stopping the Listener

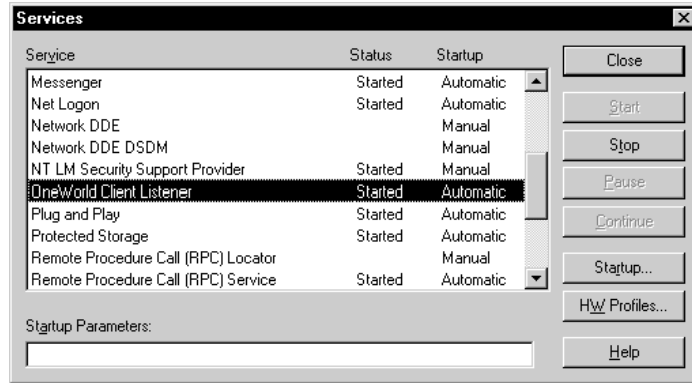
The only reason you would need to stop the Listener is if you are certain you do not want to use push installation to install OneWorld packages. If you change your mind later, you can always start the Listener again.

► 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 by following these steps:

1. Open the Control Panel.
2. Choose Services.



3. Choose OneWorld Client Listener.
4. Click Stop.

Uninstalling the Listener

► 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

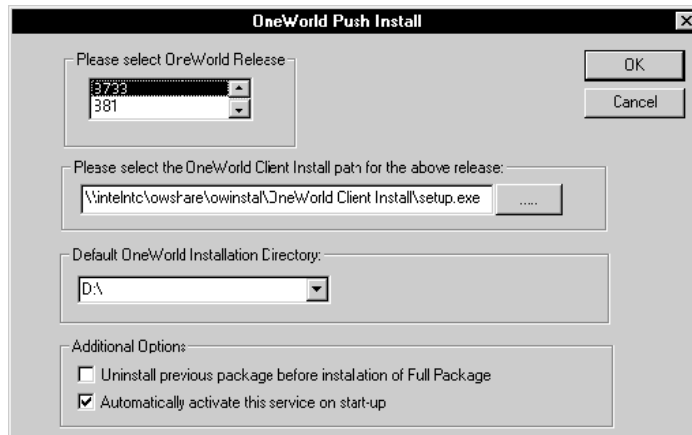
The Listener's "ear" icon that appears at the far right side of the Windows taskbar enables you to change the default parameter settings you entered when you initially installed the Listener. You can specify the following:

- The OneWorld release and client installation path for that release
- The default OneWorld 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 on the Windows taskbar.

- In the popup window, choose Change Default Parameters. The OneWorld Push Install form appears.



- Complete the following fields:
 - OneWorld release. Choose the OneWorld release you want to install through push installation
 - Path name. For the OneWorld release you selected, enter the full path name on the deployment server from which to initiate the OneWorld installation. If necessary, click on the button to the right of the field to browse for available paths.
 - OneWorld installation directory. Specify the drive where you want to install the specified release of OneWorld.
 - Uninstall previous releases. Indicate whether you want to uninstall previous packages before installing a full package. When you turn on this option, OneWorld completely uninstalls the existing package before installing 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.
- Click OK when you are finished making changes.

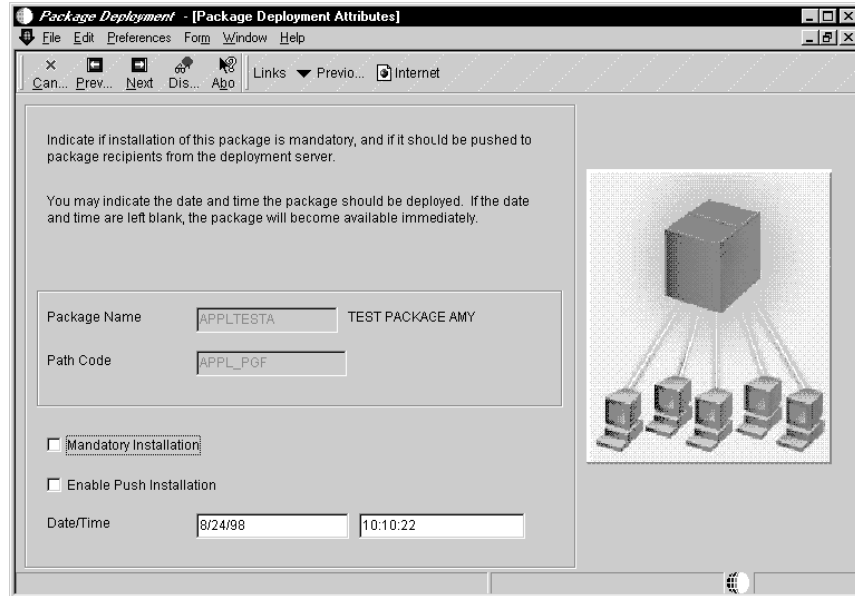
Scheduling a Package for Push Installation

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 scheduling a package using the Package Deployment application. When you schedule the package through this application, make sure to 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 Scheduler application, you can set recurrence, which determines how frequently the job runs until it completes 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. Once the package has been successfully deployed, the job does not keep running.



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 OneWorld. Alternatively, you can enter machine names manually via the Machine Identification application (P9654A). For more information about defining machine names and scheduling packages for deployment, see [Deployment](#).

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 application, you must use the Scheduler application to run the Push Installation batch application (R98825) on the server.

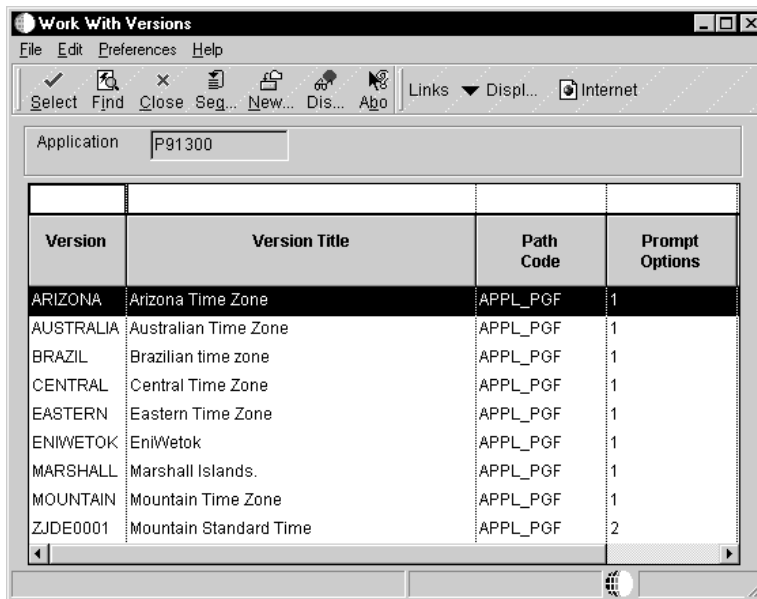
Before You Begin

Before you run the batch application, make sure you have done the following:

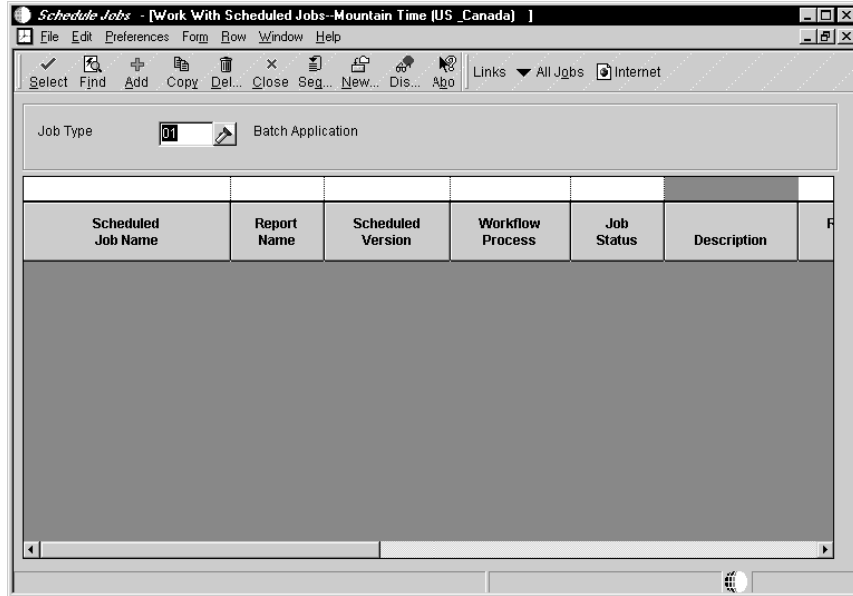
- ❑ Remind package recipients to leave their workstation running, even after hours.
- ❑ Remind users who are using a local service that they must be signed on.
- ❑ Remind package recipients to verify that the Listener is running.

► **To schedule the Push Installation batch application**

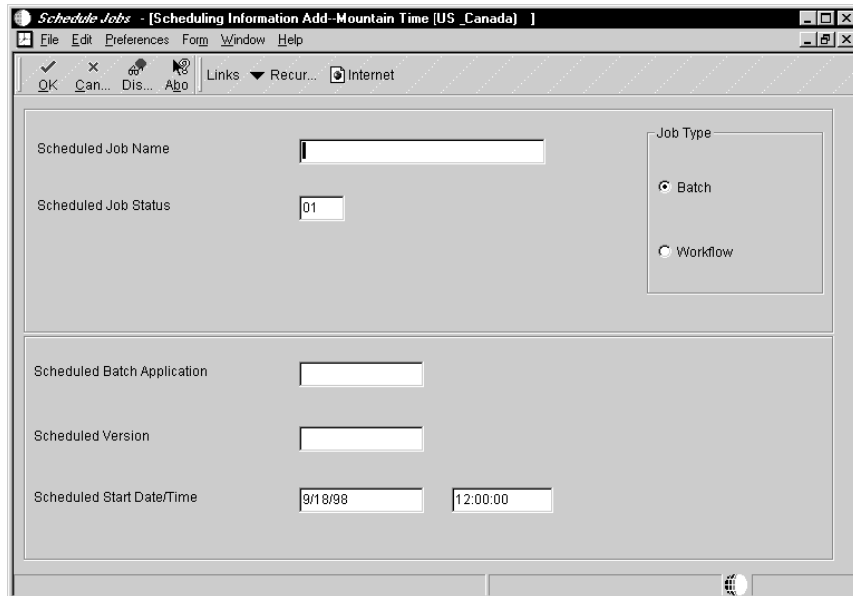
From the Job Scheduler menu (GH9015), choose Schedule Jobs (P91300).



1. On Work with Versions form, choose the time zone that applies to your setup.



2. On Work with Scheduled Jobs, click Add to enter a new job.

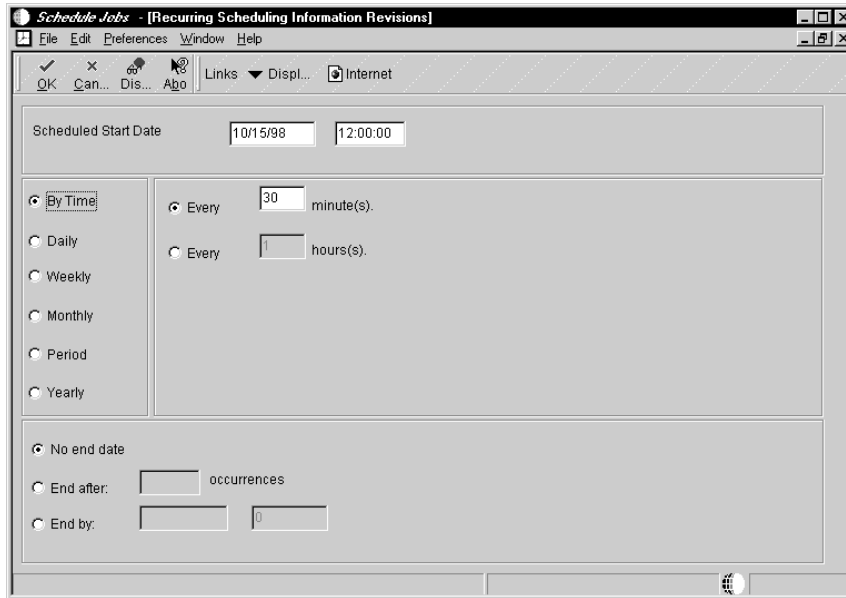


3. On Scheduling Information, click Batch as the job type and complete the following fields:

- Scheduled Job Name
- Scheduled Job Status
- Scheduled Batch Application
Type R98825 into this field
- Scheduled Version

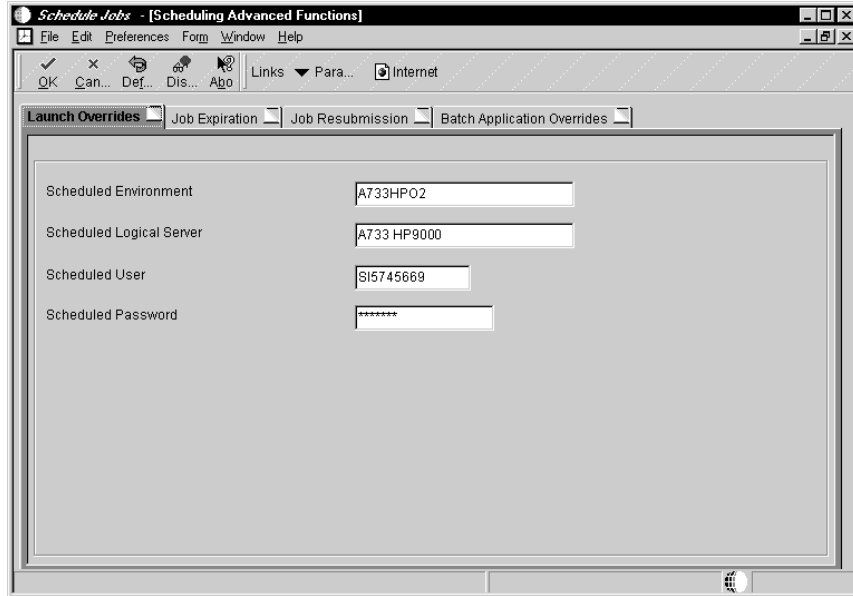
(for example, XJDE0001)

- Scheduled Start Date/Time
4. 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 you set recurrence for the Push Installation batch application to run every 30 minutes.

5. On Recurring Scheduling Information Revisions, click OK.
6. On Scheduling Information, to enter any overrides, resubmission, or expiration options, choose Advanced Functions from the Form menu.



7. On Scheduling Advanced Functions, click the tab corresponding to the information you want to enter or revise:

- Launch Overrides
- Job Expiration
- Job Resubmission
- Batch Application Overrides

For more information about using the Scheduler application, see *Scheduling Jobs* in the *System Administration Guide*.

8. Click OK.

After scheduling the job you may want to use Object Configuration Manager (P986110) to verify that the server on which the Push Installation batch application is running points to the same Package Scheduling (F98825) and Machine Master (F9650) tables that the Package Deployment application uses.

Field Descriptions

Field	Description
Scheduled Job Name	A name that uniquely identifies to the system and the user a scheduled job. Use this name to indicate the job function, such as Monthly Close or Nightly Back Up.
Scheduled Job Status	The current status of the scheduled job. As long as the status is active, the Scheduler determines if the job should be submitted to the server for

Field	Description
	execution. When the scheduled end date for the job has been reached, the status changes to Not Active. To stop the Scheduler from considering the job for submission, you can change the status to Not Active (or suspended) at any time prior to the end date. You can reactivate the job if you want the scheduler to include the job again, but you can reactivate a job only if the end date is in the future.
Scheduled Batch Application	The object name of the report that the Scheduler submits to the server.
Scheduled Version	The version of the report scheduled to run. A version identifies a specific set of data selections and sequencing settings the batch job uses.
Scheduled Start Date/Time	The next date on which the Scheduler submits the scheduled job to the server for execution.

Running the Push Package Installation Results Report

The Package and Deployment Tools menu contains a report called Push Package Installation Results (R98825B). This report shows the push installation results. This report essentially provides the same information you get when you run the Push Installation batch application (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)

You can run this report by choosing Push Package Installation Results (R98825B) from the Package and Deployment Tools menu (GH9083).

Push Installation Status Codes

The following lists the status codes and descriptions used by the Push Installation application. Codes marked with an asterisk indicate conditions in which the Push Installation application will continue attempting installation the next time the Push Installation batch application runs.

Status Code	Description
200*	Scheduled
210*	In Progress
220	Successful Install
230	Install Failed
240*	Install Running
250*	OneWorld Running
260*	Listener Not Started/Installed
270	General Error
280	Already Installed
290	Invalid Package
300	Install Attempted
310	Machine Down

Installing OneWorld Workstations from CD

If your system includes a CD writer (also known as a CD burner) 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 OneWorld Workstation Installation program.

This chapter outlines the steps necessary to set up your enterprise so you can deploy packages to the CD writer and install OneWorld from a CD.

Topics

- ❑ Defining the CD writer location
- ❑ Deploying a package to the CD writer location
- ❑ Creating the OneWorld installation CD

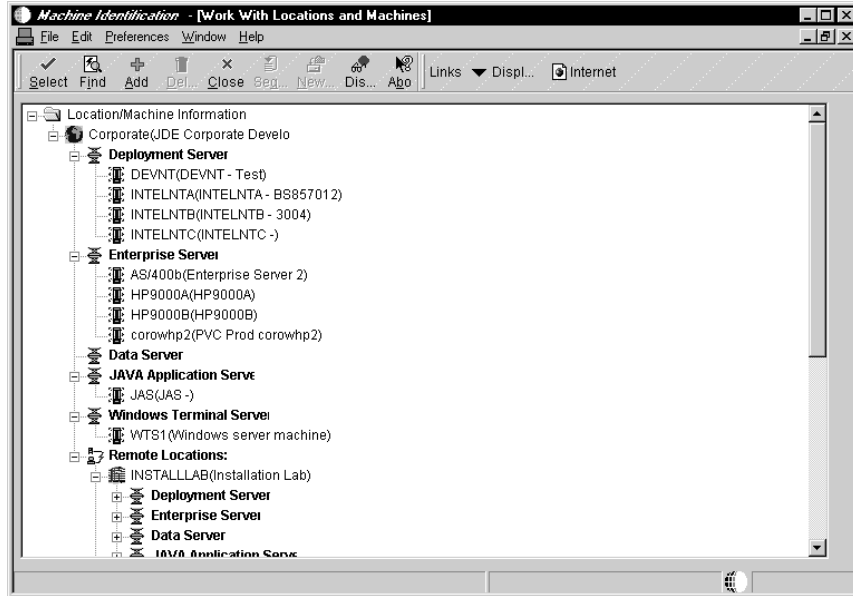
Defining the CD Writer Location

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 are essentially creating a pseudo deployment server from which you will later copy package data onto the CD by using your CD writer's software.

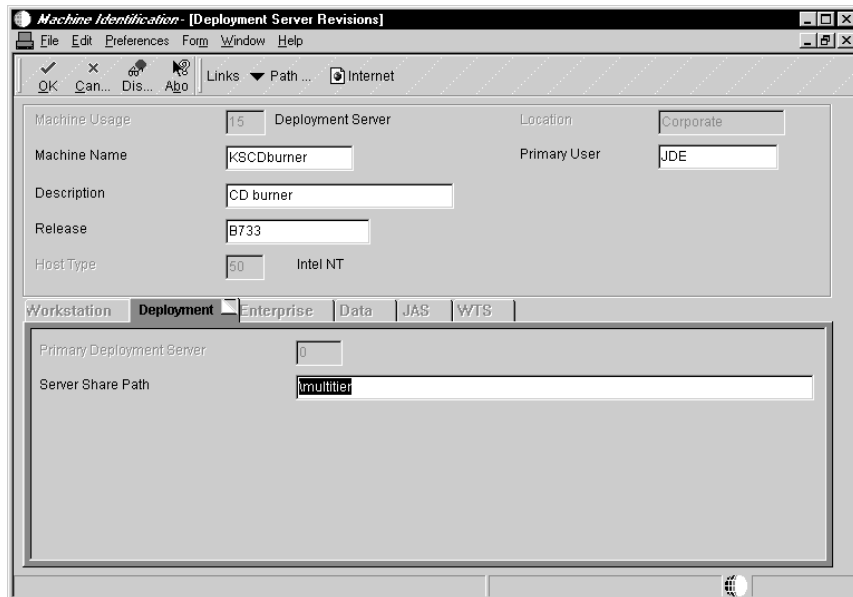
The process for defining this location is identical to defining any other new deployment server. For more information about adding a new server definition, see [Defining Machines](#).

► To define the CD writer location

From the Package and Deployment Tools menu (GH9083), choose Machine Identification (P9654A). The Work With Locations and Machines form appears.



1. Underneath the appropriate location, choose Deployment server.
2. Click Add to add a new machine. 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
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 default locations will be used.
 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 OneWorld Client Install. Copy this folder by dragging the folder to the CD writer location. (The location is the server share path you entered on the Deployment Server Revisions form.)

Field Descriptions

Field	Description
Machine Name	For World, the Location indicates the machine (server or client). For OneWorld, the Location or Machine Key indicates the name of the machine on the network (server or workstation).
Release	For World, the release number as defined in the Software Versions Repository. For OneWorld, the release number as defined in the Release Master.
Primary User	Primary User for the listed Machine.
Server Share Path	For World, the Server Share Path Field is used by the environment to determine the location of the current server. For OneWorld, this field indicates the shared directory for this Path Code. The objects that are stored on a file server will be found in this path.

Deploying a Package to the CD Writer Location

After you have defined the CD writer as a deployment server, you are ready to deploy a package to the CD writer location you specified. This task involves two procedures:

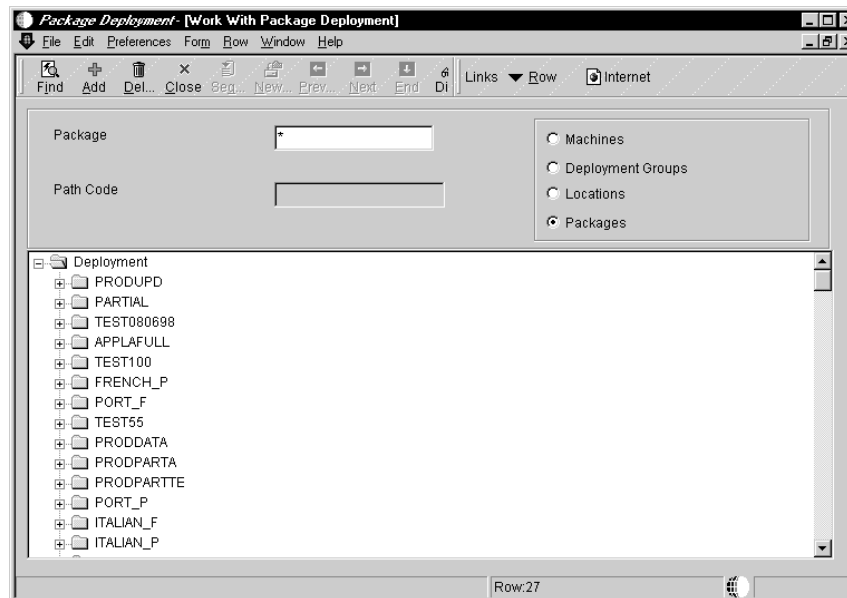
- Deploy to the CD writer location the package you want to write to the CD. For more information about deploying packages, see [Deployment](#).
- Modify the Install.inf and Package.inf files in preparation for writing the package contents to the CD.

Before You Begin

- Assemble, define, and build the package you want to write to the CD as explained in [Package Build](#).

► To deploy the package to the CD writer location

From the Package and Deployment Tools menu (GH9083), choose Package Deployment Scheduling (P9631). The Work With Package Deployment form appears.



1. Click Add. The Package Deployment Director launches.
2. Complete the Deployment Director's forms as you would for any other package. For more information about deploying packages, see [Working With the Package Deployment Director](#).

3. From the Work with Package Deployment form, find and choose the package you just scheduled for deployment. 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\helpscmp\helps.cab
Multitier\Appl_pgf\Package\Appl_B
Multitier\OneWorld_Client_Install
```

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. Using the previous example, the items copied to the CD would be \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 containing the package you deployed. This folder will have the name you entered at 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 was "Multitier.")
2. Open the folder OneWorld Client Installation, and then open the file Install.inf. That is, double-click the file's icon to launch the Microsoft Notepad application.
3. Under 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 modification:

```
[FileLocations]
PackageInfs=..\package_inf
```

4. Similarly, open the Package_inf folder and open the *package name*.inf file. In the previous example, this file would be named Appl_b.inf.
5. Under 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 OneWorld Installation CD

After you have deployed the package to the CD writer location and modified the Install.inf and Package.inf files, you are ready to copy the package contents to the CD. This process is accomplished with the software that came with your CD writer, and 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 beneath 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 was called "Multitier" and is the same name you entered at 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 database cabinet file)
- helpscomp (contains helps cabinet file)
- systemcomp (contains foundation cabinet file)
- package_inf (contains the package.inf file)
- OneWorld Client Install (contains the Workstation Installation program)

Note:

Actual names may not be the same as those listed, since there may be differences for each system.

Deploying Partial Package Templates

OneWorld users who use laptop computers may find that a full package requires too much disk space. Also, many of these users never need to use all of the OneWorld applications contained in a full package. To accommodate these users, J.D. Edwards has devised a solution that enables OneWorld administrators to create and deploy a partial package *template* specifically designed for laptop computer users. A partial package template is based on the standard OneWorld partial package, and also contains the objects necessary for laptop computer users to run specific applications.

Because each situation will be unique and the needs of every company different, the actual components put into the partial package template will vary. For example purposes, this chapter describes how to create a partial package template for the Sales Force Automation application.

You create and deploy a partial package template by using the same tools you would use to create any other partial package: the Package Assembly Director, Package Build Definition Director, and Package Deployment Director.

Topics

- ❑ Assembling a partial package template
- ❑ Building a partial package template
- ❑ Deploying a partial package template
- ❑ Testing a partial package template
- ❑ Package build time comparison

Assembling a Partial Package Template

The first step in creating a partial package template is to use the Package Assembly Director 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 users will need to run the application 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). The Work with Packages form appears.

1. Click Add to assemble a new package. The Package Assembly Director launches.
2. From the Welcome screen, click Next. The Package Information form appears.

Complete the following fields:

- Package Name
 - Description
 - Path Code
3. Click Next. The Package Type Selection form appears.

Click the Partial radio button to indicate that you are assembling a partial package.

4. Click Next. The Foundation Component form appears.

On this form and the following two forms (Help Component and Database Component,) accept the displayed default location by clicking Next, or click Browse to specify another location for the foundation, help file, or database you want to use.

For more information about entering a foundation, help location, or database location, see [Entering a Foundation Location](#), [Entering a Help Location](#), and [Entering a Database Location](#).

5. Click Next. The Object Component form appears. This form allows you to specify the individual objects you want to include in your package.
6. Click Browse to display the Object Component Selection form. Use this form to locate and select the objects you want to include in your package. These are the components that are necessary for users to run the application for the package template you are creating.

The objects 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 would 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

For more information about adding objects, see [Adding Objects to a Package](#).

7. When you are finished adding objects, click Close to return to the Object Component form.
8. 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.
9. Click Next to display the Package Component Revisions form. This form allows you to see at a glance the current foundation, help, and database locations, as well as the objects in the package and any language selection.

To change any of these package components, click the icon for the component you want to change. The form for that package component appears.

10. When you are finished assembling the package, click End to exit from the Package Assembly Director.
11. From the Work with Packages form, choose the package you just created. Activate the package by choosing Active/Inactive from the Row menu.

Building a Partial Package Template

After you have assembled the partial package you are ready to define the package build using the Package Build Definition Director. For more information about defining package builds, see *Defining Package Builds*.

► To build a partial package template

1. Access the Package Build Definition Director from the Work With Packages form by selecting the partial package you just assembled, and then choosing Build Director from the Row menu. The Package Build Definition Director form appears.

If the package's definition is still In Definition, before you can build the package you must change the status to Definition Complete. To do so, choose the package and choose Activate from the Row menu.

2. Click Next. The Package Build Location form appears. Since you are building a partial package, you will not be allowed to build the package for a server.
3. Click the Client checkbox to build a package for workstations.
4. Click Next. The Build Specification Options form appears.
5. Check the Build Specification Options checkbox.
6. Click Next. The Business Function Options form appears.

If a full package has been built recently, it is not necessary to check the Build Functions Options checkbox.

7. Click Next. The Compression Options form appears.

Check the Compress Options checkbox.

8. Click Next. The Package Build Revisions form appears.

This form allows you to see at a glance the current build options, business function options, and compression options you have specified for the package. To change any of these options, click the tab for the type of option you want to change.

9. 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.
10. Choose the partial package and choose Active/Inactive from the Row menu.
11. When you are ready to initiate the package build, choose Submit Build from the Row menu.
12. Choose a destination for the build report, and click OK:

- On Screen

- To Printer

The form automatically closes and OneWorld begins building the package. Build time varies, depending on the number and size of the items in your package. When the build is finished, the report will either display or print, depending on the destination you specified.

Deploying a Partial Package Template

After you have created 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 deploying any other package. For more information about deploying packages, see the *Deployment* section in this guide.

► To deploy a partial package template

From the Package and Deployment Tools menu (GH9083), choose Package Deployment (P9631). The Work with Package Deployment form appears

1. Click the Packages radio button, and then click Find to locate the partial package template.
2. Select the package 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 you want to deploy, and then complete the Package Deployment Director's remaining forms. These forms enable you to do the following:
 - Specify the package 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

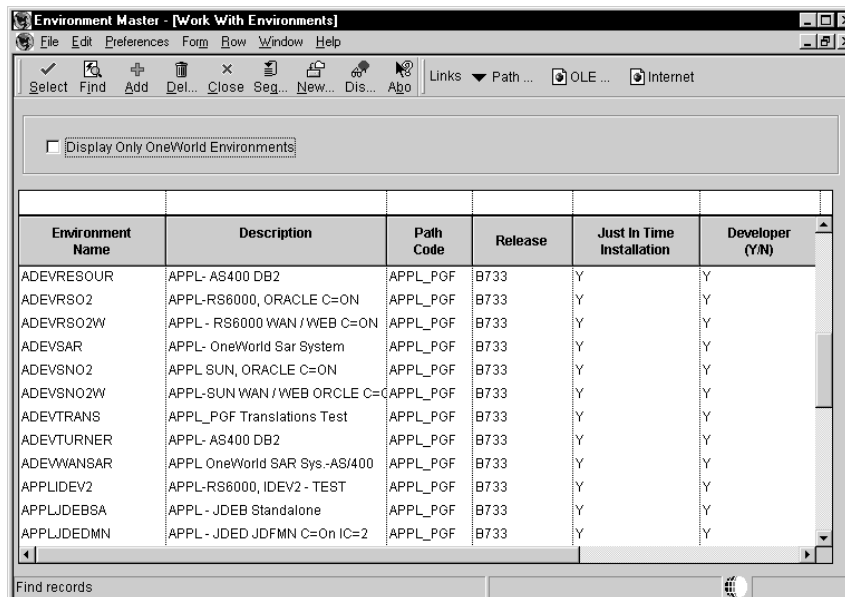
After you schedule your package for deployment, at the specified time on the date you specified, the package will be deployed to workstations. This package will be made available when the user signs onto OneWorld.

Testing a Partial Package Template

After you have installed the partial package template onto a workstation (laptop), you should test it to see if you are able to run the application. You should be able to run the application while disconnected from the network, without having objects installed onto your laptop via just-in-time installation.

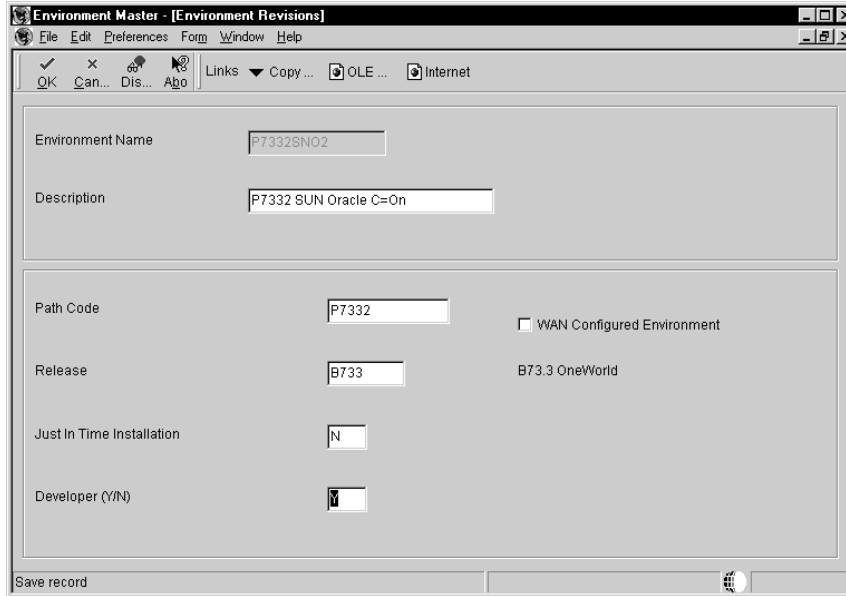
► To test a partial package template

1. Make sure the laptop is connected to the network.
2. Launch OneWorld and sign on.
3. From the Environments menu (GH9053), choose Environment Master (P0094). The Work With Environments form appears.



Environment Name	Description	Path Code	Release	Just In Time Installation	Developer (Y/N)
ADEVRESOUR	APPL- AS400 DB2	APPL_PGF	B733	Y	Y
ADEVRSO2	APPL-RS6000, ORACLE C=ON	APPL_PGF	B733	Y	Y
ADEVRSO2W	APPL - RS6000 WAN / WEB C=ON	APPL_PGF	B733	Y	Y
ADEVRSAR	APPL- OneWorld Sar System	APPL_PGF	B733	Y	Y
ADEVRSNO2	APPL SUN, ORACLE C=ON	APPL_PGF	B733	Y	Y
ADEVRSNO2W	APPL-SUN WAN / WEB ORACLE C=	APPL_PGF	B733	Y	Y
ADEVTRANS	APPL_PGF Translations Test	APPL_PGF	B733	Y	Y
ADEVTURNER	APPL- AS400 DB2	APPL_PGF	B733	Y	Y
ADEWANSAR	APPL OneWorld SAR Sys.-AS/400	APPL_PGF	B733	Y	Y
APPLIDEV2	APPL-RS6000, IDEV2 - TEST	APPL_PGF	B733	Y	Y
APPLJDEBSA	APPL - JDEB Standalone	APPL_PGF	B733	Y	Y
APPLJDEDMN	APPL - JDED JDFMN C=On IC=2	APPL_PGF	B733	Y	Y

4. Find and select the environment you used to create the partial package template. The Environment Revisions form appears.



5. Verify that the Just In Time Installation field is set to N. Setting this field to N switches off just-in-time installation. Just-in-time installation should be switched off when the laptop is not connected to the server.
6. Disconnect the laptop from the network and run the application.

You should be able to run the application while disconnected from the server. If you did not include in the package an object required by the application, you will receive an error message informing you that just-in-time installation has been disabled for the environment you are using.

If you receive this message, you will need to determine the missing objects and build another package before you can use the laptop while disconnected from the server.

Package Build Time Comparison

For comparison purposes, the following table shows package build and compression times for a typical full package, partial package, partial packages with limited numbers of objects, and the Sales Force Automation partial package used as an example in this document.

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

OneWorld 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 requiring 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, configurations that have either multiple remote sites or large numbers of users at a single site are constrained during the actual installation by network capacity. For example, software installations to workstations 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 onto workstations and servers from more than one deployment location and more than one deployment server. These additional deployment locations and servers are referred to as *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 simply 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 aren't required 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 consider implementing multitier deployment if your site has more than 50 workstations performing OneWorld software installations per day.

Topics

- ❑ Understanding multitier deployment
- ❑ Defining deployment servers
- ❑ Scheduling packages for deployment locations

- ❑ Distributing software to deployment locations
- ❑ Installing workstation packages from deployment locations
- ❑ Multitier deployment for server packages
- ❑ Understanding a multitier deployment case study

Understanding Multitier Deployment

The primary function of multitier deployment is to reduce network traffic (and the resultant delays that come with 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. This server at the remote location can then deploy the package to the workstations and servers at that location.

There are two main reasons to consider implementing a multitier deployment configuration:

- If too many workstations are installing packages from the same location, the server and network performance will suffer.
- If the workstations are remotely connected to the deployment server over a WAN, the installation time may be unacceptable.

Normally, you will decide to implement multitier deployment during CNC implementation. Although you can enable this function at any time, typically you will set it up after you have already installed OneWorld 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 this is done by modifying the user profiles to indicate the deployment location from which a user will pull 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 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 deploying in a single-tiered setup.

Multitier Deployment Terminology

The following describes multitier deployment terms:

Primary Deployment Location (tier 1)

The primary or base deployment location is where the package to be deployed to secondary or remote locations is built. OneWorld 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 OneWorld and should not be used for any other purposes in your company. See the OneWorld Installation Guide for this machine's hardware and software requirements.

Tier Deployment Location (tier 2)

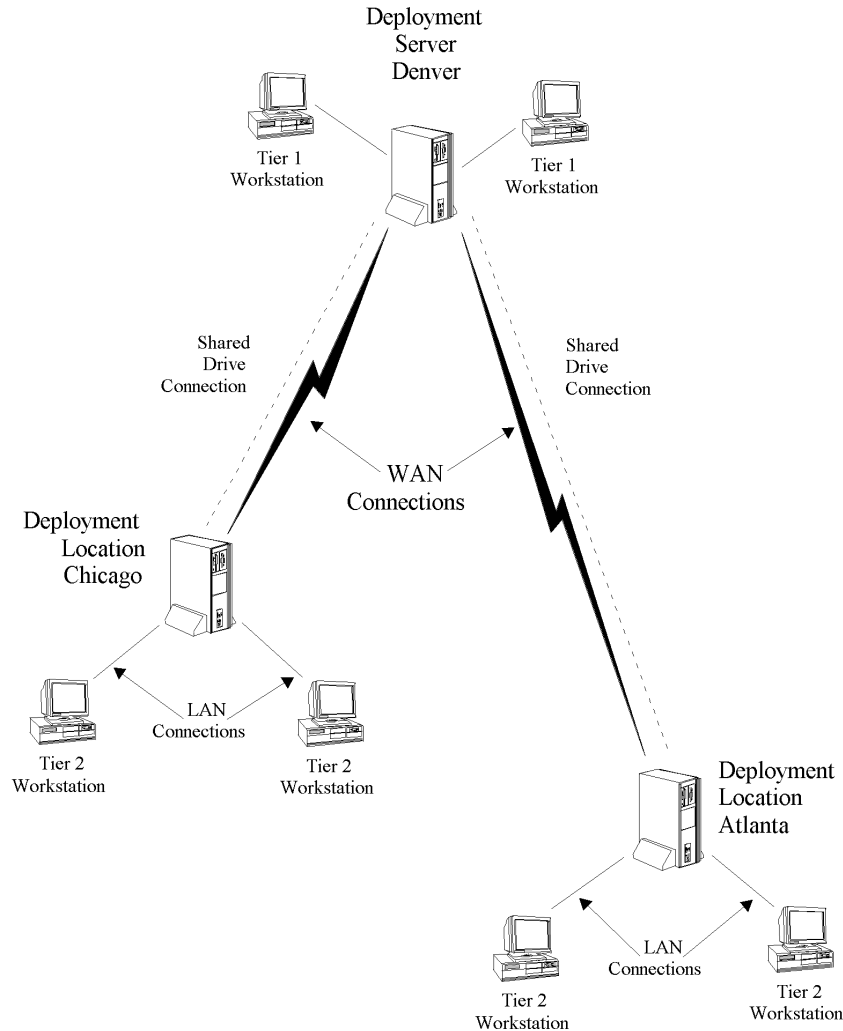
Tier deployment locations (also known as remote or secondary locations) have one or more deployment servers that allow you to install OneWorld software onto the workstations at that location. These servers receive their packages from the deployment server at the primary or base location. Machines at the tier deployment locations will not be able to 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 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 location. See the OneWorld Installation Guide for the hardware and software requirements of these machines.

Example: Two-Tier Deployment Strategy

The following illustrates a typical two-tier deployment strategy:



Multitier Implementation

Packages are always built on the deployment server at the primary location. Once you have built the package that you want to deploy to remote locations, you must follow these steps to implement multitier deployment.

7. 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 would need to define the Seattle deployment location. For information, see [Defining Locations](#).

8. Create Deployment Server Definitions

You must also define the deployment server at each remote location. For more information, see [Defining Deployment Servers](#).

Note: This step is not necessary if you used the Remote Location Workbench to create deployment server definitions when you installed OneWorld. For more information about the Remote Location Workbench, see the *OneWorld Installation Guide*.

9. Schedule the Package

The next step in the process is to schedule the package for deployment using the Package Deployment application (P9631). The process of scheduling a package for multitier deployment is identical to scheduling any other package. For more information, see [Distributing Software to Deployment Locations](#).

10. Deploy the Package to Workstations

Once you have deployed the package to the deployment server at the remote location, that server can deploy to the workstations at that location.

Before You Begin

Before you begin implementing multitier deployment you should do the following:

- ❑ Ensure that you have a thorough understanding of CNC concepts that will allow you to define and implement packages, workstation installations, path codes, central objects, replicated objects, and user profiles.
- ❑ Ensure that there is adequate disk space on the disk drive for the machine you will be using as a tier deployment location. For more information about minimum disk space requirements for a deployment server, see *OneWorld Disk Space Requirements* in the *OneWorld Installation Guide*. Also keep in mind that each full package you deploy will add approximately 1.4GB, and each additional partial package will add approximately 165MB. The amount of required disk space will vary, depending on the amount of data you replicate to a Microsoft Access database.

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, OneWorld creates a record in the Deployment Locations Definition table (F9654) for each deployment location. Each server at each deployment location must be defined.

Typically, there will be one record for the primary deployment server and one record per deployment location for each release. If you are running multiple releases of OneWorld, 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 OneWorld, you do not need to define deployment servers again. For more information about the Remote Location Workbench, see the *OneWorld Installation Guide*.

Defining a New Deployment Server

The following task describes how to define a new deployment server. You would 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

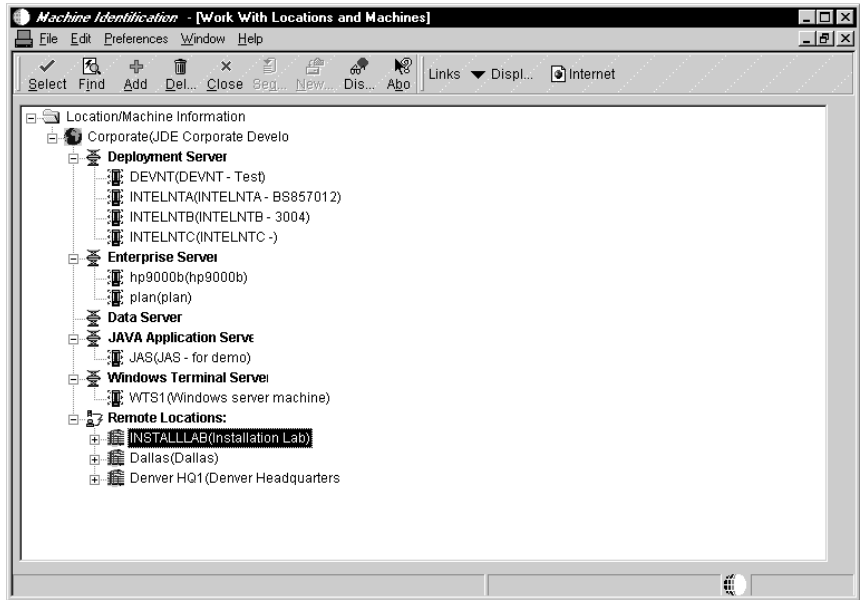
- ❑ If you are adding a new definition for a deployment server at a remote location, you first need to define that location. For information about defining locations, see [Defining Locations](#).

Note

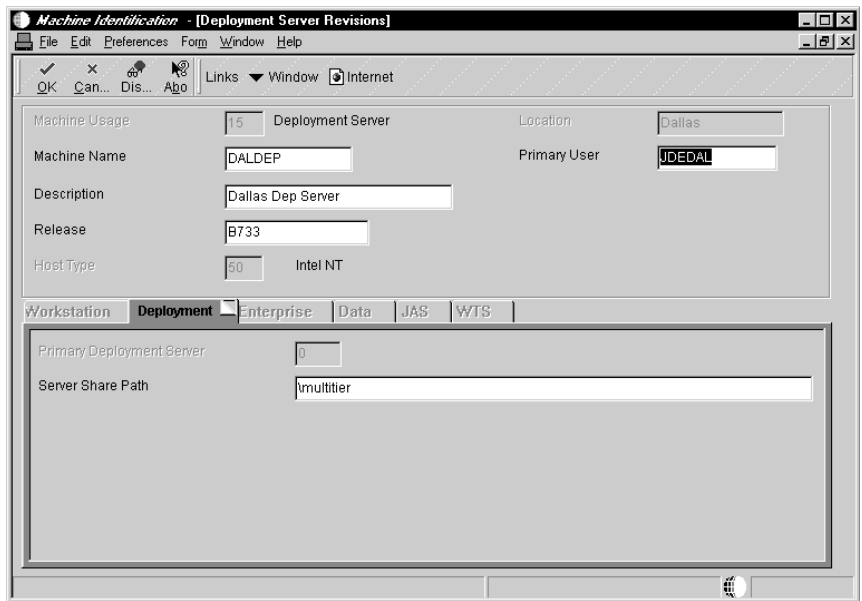
J.D. Edwards recommends that you do not define deployment locations with a DEV path code. This is because the DEV path code is normally associated with development staff that is not located at remote locations.

► **To add a new deployment server definition**

From the Package and Deployment Tools menu (GH9083), choose Machine Identification (P9654A). The Work With Locations and Machines form appears.



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.

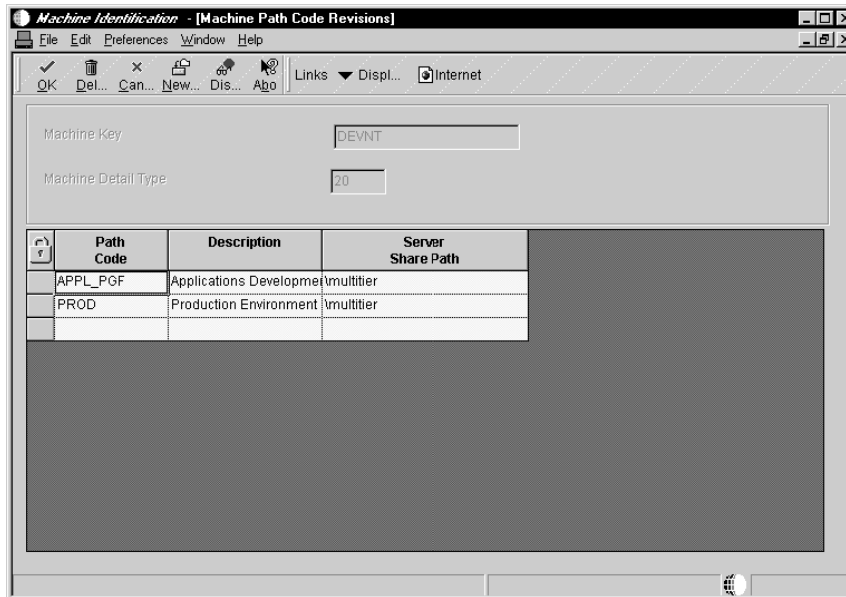


3. On the Deployment Server Revisions form, complete the following fields:

- 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.

Field Descriptions

Field	Description
Machine Name	For World, the Location indicates the machine (server or client). For OneWorld, the Location or Machine Key indicates the name of the machine on the network (server or workstation).
Description	A user defined name or remark.
Release	For World, the release number as defined in the Software Versions Repository. For OneWorld, the release number as defined in the Release Master.
Primary User	Primary User for the listed Machine.
Server Share Path	For World, the Server Share Path Field is used by the environment to determine the location of the current server. For OneWorld, this field indicates the shared directory for this Path Code. The objects that are stored on a file server will be found in this path.

Revising an Existing Deployment Server Definition

There may be situations in which you need to modify the definition for a deployment server you already defined. For example, you would need to change the definition if the server share path or OneWorld release changes, or if you want to designate a different server as your primary deployment server.

The procedure for revising an existing deployment server definition is very similar to adding a new definition.

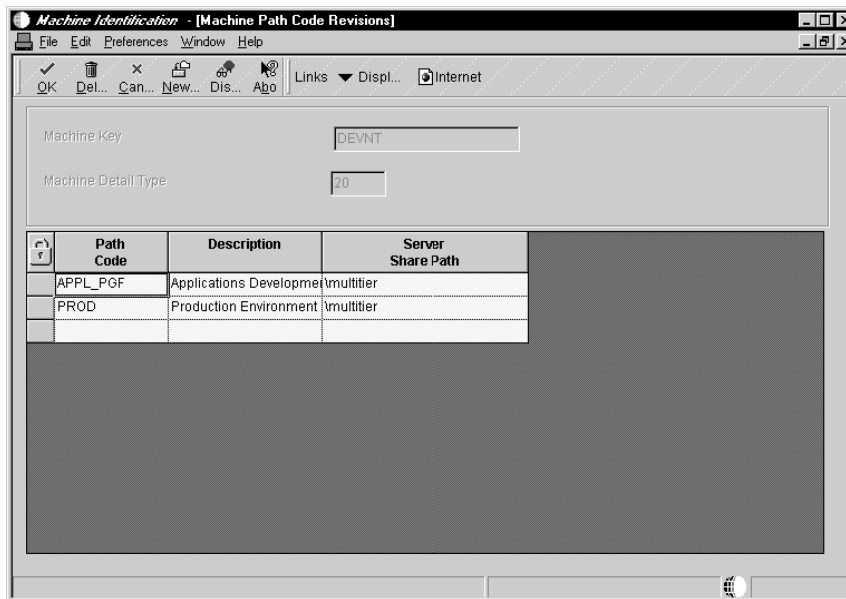
► **To revise an existing deployment server definition**

From the Package and Deployment Tools menu (GH9083), choose Machine Identification (P9654A). The Work with Locations and Machines form appears.

1. Find the deployment server whose definition you want to modify, and then click Select. The Deployment Server Revisions form appears.
2. On the Deployment Server Revisions form, 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 will be unable to change this field. A 1 at 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 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.

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.

For more information about defining machines, locations, and deployment groups, see [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 will access the tier deployment server for the help files unless you choose to deploy the help files to the workstation.

After you have created or modified a deployment location, you can schedule packages for deployment by using the Package Deployment application (P9631). The process for scheduling packages to a secondary deployment server is identical to scheduling a package to a primary server. Be sure to specify the secondary server location when you schedule the package.

Note:

When you are creating a distribution schedule, keep in mind 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.

See Also

- [Deploying Packages](#).

Distributing Software to Deployment Locations

The process of distributing software to deployment locations is handled through the Package Deployment application (P9631) or the Multitier Deployment batch application (R98825C). Use the Package Deployment application 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 scheduling application's deployment function or batch application. You will be pushing the software if you run the application or report on the primary deployment server or from a workstation. Conversely, you will be pulling the software if you run the application from a workstation at the tier deployment location. In either case, make sure to execute the application on the primary deployment server machine for push, or the destination deployment location for pull.

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 has been automatically distributed through the Package Deployment application or Multitier Deployment batch application, you will have to 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.

Topics

- ❑ Distributing software through Package Deployment
- ❑ Distributing software through the Multitier Deployment batch application
- ❑ Copying workstation installation programs to deployment locations

See Also

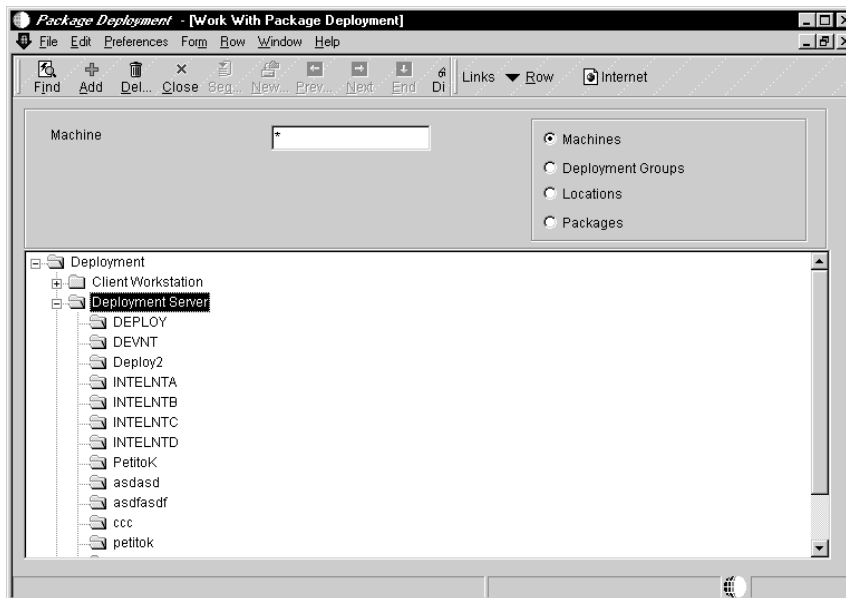
- Scheduling Packages for Deployment Locations

Distributing Software through Package Deployment

Use this method if 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 you specify in the scheduling fields.

► To distribute software through Package Deployment

From the Package and Deployment Tools menu (GH9083), choose Package Deployment (P9631). The Work with Package Deployment form appears.



1. Click the Machines option at the upper right of the form, and then click Find. Choose the deployment server to which you want to deploy.

If you have scheduled 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.

2. Choose the deployment server name to deploy all packages listed under the server name. Alternatively, choose only the package you want to deploy.
3. From the Row menu, choose Deploy. This launches the R98825C batch application that enables you to deploy to deployment servers.

This same Deploy option is also used to launch a batch application that deploys enterprise server packages, so if you choose an enterprise server name or the Enterprise Server folder on the Work With Package

Deployment form, the Enterprise Server Deployment batch application will be launched, not the Multitier Deployment batch application. When you choose a workstation, the Deploy option is not available.

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 application (P9631). Running the batch application 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 will have no effect. In this example, in order to distribute the software you must run the batch process report 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 (P98305). The Work with Batch Versions form appears.

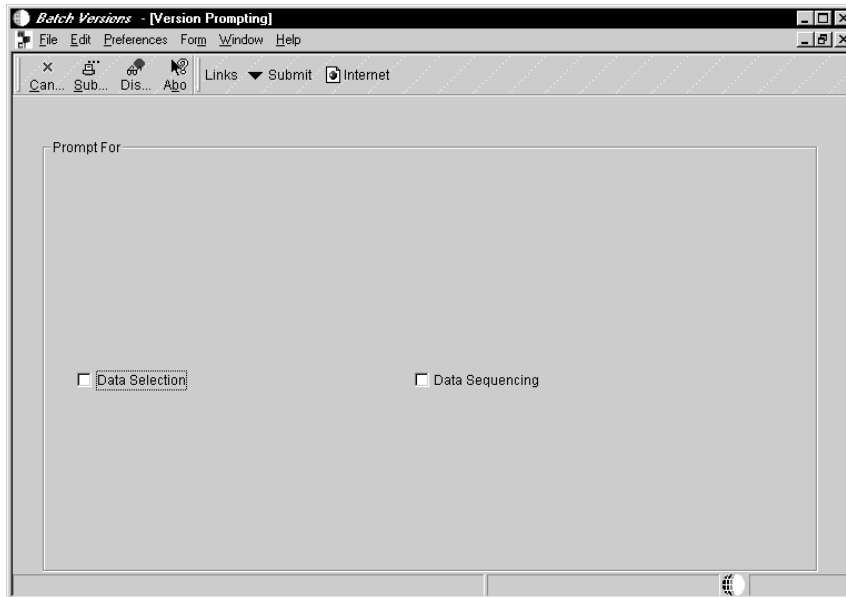
Version	Version Title	User	Last Modified	Security
XJDE0001	Multi Tier Deployment	JL5562379	8/7/98:0	No Security

1. Complete the following field:

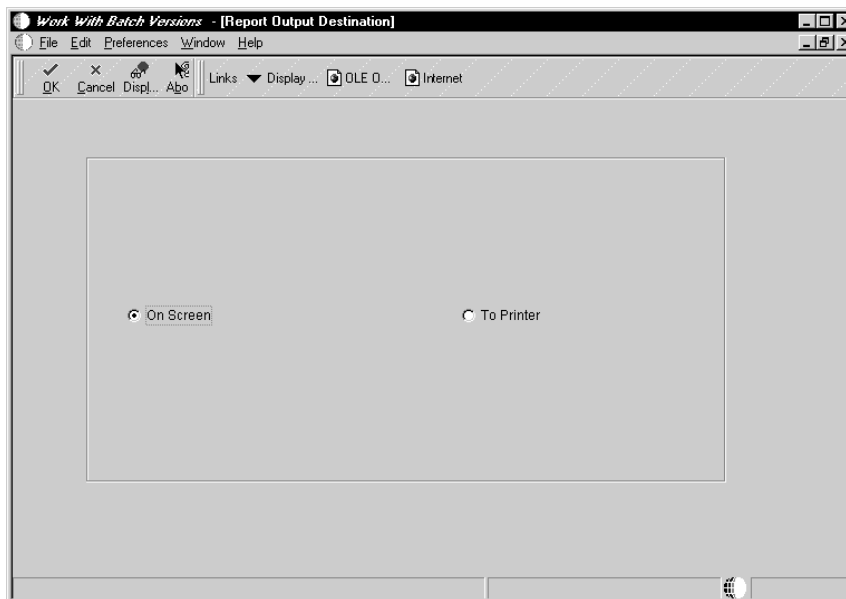
- Batch Application

Enter R98825C at the Batch Application field.

2. Find and choose the Multitier Deployment version 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 application. This output should indicate that the report was successfully completed.

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.

Doing these steps ensures that OneWorld runs the Client Workstation Installation program locally at the deployment location. If you do not do these two steps, OneWorld looks for the Client Workstation Installation application and associated files at the base location, and will copy the files across the WAN to the deployment location.

See Also

- *Understanding Package.inf Files* for information about locating and modifying the package.inf file.

► **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\B7333\OneWorld Client Install
```

2. Open the package.inf file and locate the [FileLocations] section. Change the PackageInfs line to reflect the deployment server machine name and 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. Exit from the package.inf file by choosing 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 that of a normal package installation. Basically, it consists of running the setup.exe program.

In the case of multitier deployment, this program is resident on the tier deployment location in the \client subdirectory that is under the base installation directory. The workstation attached to the deployment location must have read access to this directory on the deployment location.

For more information about the Workstation Installation program, see the *OneWorld Installation Guide*.

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.

Packages are typically built on the primary deployment server at the base location, but there might be advantages to using a different server for installations from the base location. There might also be situations where it would be preferable to deploy a server package to a server at a remote location rather than requiring remote users to access the server package over a WAN.

Server package multitier deployment gives users the ability 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.

Server multitier deployment works through the Package Deployment application (P98631). When you deploy a server package, the following components are copied:

- Foundation, Data, and Helps
- Subdirectories under the package name directory:
 - bin32
 - include
 - lib32
 - obj
 - source
 - spec
- Subdirectories under the server build directories:
 - bin32
 - spec
 - obj

- lib32
- The ServerPackage.inf file under the server build directories
- The package.inf file
- The pack directory under the package name directory

When server builds only are deployed without client builds, none of the subdirectories under the package name directory are copied except for the pack subdirectory.

The following 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 <i><server share path>\<path code>\<package name></i> on the destination deployment server.
Foundation, data, and helps	Copied from the path indicated in the SrcDirs section of the package.inf file.	If the default location is selected, foundation, data and helps are copied to <i><server share path>\systemcomp</i> on the destination deployment server.
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 Release Master table (F00942). Otherwise, this file is copied from the path to the package.inf file in the Machine Detail table (F9651).	Copied to <i><server share path>\package_inf</i> on the destination deployment server.

Smart Deployment

Server multitier deployment incorporates the "smart deployment" feature, which tells OneWorld to make available only the server packages that match the package destination's available servers. For example, if there are server packages 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 matching the destination's platform 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.

Deploying a Server Package with Multitier Deployment

After you have created the server package you want to deploy using multitier deployment, you can use the Package Deployment application to schedule that package to a server at the same location or a remote location. Make certain that the server package is compressed, because unlike client packages, you will not be allowed to deploy a non-compressed server package using multitier deployment.

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.

Before You Begin

- ❑ Assemble and build the server package as described in the *Package Build* section.

The following task summarizes the steps involved in scheduling a server package for multitier deployment. This process is essentially the same as for scheduling packages within the same location. For more detailed information, see the *Deployment* section.

Unless specified otherwise, after the Package Deployment Director launches, click Next to proceed to the next form.

► To schedule a server package for multitier deployment

From the Package and Deployment Tools menu (GH9083), choose Package Deployment (P9631). The Work With Package Deployment form appears.

1. Click Add to launch the Package Deployment Director.
2. Select the server package you want to deploy.
3. On the Package Deployment Targets form, choose the Deployment Server option.
4. On the Package Deployment Attributes form, enter the deployment date and time, and choose any deployment options.
5. On the Deployment Server Selection form, enter the machine name to which you want to deploy the server package. Be sure to select the machine by double-clicking in the column to the left of the deployment server name.
6. On the Build Selection form, select the build (or version) of the package you want to deploy by double-clicking in the column next to the server name.

Because of the Smart Deployment feature, only package builds that match the configuration at the destination location will be available for selection. For example, if there are package builds for an AS/400 and an HP9000 but the destination location has only an HP9000, the AS/400 build will not be available for selection.

7. Click Close to exit from the Build Selection form.
8. On the Work With Package Deployment form, click End to finish the server package scheduling process.

After you have scheduled the package, you can deploy it by using one of the two methods described in *Distributing Software to Deployment Locations*: either through the Package Deployment application, or through the Multitier Deployment batch process.

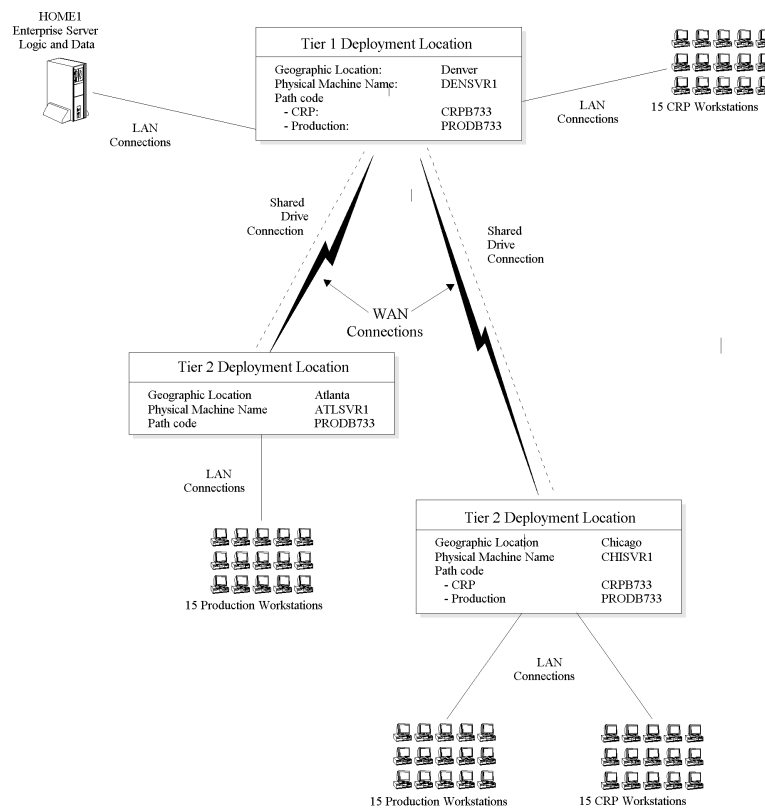
Understanding a Multitier Deployment Case Study

This chapter presents a case study for a two-tier deployment environment. As this case study shows, it is generally not efficient to deploy OneWorld packages to workstations using WAN connections. 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 OneWorld 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 OneWorld 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	
	Denver	DENSVR1
Deployment server name	Atlanta	ATLSVR1
	Chicago	CHISVR1
	Denver	15
Prototype workstations	Atlanta	0
	Chicago	15
	Denver	0
Production workstations	Atlanta	15
	Chicago	15
OneWorld release	B7333	
	Denver	1
Deployment tier	Atlanta	2
	Chicago	2
	Denver	PD7333 PY7333
Path codes	Atlanta	PD7333
	Chicago	PD7333
		PY7333

► **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.** On the deployment server (DENSVR1 in this example)

Use the Machine Identification application (P9654A) to define all deployment locations. See [Defining Locations](#).

For this case study, you would define three locations: one for each deployment location (Denver, Atlanta, and Chicago). Values for the fields on the Location Revisions form are summarized in the following table.

Field	Value
Location	<p>Enter the name of the deployment location.</p> <p>In this case study, you would assign names for each physical deployment location:</p> <p>Denver</p> <p>Atlanta</p> <p>Chicago</p>
Description	<p>Enter a description (any value up to 30 characters) for each deployment location.</p> <p>For example:</p> <p>Denver: Denver - Tier 1</p> <p>Atlanta: Atlanta - Tier 2</p> <p>Chicago: Chicago - Tier 2</p>
Location Code	<p>Enter the current location for OneWorld deployment.</p> <p>For example, DEN.</p>
Parent Location	<p>Enter the name of the parent location for the location you are adding.</p> <p>For example, Corporate.</p>

2. Create Deployment Server Definitions.

Use the Machine Identification application (P9654A) to create a definition for each deployment server at the deployment locations you created. For this case study, you would need to define a deployment server for Atlanta and Chicago. The deployment server in Denver would already be defined because it is the primary (tier 1) server. See [Defining Deployment Servers](#).

For this case study, you would 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, you would 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 OneWorld release. For example, B7333.
Primary User	Enter the primary user for the machine you entered.
Server Share Path	Enter the name of the shared directory for the path code where system files and other files reside. For example, \B7333.

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 application (P9631) or the Multitier Deployment batch application (R98825C). See [Distributing Software to Deployment Locations](#).

Index

Activating a package, 111

Add Object

- batch applications moved, 50
- business function data structures moved, 48
- business function event rules moved, 48
- business functions moved, 47
- business views moved, 47
- embedded event rules moved, 48
- interactive applications moved, 49
- media objects moved, 49
- tables moved, 46

Administrators

- CNC, 4
- hardware, network, and third-party software, 5

Application consultant, 4

Application developers, 4

Application project leaders, 4

Application text modification rules, 38

Assembling a package

- adding features, 109
- adding objects, 105

Assembling Packages, 84

Batch application

- scheduling the push installation batch application, 245

Batch applications

- moved with Check In/Out, Add Object, and Get Object, 50
- moved with Workstation Installation, 49, 50
- moving, 50

Build

- changing status, 180
- history, 138
- options, 134
 - revising, 134
- process files, 51

Build Selection Form, 222

Building Business Functions, 118

BusBuild, 118

Business function

- build
 - AS/400, 56
 - UNIX server, 52
 - Windows NT server, 54

- data structures
 - moved with Check In/Out, Add Object, and Get Object, 48
 - moved with Workstation Installation, 48
- event rules
 - moved with Check In/Out, Add Object, and Get Object, 48
 - moving, 48
- modification rules, 43
- moving, 48
- Business Function Errors log, 177
- Business function event rules
 - moved with Workstation Installation, 49
- Business functions
 - moved with Check In/Out, Add Object, and Get Object, 47
 - moved with Workstation Installation, 47
- Business views
 - modification rules, 40
 - moved with Check In/Out, Add Object, and Get Object, 47
 - moved with Workstation Installation, 47
 - moving, 47
- C business functions
 - moving, 47
- Central objects, 58
- Central Objects
 - correlating, 58
- Changing Listener default parameter settings, 243
- Check In/Out
 - business function data structures moved, 48
 - business function event rules moved, 48
 - business functions moved, 47
 - business views moved, 47
 - embedded event rules moved, 48
 - interactive applications moved, 49, 50
 - media objects moved, 49
 - tables moved, 46
- CNC administrator, 4
- CNC consultant, 4
- Comparing deployment methods, 22
- Configurations
 - full, 67
 - partial, 67
- Consultant
 - application, 4
 - CNC, 4
 - custom solution, 4
 - hardware, network, and third-party software, 5
- Control tables modification rules, 39
- Copying
 - records, 62

- Copying a built package, 135
- Copying a package, 112
- Correlating replicated and central objects, 58
- CRP, 13
- Custom solution consultant, 4
- Data dictionary
 - moving items, 50
 - Storing, 50
 - tables, 50
- Data Dictionary Items
 - moving, 50
- Data structures modification rules, 42
- Database Component Selection form, 103
- Database Item Revisions form, 104
- Database Location, 103
- Default
 - changing Listener default parameter settings, 243
- Defining a Package Build, 119
- Defining a software distribution schedule, 279
- Defining deployment locations, 273
- Defining Machines, 201
- Deleting a package, 114
- Deploying
 - server packages, 231
- Deployment
 - data, 28
 - database for store and forward users, 30
 - groups, 209
 - locations, 200
 - defining, 273
 - defining a software distribution schedule, 279
 - distributing software, 280
 - installing workstation packages, 285
 - scheduling packages, 279
 - methods, 9
 - comparing, 22
 - understanding, 9
 - multitier, 22, 267
 - features, 269
 - implementation, 271
 - terminology, 270
 - understanding, 269
 - Multitiered Locations, 198
 - new replicated local database, 28
 - to servers, 198
 - to workstations from CD, 199
 - two tier
 - example, 271

- two-tier strategy, 271
- understanding, 66, 197
- workstations with OneWorld, 198
- workstations without OneWorld, 197
- Deployment Client Workstation Selection form, 221
- Deployment Groups Revisions form, 210
- Deployment Groups Selection form, 225
- Deployment Location Selection form, 225
- Deployment Properties Revisions form, 226
- Deployment server
 - definition, 270
- Deployment Server Revisions form, 274
- Deployment Server Selection form, 221
- DEV, 13, 17
- Developers
 - application, 4
- Developing short-term fixes, 20
- Development process
 - working with, 17
- Development processes
 - developing short-term fixes, 20
 - normal, 17
- Distributing software through the scheduling application, 281
- Distributing software to deployment locations, 280
- Embedded event rules
 - moved with Check In/Out, Add Object, and Get Object, 48
 - moved with Workstation Installation, 48
 - moving, 48
- Ensuring the integrity of the production environment, 15
- Enterprise Server Selection form, 223
- Event rule
 - business function, 48
 - embedded, 48
 - modification rules, 41
- Feature Component Selection form, 110
- Features
 - adding to a package, 109
- Forms
 - Build Selection, 222
 - Database Component Selection, 103
 - Database Item Revisions, 104
 - Deployment Client Workstation Selection, 221
 - Deployment Groups Revisions, 210
 - Deployment Groups Selection, 225
 - Deployment Location Selection, 225
 - Deployment Properties Revisions, 226
 - Deployment Server Revisions, 274
 - Deployment Server Selection, 221

Enterprise Server Selection, 223
 Feature Component Selection, 110
 Foundation Component Selection, 97
 Foundation Item Revisions, 98
 Help Component Selection, 101
 Help Item Revisions, 102
 Location Revisions, 208
 Machine Path Code Revisions, 275
 Machine Revisions, 202
 Object Component Selection, 106
 Package Assembly Director, 88, 260
 Package Build Director, 121
 Package Deployment Attributes, 220
 Package Deployment Director, 218, 263
 Package Deployment Groups, 210
 Package Deployment Targets, 219
 Package Information, 88, 260
 Package Selection, 122, 219, 263
 Report Output Destination, 283
 Scheduled Packages, 229
 Solution Explorer Record Copy, 62
 Version Prompting form, 283
 View Logs, 171
 Work With Batch Versions Deployment, 282
 Work with Locations and Machines, 274
 Work With Locations and Machines, 202
 Work With Package Build Definition, 120
 Work with Package Deployment, 218, 263
 Work With Package Deployment Selection, 223
 Work With Packages, 260
 Foundation Component Selection form, 97
 Foundation Item Revisions form, 98
 Foundation location, 97
 Full configurations, 67
Full package, 8, 22, 70
 Full workstation, 67
 Get Object

- batch applications moved, 50
- business function data structures moved, 48
- business function event rules moved, 48
- business functions moved, 47
- business views moved, 47
- embedded event rules moved, 48
- interactive applications moved, 49
- media objects moved, 49
- tables moved, 46

 Hardware administrators, 5
 Hardware consultants, 5

- Help Component Selection form, 101
- Help Item Revisions form, 102
- Help Location, 101
- INF files, 183
- Installing a scheduled package, 229
- Installing workstation packages from deployment locations, 285
- Interactive applications
 - modification rules, 34
 - Moved with Check In/Out, Add Object, and Get Object, 49
 - moving, 49
- JITI, 69
- Just-in-time installation
 - disabling, 15
 - sites, 21
 - understanding, 69
- Listener
 - changing default parameter settings, 243
 - definition, 235
 - installing, 237
 - installing via Silent Installation, 240
 - stopping, 242
 - uninstalling, 243
- Location
 - database, 103
 - deployment, 200
 - foundation, 97
 - help, 101
- Location Revisions form, 208
- Logs
 - Business Function Errors, 177
 - Missing Business Function Source Errors, 178
 - Package Build, 174
 - Package Statistics, 174
- Machine Path Code Revisions form, 275
- Machine Revisions form, 202
- Machines
 - defining, 201
- Managing objects with the SAR system, 45
- Media object data structures
 - moving, 49
- Media objects
 - moved with Check In/Out, Add Object, and Get Object, 49
 - Moved with Workstation Installation, 49
- Missing Business Function Source Errors log, 178
- Modification rules
 - application text, 38
 - business functions, 43
 - business views, 40

- control tables, 39
- data structures, 42
- event rules, 41
- interactive applications, 34
- OneWorld, 34
- report specifications, 36
- reports, 36
- table specifications, 38
- versions, 43
- Moving
 - batch applications, 50
 - business function event rules, 48
 - business functions, 48
 - business views, 47
 - C business functions, 47
 - Data Dictionary Items, 50
 - embedded event rules, 48
 - interactive applications, 49
 - media object data structures, 49
 - tables, 46
- Moving business views, 47
- Multitier deployment, 267
 - case study, 290
 - features, 270
 - implementation, 271
 - terminology, 270
 - understanding, 269
- Network administrators, 5
- Network consultants, 5
- Object Component Selection form, 106
- Object movement
 - understanding, 46
- Objects
 - adding to a package, 105
 - central, 58
 - managing, 45
 - preserved after an upgrade, 33
 - replaced after an upgrade, 33
 - replicated, 58
- OneWorld
 - modification rules, 34
 - understanding roles, 4
- Package
 - activating, 111
 - assembling, 84
 - build definition, 119
 - build history, 138, 170
 - build processing options, 134

- build resubmissions, 138, 179
- copying, 112
- cumulative, 21
- defining build, 115
- deleting, 114
- deploying
 - server, 231
- deployment, 9
- deployment groups, 209
- full package**, 8, 70
- INF files, 183
- installing a scheduled package, 229
- names, 8
- partial package**, 8, 21, 70
- revising, 111
- schedule, 8
- server packages, 76
- understanding packages, 8
- update, 21
- update package**, 8, 71
- Package Assembly Director, 260
 - default values, 86
 - reviewing selections, 95
- Package Assembly Director form, 88
- Package Build Director, 121
- Package Build History, 169
- Package Build log, 174
- Package Build Logs, 169
- Package Deployment Attributes form, 220
- Package Deployment Director form, 218, 263
- Package Deployment Groups form, 210
- Package Deployment Targets form, 219
- Package Information form, 88, 260
- Package management
 - overview, 2
 - planning, 12
 - setup, 12
- Package Management guide
 - understanding, 6
- Package Selection form, 122, 219, 263
- Package Statistics log, 174
- Package types**
 - Full package**, 8, 70
 - partial package, 70
 - Partial package**, 8
 - Update package**, 8, 71
- Packages
 - build process overview, 116

- copying, 135
- deployment overview, 72
- types, 70
- understanding, 66
- Partial package**, 8, 21, 70
- Partial workstation, 67
- Partial workstation configurations, 67
- Path code
 - CRP, 13
 - DEV, 13
 - development process, 17
 - PRIST, 13
 - PROD, 13, 17
 - recommended, 13
- Performing backups and restoring objects, 60
- PRIST, 13
- PROD, 13, 17
- Production environment
 - ensuring integrity, 15
- Push Installation
 - preparing the enterprise server, 236
 - preparing workstations, 236
 - changing default parameter settings, 243
 - installing the Listener, 237
 - installing the Listener via Silent Installation, 240
 - stopping the Listener, 242
 - uninstalling the Listener, 243
 - running the push package installation results report, 250
 - scheduling a package, 244
 - scheduling the Push Installation batch application, 245
 - status codes, 251
- Records
 - copying, 62
- Replicated local database, 28
- Replicated Objects
 - correlating, 58
- Report Output Destination form, 283
- Report specification modification rules, 36
- Resubmitting builds, 138, 179
- Revising a deployment group, 212
- Revising a package, 111
- Revising a package's build options, 134
- Scheduled Packages form, 229
- Scheduling application
 - distributing software, 281
- Server build
 - AS/400, 56
 - UNIX, 52

- Windows NT, 54
- Server packages, 76
 - deploying, 231
 - understanding, 10
- Short-term fixes
 - developing, 20
- Software
 - administrators, 5
 - consultants, 5
 - distributing
 - through the scheduling application, 281
 - distributing to deployment locations, 280
- Software distribution schedule
 - for deployment locations, 279
- Solution Explorer
 - copying control table records, 62
- Solution Explorer Record Copy form, 62
- Status codes
 - push installation, 251
- Storing the data dictionary, 50
- Sun platform, 82
- Table
 - Data Dictionary, 50
- Table specifications modification rules, 38
- Tables
 - moved with Check In/Out, Add Object, and Get Object, 46
 - moved with Workstation Installation, 47
 - moving, 46
- Third-party software administrators, 5
- Third-party software consultants, 5
- Tier deployment location, 270
- Tier workstations, 270
- Two tier deployment, 271
 - example, 271
- Understanding files created by the build process, 51
- Understanding object movement, 46
- Understanding OneWorld modification rules, 34
- Understanding server packages, 10
- Update package**, 8, 71
- Version Prompting form, 283
- Versions modification rules, 43
- View Logs form, 171
- Viewing logs, 169, 171
- Work With Batch Versions form, 282
- Work with Locations and Machines form, 274
- Work With Locations and Machines form, 202
- Work With Package Build Definition form, 120
- Work with Package Deployment form, 218, 263

Work With Package Deployment form, 223
Work With Packages form, 260
Working with the normal development process, 17
Workstation Build
 files created, 51
Workstation Installation
 batch applications moved with, 49, 50
 business function data structures moved with, 48
 business function event rules moved with, 49
 business functions moved with, 47
 business views moved with, 47
 embedded event rules moved with, 48
 media objects moved with, 49
 tables moved with, 47
 understanding, 9