ORACLE
JD EDWARDS ENTERPRISEONE

JD Edwards EnterpriseOne Tools 8.98 Package Management Guide

September 2008

ORACLE

JD Edwards EnterpriseOne Tools 8.98 Package Management Guide
SKU E1_TOOLS898TPK-B 0908

Trademark Notice

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

License Restrictions Warranty/Consequential Damages Disclaimer

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

Subject to patent protection under one or more of the following U.S. patents: 5,781,908; 5,828,376; 5,950,010; 5,960,204; 5,987,497; 5,995,972; 5,987,497; and 6,223,345. Other patents pending.

Warranty Disclaimer

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

Restricted Rights Notice

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS

Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

Hazardous Applications Notice

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Third Party Content, Products, and Services Disclaimer

This software and documentation may provide access to or information on content, products and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third party content, products and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third party content, products or services.

Open Source Disclosure

Oracle takes no responsibility for its use or distribution of any open source or shareware software or documentation and disclaims any and all liability or damages resulting from use of said software or documentation. The following open source software may be used in Oracle's JD Edwards EnterpriseOne products and the following disclaimers are provided:

Contents

About This Documentation Preface

JD Edwards EnterpriseOne implementation guides provide you with the information that you need to implement and use JD Edwards EnterpriseOne applications from Oracle.

This preface discusses:

- JD Edwards EnterpriseOne application prerequisites.
- Application fundamentals.
- Documentation updates and downloading documentation.
- Additional resources.
- Typographical conventions and visual cues.
- Comments and suggestions.
- Common fields in implementation guides.

Note. Implementation guides document only elements, such as fields and check boxes, that require additional explanation. If an element is not documented with the process or task in which it is used, then either it requires no additional explanation or it is documented with common fields for the section, chapter, implementation guide, or product line. Fields that are common to all JD Edwards EnterpriseOne applications are defined in this preface.

JD Edwards EnterpriseOne Application Prerequisites

To benefit fully from the information that is covered in these books, you should have a basic understanding of how to use JD Edwards EnterpriseOne applications.

You might also want to complete at least one introductory training course, if applicable.

You should be familiar with navigating the system and adding, updating, and deleting information by using JD Edwards EnterpriseOne menus, forms, or windows. You should also be comfortable using the World Wide Web and the Microsoft Windows or Windows NT graphical user interface.

These books do not review navigation and other basics. They present the information that you need to use the system and implement your JD Edwards EnterpriseOne applications most effectively.

Application Fundamentals

Each application implementation guide provides implementation and processing information for your JD Edwards EnterpriseOne applications.

For some applications, additional, essential information describing the setup and design of your system appears in a companion volume of documentation called the application fundamentals implementation guide. Most product lines have a version of the application fundamentals implementation guide. The preface of each implementation guide identifies the application fundamentals implementation guides that are associated with that implementation guide.

The application fundamentals implementation guide consists of important topics that apply to many or all JD Edwards EnterpriseOne applications. Whether you are implementing a single application, some combination of applications within the product line, or the entire product line, you should be familiar with the contents of the appropriate application fundamentals implementation guides. They provide the starting points for fundamental implementation tasks.

Documentation Updates and Downloading Documentation

This section discusses how to:

- Obtain documentation updates.
- Download documentation.

Obtaining Documentation Updates

You can find updates and additional documentation for this release, as well as previous releases, on Oracle's PeopleSoft Customer Connection website. Through the Documentation section of Oracle's PeopleSoft Customer Connection, you can download files to add to your Implementation Guides Library. You'll find a variety of useful and timely materials, including updates to the full line of JD Edwards EnterpriseOne documentation that is delivered on your implementation guides CD-ROM.

Important! Before you upgrade, you must check Oracle's PeopleSoft Customer Connection for updates to the upgrade instructions. Oracle continually posts updates as the upgrade process is refined.

See Also

Oracle's PeopleSoft Customer Connection, http://www.oracle.com/support/support_peoplesoft.html

Downloading Documentation

In addition to the complete line of documentation that is delivered on your implementation guide CD-ROM, Oracle makes JD Edwards EnterpriseOne documentation available to you via Oracle's website. You can download PDF versions of JD Edwards EnterpriseOne documentation online via the Oracle Technology Network. Oracle makes these PDF files available online for each major release shortly after the software is shipped.

See Oracle Technology Network, http://www.oracle.com/technology/documentation/psftent.html

Additional Resources

The following resources are located on Oracle's PeopleSoft Customer Connection website:

Resource	Navigation
Application maintenance information	Updates + Fixes
Business process diagrams	Support, Documentation, Business Process Maps

Resource	Navigation
Interactive Services Repository	Support, Documentation, Interactive Services Repository
Hardware and software requirements	Implement, Optimize + Upgrade; Implementation Guide; Implementation Documentation and Software; Hardware and Software Requirements
Installation guides	Implement, Optimize + Upgrade; Implementation Guide; Implementation Documentation and Software; Installation Guides and Notes
Integration information	Implement, Optimize + Upgrade; Implementation Guide; Implementation Documentation and Software; Pre-Built Integrations for PeopleSoft Enterprise and JD Edwards EnterpriseOne Applications
Minimum technical requirements (MTRs)	Implement, Optimize + Upgrade; Implementation Guide; Supported Platforms
Documentation updates	Support, Documentation, Documentation Updates
Implementation guides support policy	Support, Support Policy
Prerelease notes	Support, Documentation, Documentation Updates, Category, Release Notes
Product release roadmap	Support, Roadmaps + Schedules
Release notes	Support, Documentation, Documentation Updates, Category, Release Notes
Release value proposition	Support, Documentation, Documentation Updates, Category, Release Value Proposition
Statement of direction	Support, Documentation, Documentation Updates, Category, Statement of Direction
Troubleshooting information	Support, Troubleshooting
Upgrade documentation	Support, Documentation, Upgrade Documentation and Scripts

Typographical Conventions and Visual Cues

This section discusses:

- Typographical conventions.
- Visual cues.
- Country, region, and industry identifiers.
- Currency codes.

Typographical Conventions

This table contains the typographical conventions that are used in implementation guides:

Typographical Convention or Visual Cue	Description
Bold	Indicates PeopleCode function names, business function names, event names, system function names, method names, language constructs, and PeopleCode reserved words that must be included literally in the function call.
Italics	Indicates field values, emphasis, and JD Edwards EnterpriseOne or other book-length publication titles. In PeopleCode syntax, italic items are placeholders for arguments that your program must supply. We also use italics when we refer to words as words or letters as letters, as in the following: Enter the letter *O*.
KEY+KEY	Indicates a key combination action. For example, a plus sign (+) between keys means that you must hold down the first key while you press the second key. For ALT+W, hold down the ALT key while you press the W key.
`Monospace font`	Indicates a PeopleCode program or other code example.
" " (quotation marks)	Indicate chapter titles in cross-references and words that are used differently from their intended meanings.
. . . (ellipses)	Indicate that the preceding item or series can be repeated any number of times in PeopleCode syntax.
{ } (curly braces)	Indicate a choice between two options in PeopleCode syntax. Options are separated by a pipe (|).
[ ] (square brackets)	Indicate optional items in PeopleCode syntax.
& (ampersand)	When placed before a parameter in PeopleCode syntax, an ampersand indicates that the parameter is an already instantiated object. Ampersands also precede all PeopleCode variables.

Visual Cues

Implementation guides contain the following visual cues.

Notes

Notes indicate information that you should pay particular attention to as you work with the JD Edwards EnterpriseOne system.

Note. Example of a note.

If the note is preceded by *Important!,* the note is crucial and includes information that concerns what you must do for the system to function properly.

Important! Example of an important note.

Warnings

Warnings indicate crucial configuration considerations. Pay close attention to warning messages.

Warning! Example of a warning.

Cross-References

Implementation guides provide cross-references either under the heading "See Also" or on a separate line preceded by the word *See.* Cross-references lead to other documentation that is pertinent to the immediately preceding documentation.

Country, Region, and Industry Identifiers

Information that applies only to a specific country, region, or industry is preceded by a standard identifier in parentheses. This identifier typically appears at the beginning of a section heading, but it may also appear at the beginning of a note or other text.

Example of a country-specific heading: "(FRA) Hiring an Employee"

Example of a region-specific heading: "(Latin America) Setting Up Depreciation"

Country Identifiers

Countries are identified with the International Organization for Standardization (ISO) country code.

Region Identifiers

Regions are identified by the region name. The following region identifiers may appear in implementation guides:

- Asia Pacific
- Europe
- Latin America
- North America

Industry Identifiers

Industries are identified by the industry name or by an abbreviation for that industry. The following industry identifiers may appear in implementation guides:

- USF (U.S. Federal)

- E&G (Education and Government)

Currency Codes

Monetary amounts are identified by the ISO currency code.

Comments and Suggestions

Your comments are important to us. We encourage you to tell us what you like, or what you would like to see changed about implementation guides and other Oracle reference and training materials. Please send your suggestions to your product line documentation manager at Oracle Corporation, 500 Oracle Parkway, Redwood Shores, CA 94065, U.S.A. Or email us at appsdoc@us.oracle.com.

While we cannot guarantee to answer every email message, we will pay careful attention to your comments and suggestions.

Common Fields Used in Implementation Guides

Address Book Number Enter a unique number that identifies the master record for the entity. An address book number can be the identifier for a customer, supplier, company, employee, applicant, participant, tenant, location, and so on. Depending on the application, the field on the form might refer to the address book number as the customer number, supplier number, or company number, employee or applicant ID, participant number, and so on.

As If Currency Code Enter the three-character code to specify the currency that you want to use to view transaction amounts. This code enables you to view the transaction amounts as if they were entered in the specified currency rather than the foreign or domestic currency that was used when the transaction was originally entered.

Batch Number Displays a number that identifies a group of transactions to be processed by the system. On entry forms, you can assign the batch number or the system can assign it through the Next Numbers program (P0002).

Batch Date Enter the date in which a batch is created. If you leave this field blank, the system supplies the system date as the batch date.

Batch Status Displays a code from user-defined code (UDC) table 98/IC that indicates the posting status of a batch. Values are:

Blank: Batch is unposted and pending approval.

A: The batch is approved for posting, has no errors and is in balance, but has not yet been posted.

D: The batch posted successfully.

E: The batch is in error. You must correct the batch before it can post.

P: The system is in the process of posting the batch. The batch is unavailable until the posting process is complete. If errors occur during the post, the batch status changes to *E*.

U: The batch is temporarily unavailable because someone is working with it, or the batch appears to be in use because a power failure occurred while the batch was open.

Branch/Plant Enter a code that identifies a separate entity as a warehouse location, job, project, work center, branch, or plant in which distribution and manufacturing activities occur. In some systems, this is called a business unit.

Business Unit Enter the alphanumeric code that identifies a separate entity within a business for which you want to track costs. In some systems, this is called a branch/plant.

Category Code Enter the code that represents a specific category code. Category codes are user-defined codes that you customize to handle the tracking and reporting requirements of your organization.

Company Enter a code that identifies a specific organization, fund, or other reporting entity. The company code must already exist in the F0010 table and must identify a reporting entity that has a complete balance sheet.

Currency Code Enter the three-character code that represents the currency of the transaction. JD Edwards EnterpriseOne provides currency codes that are recognized by the International Organization for Standardization (ISO). The system stores currency codes in the F0013 table.

Document Company Enter the company number associated with the document. This number, used in conjunction with the document number, document type, and general ledger date, uniquely identifies an original document.

If you assign next numbers by company and fiscal year, the system uses the document company to retrieve the correct next number for that company.

If two or more original documents have the same document number and document type, you can use the document company to display the document that you want.

Document Number Displays a number that identifies the original document, which can be a voucher, invoice, journal entry, or time sheet, and so on. On entry forms, you can assign the original document number or the system can assign it through the Next Numbers program.

Document Type Enter the two-character UDC, from UDC table 00/DT, that identifies the origin and purpose of the transaction, such as a voucher, invoice, journal entry, or time sheet. JD Edwards EnterpriseOne reserves these prefixes for the document types indicated:

P: Accounts payable documents.

R: Accounts receivable documents.

T: Time and pay documents.

I: Inventory documents.

O: Purchase order documents.

S: Sales order documents.

Effective Date

Enter the date on which an address, item, transaction, or record becomes active. The meaning of this field differs, depending on the program. For example, the effective date can represent any of these dates:

- The date on which a change of address becomes effective.
- The date on which a lease becomes effective.
- The date on which a price becomes effective.
- The date on which the currency exchange rate becomes effective.
- The date on which a tax rate becomes effective.

Fiscal Period and **Fiscal Year**

Enter a number that identifies the general ledger period and year. For many programs, you can leave these fields blank to use the current fiscal period and year defined in the Company Names & Number program (P0010).

G/L Date (general ledger date)

Enter the date that identifies the financial period to which a transaction will be posted. The system compares the date that you enter on the transaction to the fiscal date pattern assigned to the company to retrieve the appropriate fiscal period number and year, as well as to perform date validations.

JD Edwards EnterpriseOne Package Management Preface

This preface discusses JD Edwards EnterpriseOne Package Management companion documentation.

JD Edwards EnterpriseOne Package Management Companion Documentation

Additional, essential information describing the setup and design of Oracle's JD Edwards EnterpriseOne system resides in companion documentation. The companion documentation consists of important topics that apply to many or all of Oracle's JD Edwards EnterpriseOne Tools.

This guide contains references to server configuration settings that JD Edwards EnterpriseOne stores in configuration files (such as jde.ini, jas.ini, jdbj.ini, jdelog.properties, and so on). It is highly recommended that you only access and manage these settings for the supported server types using the Server Manager program. See the JD Edwards EnterpriseOne Tools Release 8.98 Server Manager Guide for more information.

You should be familiar with the contents of these documents. The following companion documents contain information that applies specifically to Oracle's JD Edwards EnterpriseOne Package Management tool:

- Installation Guide
- System Administration
- Object Management Workbench
- Server & Workstation Administration
- Configurable Network Computing Implementation

Customers must conform to the supported platforms for the release as detailed in the JD Edwards EnterpriseOne minimum technical requirements. In addition, JD Edwards EnterpriseOne may integrate, interface, or work in conjunction with other Oracle products. Refer to the cross-reference material in the Program Documentation at http://oracle.com/contracts/index.html for Program prerequisites and version cross-reference documents to assure compatibility of various Oracle products.

See Also

JD Edwards EnterpriseOne Tools 8.98 System Administration Guide, "Getting Started with JD Edwards EnterpriseOne Tools System Administration"

JD Edwards EnterpriseOne Tools 8.98 Object Management Workbench Guide, "Getting Started with JD Edwards EnterpriseOne OMW"

JD Edwards EnterpriseOne Tools 8.98 Server and Workstation Administration Guide, "Getting Started with JD Edwards EnterpriseOne Tools Server and Workstation Administration"

JD Edwards EnterpriseOne Tools 8.98 Configurable Network Computing Implementation Guide, "Getting Started with JD Edwards EnterpriseOne Tools Configurable Network Computing Implementation," Configurable Network Computing Implementation

CHAPTER 1

Getting Started with JD Edwards EnterpriseOne Package Management

This chapter discusses:

- JD Edwards EnterpriseOne Package Management overview
- JD Edwards EnterpriseOne Package Management implementation

JD Edwards EnterpriseOne Package Management Overview

Oracle's JD Edwards EnterpriseOne Package Management describes how to set up and maintain processes to develop and deploy custom modifications that are created with Oracle's JD Edwards EnterpriseOne tools. You can use the guide to set up an environment in which you can deploy custom modifications that were made with development tools. It also provides information about applying modification rules, transferring objects, checking out development objects, and working with the data dictionary.

JD Edwards EnterpriseOne Package Management Implementation

JD Edwards EnterpriseOne standardizes and automates software installation, making many steps transparent to users. Technical setup is pre-configured to meet the requirements of many JD Edwards EnterpriseOne customers. In addition, JD Edwards EnterpriseOne products are pre-integrated and share a common database, which reduces the implementation process, minimizes ongoing administration, and provides customers with the flexibility to add new applications, modules, and tools as needed.

CHAPTER 2

Understanding Package Management

This chapter discusses:

- Customer and consultant roles.
- Packages.
- Types of packages.
- Object change tracking.
- The integrity of the production environment.
- Deployment methods.
- JD Edwards EnterpriseOne package implementation.

Customer and Consultant Roles

Typically, both consultants and customers participate in an implementation. Consultants perform these roles:

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

Customers perform these roles, which parallel the consultant roles:

- CNC administrator.
- Application developer.
- Application project leader.
- Hardware, network, and third-party software administrator.

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

CNC Consultant and CNC Administrator

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

Custom Solution Consultants and Application Developers

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 configured software. While the CNC administrator performs the version control functions that build and deploy software, the customer solution consultant must help develop the internal procedures that document the application development cycle for your business.

Application Consultants and Application Project Leaders

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

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

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

Packages

The purpose of a package is to group software modifications so that they can be deployed to workstations, web servers, and enterprise servers. A package defines and contains the components to deploy. A package can contain everything that is needed to run the software (as in an initial installation for a new workstation), or only updates or changes to existing applications.

Why Packages Are Needed

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

Packages enable you to deploy software changes and new applications to your users, or to install the software on a workstation for the first time. After you have defined and built a package, you can deploy it using Oracle's JD Edwards EnterpriseOne Client Workstation Installation application or Deployment Director application (P9631).

You might need to update or set up a workstation or an enterprise, logic, or application server with JD Edwards EnterpriseOne software for a variety of reasons, such as:

- You want to set up a workstation for a new user or role.
- 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 the workstations, and you need to create a minimum configuration.
- You need to update the servers with custom modifications that you have developed using the toolset.

JD Edwards EnterpriseOne provides solutions to meet all of these needs. First, the system enables you to create a package that defines and contains the location of the components that need to be distributed to the workstations. To deploy these components, you must define and build a package.

Types of Packages

You can build these types of packages:

- Full client
- Full server
- Full mobile
- Update client
- Update server
- Update mobile

Full Client Packages

Full packages are static, point-in-time snapshots of the central objects for the path code on which the package is based. A full package contains everything that developers need to run in order to develop in JD Edwards EnterpriseOne software. Specifically, a full package includes a full set of specifications (specs), a full set of business function dlls, source files, object files, header files, bitmaps, an INF file that defines where the foundation, data, and features are located, as well as other information about the package. The package specs are stored in an Oracle Enterprise Edition (OEE) or SQL Server Express (SSE) repository and the specification data is in a platform-independent XML format. Use this package type when you want to create a full workstation configuration.

In a full package, every application for which users are licensed is available to them. Because specs reside on the workstation, information is processed locally, which eliminates network traffic. When you deploy a full package, a few specs and tables might be installed just-in-time, but the impact on network performance is insignificant.

Full packages are primarily for initial installations and are normally deployed using the JD Edwards EnterpriseOne Client Workstation Installation application. You can also use Oracle's JD Edwards EnterpriseOne Deployment Director application to install a full package on a computer on which JD Edwards EnterpriseOne is already installed.

Full Server Packages

Full server packages are the same as client packages except that:

- The server package does not include client-specific business functions.
- During the package build process, you can select to build only the runtime system code on a server package.

  The administrator cannot select a different system code.
- The server package does not include features or local data.
- During the package build process, you can specify the database data source in which to build the spec repository.

  This repository can be shared by several enterprise and web servers. The specs are built in a platform-independent XML format. The spec repository is portable across dissimilar servers.

Full Mobile Packages

Full mobile packages are similar to full packages but contain only a subset of the specs that are delivered with a full package. A mobile package includes all business functions, tables, table event rules (ERs), client-only named ERs, mobile applications, and mobile batch applications (UBEs). The system administrator must flag all required mobile applications and UBEs as *mobile* within Oracle's JD Edwards EnterpriseOne Object Management Workbench.

A full mobile package can be built only if the JD Edwards EnterpriseOne client has at least one environment marked as *mobile* in the path code for which the package is being built. During the package build, MSDE databases are built as a mirror image of the mobile environment's tables based on the mobile environment's Object Configuration Manager (OCM). The MSDE databases are then deployed to a JD Edwards EnterpriseOne mobile workstation on a Microsoft WIN32 client. Only one mobile environment can exist per path code.

When full mobile packages are deployed to workstations, the OCM is changed to point locally to the deployed MSDE database. Users can then access JD Edwards EnterpriseOne mobile applications and perform tasks while disconnected from the network. The data within the local MSDE repository is synchronized with the corporate RDBMS database when the user connects to the network and initiates a synchronization.

Update Client Packages

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

Unless a package includes applications that do not have specs, the update package is a point-in-time copy of the central objects for a particular path code. The specs for the update package are built in XML/TAM format, not XML/RDBMS format. If the update package contains an application without specs, the system copies the required specs directly from the central objects data source at runtime.

When a user signs into JD Edwards EnterpriseOne after receiving one or more update packages, a form that lists the available update packages appears. If the user decides to take one or more of these packages, the system loads the user's workstation with all objects in the package. The objects in an update package replace those same objects on the workstation. All other objects on the workstation remain the same.

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

Because of the way that just-in-time installation works, performance across a wide area network (WAN) might be slow if the update package contains only applications without specs. To improve performance on the WAN, include specs for each application in the package.

All update packages require a full package on which the update package is based. This full package is called the parent package. By default, the parent package is updated by the update package. When this occurs, all objects in the update package are merged into the parent package.

Note. The jde.ini file on the client includes a setting called UpdateParentPackage that determines whether the parent is updated. If this setting is not present or set to Y, the parent package is updated by the update package. If this setting is N, the parent is not updated.

Note. Specs are always in XML format. However, they are stored in a TAM file for update packages and in an RDBMS for a full package. Hence, full package specs are in XML/RDBMS format and update package specs are in XML/TAM format.

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

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

Update Server Packages

Update server packages are the same as update client packages, with these exceptions:

- Update server packages include only the objects that were described previously for full server packages.
- When you are building an update server package, enterprise servers are automatically selected based on the parent package. Update packages can be deployed only to the enterprise and web servers that have the parent package deployed.
- Update packages must be parented to live full packages for them to be deployed.
- Specs are built only on the deployment server.

  They are merged directly into the spec repository at deployment time only.
- Specs are built in XML/TAM format and not in XML/RDBMS format.

Update Mobile Packages

An update mobile package is an update package that is used to update mobile clients. An update mobile package is automatically built when the parent package is a marked as *mobile.*

Package build parses the list of objects in an update package and creates a separate set of specs for those objects that are marked as *mobile,* so a deployed mobile update package contains everything an update client package does except it has a smaller set of specs.

Recommendations for Developers

When developers install a full package, they must select the option to install development objects. When they install the development objects with a full package, these objects will be automatically updated when developers later install an update package. Developers receive source code, header files, libraries, and DLLs.

Object Change Tracking

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

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

Implementing version control might require a staff of information technology professionals. For example, a company has several hundred developers and a complex CNC configuration. To manage version control for multiple developers, the product version control group consists of:

- One manager to oversee coordination within the department.
- One supervisor to coordinate the package builds, coordinate object transfers, and troubleshoot problems.

- Two server specialists to build server packages.
- Four technical specialists to build workstation packages, perform object transfers, and run automated testing before releasing the package to Quality Assurance.
- One night operator to build workstation packages, build server packages, and clear build errors.

Path Codes

If you are not planning any development projects, you need only three path codes (sets of central objects): PY900, PD900, and PS900. If you plan to modify the software extensively, you also should create a development path code (DV900).

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

DV900	The path code that you use for routine development. After successfully testing the objects that you develop, transfer them to the PY900 path code using the Object Transfer application, and distribute them to the users using the package build and deployment process.
PY900	A path code that contains a practice set of objects that you test during the conference room pilot before you transfer objects to production. Use this path code to deploy quick corrections or make minor modifications to objects that you will transfer to production. You also can use this path code to test modifications that were made in the development path code before you transfer the objects to the production path code.
PD900	The production path code from which just-in-time installations and production server objects are deployed. After you test software changes in PY900, transfer them to PD900, and then deploy the changes to the enterprise servers and workstations.
PS900	The set of pristine objects that is included with the software. You should not make changes to this path code other than to apply documented Oracle changes. Use this path code to compare standard software to any custom software that you have implemented in other path codes. You should keep a copy of this path code so that you have an unchanged copy of JD Edwards EnterpriseOne in case you need to undo any of the changes.

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

Suggested Package Names

You should maintain two versions of each package, an A and a B version, so that you can alternate between these versions when you build packages. The advantage of this approach is that users always have a package available to them, even when you are building the latest version of that package. For example, package PRODB would be available to users while you are building PRODA. Then, after you release PRODA, you would build the next package into PRODB, and so on. This setup gives you two full packages (A and B) for production, as illustrated in this table:

PD900FA	Standard Production Full A
PD900FB	Standard Production Full B

Update packages might be named in the following way:

PD900UA	Production Update Package 1
PD900UB	Production Update Package 2
PD900UA	Production Update Package 3
PD900UB	Production Update Package 4

A deployed server package cannot be revised, rebuilt, or deleted. It must be replaced with another package first. For this reason, you must have at least two server packages available so that you can alternate between them.

Note. The maximum length for a package name is eight characters.

See Also

JD Edwards EnterpriseOne Tools 8.98 Security Administration Guide, "Using Security Workbench," Understanding Security Workbench

The Integrity of the Production Environment

As soon as you transfer objects into the PD900 path code, end users can access the changes. Therefore, you should test the modified objects before you transfer them to the production path code.

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

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

  Note. JITI affects users who received an update package containing an application without specs.

  See Chapter 2, "Understanding Package Management," Deployment Methods, page 11.

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

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

The Normal Development Process

These lists provide an overview of how you should perform the normal development cycle.

In the DV900 path code, complete these tasks:

- Make modifications.
- Test the modifications.
- Transfer the objects to PY900.

In the PY900 path code, complete these tasks:

- Build the package.
- Test the modifications.
- Deploy server objects to the CRP path code on the enterprise server, and then test the objects.
- Schedule the package.
- Transfer the objects to PD900.

In the PD900 path code, complete these tasks:

- Build the package.
- Schedule the package.
- Deploy the server objects to the PD900 path code on the enterprise server, and then test the objects.

A Typical Development Process

The following steps are a typical process for modifying objects and deploying them through successive path codes and into the production environment.

1. Check out the objects from the DV900 path code, modify them, test them, and check them back in.
2. Use an incident-tracking system or any numbering system to track changes.

   Always use an incident number when you check in the objects.
3. If the objects need to reside on the logic server, transfer them to the DV900 path code on that server.
4. Test the objects by comparing them to the objects on the server.
5. Use an incident number with Object Transfer to transfer the objects to the PY900 path code.

   Use the checkout log to confirm the transfer (optional). The objects are not in production, but they are now available for you to build a test package in the PY900 path code.
6. Build a full or update package.
7. Test the newly built, unreleased package in the PY900 path code.

   You test the package only by comparing it to workstation processes, not to server processes. Although the name of this package will probably be PY900U1 (update package number 1 for the CRP path code), it is a test package because you have not released it to the users.

8. Schedule the update package to deploy to a test machine and test it in an environment that contains CRP objects with CRP data.
9. Deploy server objects to the PY900 path code on the enterprise server and test them.

   If you prefer, you can build the server package and schedule the deployment at the same time that you build and schedule the workstation package. Building these packages simultaneously can save you time, although this method puts a greater load on the server.
10. Schedule the new package to deploy to CRP users.
11. Use an incident number with Object Transfer to transfer the object to the PD900 path code.

    Use the checkout log to confirm the transfer (optional). The objects are now in the production environment and are available for you to build a package in the PD900 path code.

    Note. If just-in-time installation is enabled, users can access the objects immediately.
12. Build a full or update package for client workstations.
13. Perform a server package build.

    You can transfer the server package now or wait until it has been tested on a workstation.
14. Schedule the new package to deploy to end-user workstations.
15. Deploy the server objects to the PD900 path code on the enterprise server and test them.

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

Developing Short-Term Changes

Sometimes you need to make a simple change to an application that is undergoing major enhancement work in the DV900 path code. When an object has been modified in the DV900 path code, you might get unexpected results if you make a simple change and quickly deploy it. Therefore, you should make the change in both the PY900 and the DV900 path codes, and deploy the change using the PY900 path code. This method enables you to deploy the change quickly to users without interfering with the major enhancement work in the DV900 path code.

Deployment Methods

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

JD Edwards EnterpriseOne offers several deployment applications, each with its own specific purpose and advantages. The method that you select depends mainly on the type of package that you want to deploy.

This section discusses:

- Package deployment.
- Multitier deployment.
- Cumulative and noncumulative update packages.
- Comparing deployment methods.
- Deploying various types of modifications.

- Just-In-Time Installation.
- Recommendations for sites using full packages with JITI.
- Disabling just-in-time installation.

Package Deployment

The JD Edwards EnterpriseOne Deployment Director application enables an administrator to deploy a built package to users, groups, locations, or enterprise servers. For WIN32 clients, the administrator can specify the date and time when the package is made available and whether the package is mandatory or optional.

Users who receive a mandatory package will not be able to access JD Edwards EnterpriseOne until they install the package. Users who receive an optional package can install the package or decline it.

No option is available to schedule a server package for future deployment. The package is immediately deployed.

Multitier Deployment

Multitier deployment enables workstations to install software from more than one deployment location and more than one deployment server. You should consider multitier deployment if your site has more than 50 workstations performing software installations per day, or if you are deploying JD Edwards EnterpriseOne software across a WAN connection.

Cumulative and Noncumulative Update Packages

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

1. Change the package assembly status to *Inactive.*
2. Go to the Package Revisions form.
3. Add the changed or new objects to the package.
4. Rebuild the package.
5. Redeploy the package.

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

Comparing Deployment Methods

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

- A new user loading a new machine should use the Install Manager to load a full package, plus any update packages that you have instructed users to load since the last package build.

  Therefore, you need a manual tracking system to track which update packages must be applied after installing a particular package.
- All update packages must use the JD Edwards EnterpriseOne Deployment Director application to be scheduled to workstations unless you are using the Push Installation feature.
- Full packages can also use JD Edwards EnterpriseOne Deployment Director if JD Edwards EnterpriseOne is already loaded on a machine.

You can use Oracle's JD Edwards EnterpriseOne Push Installation feature to deploy to a machine that does not have JD Edwards EnterpriseOne installed.

- Use Oracle's JD Edwards EnterpriseOne Silent Installation application to submit a workstation installation request through command line arguments.

  Do not use this application for an initial installation.

- Use Oracle's JD Edwards EnterpriseOne Multitier Deployment process to install from more than one deployment location.

  You should consider this method if you have more than 50 workstations performing software installations per day or if you have users on a WAN.

- Use the JD Edwards EnterpriseOne Deployment Director application when you need to push objects in a server package to enterprise servers.

Deploying Various Types of Modifications

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

This table shows which types of changes are installed with a full package, an update package with specs, and an update package with no specs.

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

Modification	Full Package	Update Package with Specs, Business Functions, and Named Event Rules (NERs)	Update Package with Business Functions and NERs but no Specs
Imbedded event rules in a report	X	X	
Report data structure	X	X	
Report vocabulary overrides	X	X	
Report processing options	X	X	
Versions and processing option values (depends on processing options)	X	X	
Imbedded event rules in versions	X	X	
Processing option templates	X	X	
Business Views			
Added or changed fields	X	X	
Tables			
Structure (specifications)	X	X	
Indexes	X	X	
Joins	X	X	
Generic text data structure	X	X	
Data dictionary items			
Foundation code (required for full packages, optional for update packages)	X	X	X
Foreign languages	X	X	
Non-Oracle objects (custom items must be defined in the JD Edwards EnterpriseOne Central Objects database, and can be deployed through any package type)	X	X	

Modification	Full Package	Update Package with Specs, Business Functions, and Named Event Rules (NERs)	Update Package with Business Functions and NERs but no Specs
Replicated local data (required for full packages, optional for update packages)	X	X	X
New icons	X	X	

For an update package with JITI, these changes are installed:

- Applications.
- Imbedded ER.
- Vocabulary overrides (FDA text).
- Data structure.
- Processing options (report).
- Batch applications.
- Report.
- Imbedded ER in a report.
- Report data structure.
- Report vocabulary overrides.
- Report processing options.
- Versions and processing option values (depends on processing options).
- Imbedded ER in versions.
- Business Views.
- Added or changed fields.
- Miscellaneous.
- Foundation code (required for full packages, optional for update packages).
- Foreign languages.
- Non-Oracle objects (custom items can be deployed through any package type).
- Replicated local data (required for full packages, optional for update packages).

Just-in-Time Installation

Just-in-Time Installation (JITI) occurs when the system retrieves an application at runtime and loads it on the workstation the first time that you select that application from the menu. Loading happens only once; the next time that you select the same application, it is still loaded on the workstation. JITI applies only to applications. Business functions cannot be installed through JITI.

JITI works when the workstation receives an update package that does not contain specs. Update packages that contain specs do not require the JITI process.

When you receive an update package that contains a changed application without the new specs, the system first determines whether specs for the application reside on the workstation. If so, it deletes from the workstation old versions of that application. Then, the next time that you select that application from the menu, the system loads the new version of that application.

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

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

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

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

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

Recommendations for Sites Using Full Packages with JITI

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

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

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

Disabling JITI

Select one of these methods to disable JITI:

- Use Oracle's JD Edwards EnterpriseOne Work with Environments application (P0094) to disable JITI for the environment.

  When JITI is disabled, users who sign on to that environment can access only those applications for which they have specifications. You should use this method during the cumulative installation process when you update the production central objects with new changes.

- Use application security to disable JITI for a particular application, for an end user, or for a role profile.

  When a user accesses an application for which the specifications do not reside on the user's workstation, the system first reviews P0094 to determine whether JITI is disabled for the entire environment. If JITI is on for the environment, the system determines whether application security prevents the user from using JITI. Application security has a field called *not allowed to install.*

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

Package Implementation

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

1. Assemble the package.

   During this step, you specify the type of package that you are building and provide a name, path code, and package description. Next, you assemble the package by specifying the objects, data, foundation, features, and so on that you want to include in the package. If you are building an update package, you can specify individual objects to include.

   To simplify the process of assembling a package, Oracle's JD Edwards EnterpriseOne Package Assembly application (P9601) includes the Package Assembly Director, which displays a series of forms that guide you through the steps of naming the package and adding the objects that you want to include in the package.

2. Define the package build.

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

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

   You also need to specify whether the package is for a workstation, a server, or both. If the package is for servers, you must specify the servers for which the package should be built and select the spec database data source.

   To simplify the build process, Oracle's JD Edwards EnterpriseOne Package Build Director application (P9621) includes the Package Build Definition Director, which displays a series of forms that guide you through the steps of specifying where to build the package, whether to include specifications, whether to compress or build business functions, and so on.

3. Build the package.

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

4. Schedule the package for deployment.

   After you have defined and built the package, it is ready for distribution. Depending on the package type, you can deploy packages through JD Edwards EnterpriseOne Client Workstation Installation application or Deployment Director application.

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

When you schedule the package, you can indicate whether package installation is mandatory or optional. At this same time, you can specify whether you want the package to be deployed using push installation, which requires no interaction with the package recipient.

5. Deploy the package to deployment, enterprise, and web servers.

   Use the JD Edwards EnterpriseOne Deployment Director application to move any changed objects to the enterprise server.

   If you specify a server during the package build definition process, the system 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.

   Web servers automatically retrieve the package information from their configured business function logic servers.

   See Chapter 7, "Deploying Packages," Understanding Deployment to Web Servers, page 147.

CHAPTER 3

Understanding Objects

This chapter discusses:

- Objects
- Modification rules

Objects

This section discusses:

- Object storage.
- Object movement.
- Performing backups and restoring objects.
- Correlating replicated and central objects.

Object Storage

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

The system stores objects in three formats:

- Central objects are stored in a relational database format.

  Objects are stored in a central location to enable deployment and development. Central objects consist of object specifications for each JD Edwards EnterpriseOne object and C components for code-generated objects. Central object specifications are stored in a relational database on a data server.

- Package objects, or replicated objects, are stored in XML format in a relational database.

  Package objects are created during the package build process. You can specify the database data source in which to build the shared spec repository. This shared repository usually exists on a data server. Several enterprise servers and web servers can share this repository. A set of replicated objects is also stored in a local database on each developer workstation.

- Serialized objects are stored in database tables and used by web clients at runtime.

  Web servers use on-demand generation to create serialized objects from the shared spec data source. The generator converts specs into Java code, which enables you to access the JD Edwards EnterpriseOne applications in HTML. The system stores the objects in a relational database and retrieves them at runtime.

Object Movement

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

- Check in or check out objects.
- Add existing objects to a project.
- Perform a get object in Oracle's JD Edwards EnterpriseOne Object Management Workbench application (P98220).
- Run Oracle's JD Edwards EnterpriseOne Workstation Installation application.

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

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

Object Type	Movement
Table (object type TBLE)	These objects move: • Table specs • Table event rule specifications (If the tables have event rules) • Source files (*.c) • Table header files (*.h) • Object files (*.obj) (If the objects have event rules) • Table event rule include files (*.hxx) (If the tables have event rules) The table header is not the same as the actual table that resides in a database. The table itself is created through Table Design Aid when you generate it. The JD Edwards EnterpriseOne Workstation Installation program copies this table to the workstation if the table is stored in the Supported Local Database.
Business view (object type BSVW)	These objects move: • Specifications • .h files (if generated)

Object Type	Movement
C business function (object type BSFN, source language C)	These objects move: • Specifications • Source files (*.c) • Header files (*.h) • Object files (*.obj)
NER business functions	Business function event rules (object type BSFN) can be checked out. When business function event rules are checked out, the .h file moves to the include directory, the .c file moves to the source directory, the .obj moves to the obj directory, and the local specifications are updated. These objects move: • Specifications • Source (*.c) • Header (*.h) • Object (*.obj)
Business function data structure (object type DSTR)	Specifications move.
Embedded event rules	You cannot check out embedded event rules. Embedded event rules move when you check out the object in which the event rule is embedded. For example, if embedded event rules are attached to a table, interactive application, or batch application, when you move the table or application, the specifications for the embedded event rules move with it.
Media object data structure (object type GT)	Data structure specifications move.
Interactive application (object type APPL)	These specifications move: • Application • Form • Form data structure • Embedded event rules

Object Type	Movement
Batch application (object type UBE)	These objects move: • Report and event rule specifications (check in and check out the version separately from the report) • .h file (if generated by the developer)
Business Services (object type BSFN, source language Java) These objects are type BSFN except for: • F9860.SIPARDLL = “ • F9860.SISRCLNG = “SBF”	These objects can be checked out and checked back in. The IDE for development of business services objects is JDeveloper. When they are checked out, the <F9860.SIOBNM>.zip file is copied from the path code check-in location on the deployment server (<path code>/java/sbfjars) to the <EnterpriseOne client install>/<path code>/java/source/<F9860.SIOBNM> folder on the Microsoft WIN32 client, and the contents are extracted. When they are checked in, the contents of the <EnterpriseOne client install>/<path code>/java /source/<F9860.SIOBNM> folder are compressed to a <F9860.SIOBNM>.zip file and the file is copied to the path code check-in location.

See Also

JD Edwards EnterpriseOne Tools 8.98 Object Management Workbench Guide, "Getting Started with JD Edwards EnterpriseOne OMW," JD Edwards EnterpriseOne OMW Overview

Performing Backups and Restoring Objects

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

Consider these scenarios and solutions when developing your backup strategy:

- A company does not allow the developers to back up directory data to the server because of space concerns.

  Developers are required to check in their development objects at specific time intervals, such as every eight hours, to avoid rework. Unless you have unlimited disk space on a file server to enable 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 can check in unfinished or malfunctioning applications to the DEV path code.

- For workstation backups, end users should not have non-replicated data on their machines.
- For development server backups: At a certain company, the IT department backs up both the development file server (normally the deployment server) and necessary databases (central objects, Object Librarian, and data dictionary).

  When a developer needs to restore a particular object from backups, a database administrator restores the export to a path code called Restore. The developer checks out the object from Restore, ensures that the object functions as expected, and checks the object into the normal development path code.

- For deployment server backups: In most cases, you do not need to back up the entire server nightly.

  However, under certain conditions, you might need to back up these directories nightly:

  - The DEV path code, if you are modifying objects, building new packages, or updating the database that is delivered during a workstation installation.
  - MEDIA OBJ, if your media objects reside on the deployment server.
  - Data sources in Oracle or SQL Server, if your system data or any other important data is stored on the deployment server.
- For enterprise server backups:
  - Back up the DBMS nightly.

    You should use the backup tool that your database vendor provides.
  - Back up objects by backing up the entire directory.
  - Also back up the PROD and DEV path codes and the jde.ini file.

    Path codes are updated when the Version Control administrator deploys an object that was modified by developers who are authorized to access the Server Package application, and by end users who create new batch versions that will be run on the server.

See Also

JD Edwards EnterpriseOne Tools 8.98 Server and Workstation Administration Guide, "Backing Up JD Edwards EnterpriseOne Tables," Backing Up JD Edwards EnterpriseOne Tables on Servers

Correlating Replicated and Central Objects

This table describes the correlation between the spec data in the package build objects and the central objects that are stored in a relational database that has a binary large object (BLOB):

Replicated Package Build Object	Central Object
F98710<package name>	The F98710 table contains one record for each table.
F98711<package name>	The F98711 table contains one record for each column in a table.
F98712<package name>	The F98712 table contains one record for each table index.
F98713<package name>	The F98713 table contains one record for each column in an index.
F98720<package name>	The F98720 table contains one record for each business view.
F98740<package name>	The F98740 table contains one record for each event that has event rules for applications, reports, or tables. (Named event rule links are stored in the F9862).
F98741<package name>	The F98741 table contains one record for each line of event rules.
F98743<package name>	The F98743 table contains one record for each business function, processing option, form interconnection, report interconnection data structures, and power form.
F98750<package name>	The F98750 table contains override text for applications.

Replicated Package Build Object	Central Object
F98751<package name>	The F98751 table contains one record for every column, grid line, button, hyperitem, control, and so on, in the application.
F98752<package name>	The F98752 table contains one record for each application. If the application has processing options, that information is also stored in the record.
F98753<package name>	The F98753 table contains one record for each form (and also includes references to the data structures).
F98760<package name>	The F98760 table contains override text for batch reports.
F98761<package name>	The F98761 table contains one record for each section, column, sort, constant, and so on, in Batch Reports and Versions.
F98762<package name>	The F98762 table contains one record for each function in BSFN.
F98770<package name>	The F98770 table contains only one record for each package. This table is empty in Central Objects.
CGTYPE	Code Generator Form Types are stored in specification format only.
DDDICT DDTEXT	One record exists for each data dictionary item that has been just-in-time installed. This is data dictionary text.
GLBLTBL	This cache information from data dictionary and table specifications contains runtime table and override information. This is built dynamically the first time that a table is used.
SMRTTMPL	This is field information required by the data structure.
F9200 F9202 F9203 F9207 F9210	F9200 F9202 F9203 F9207 F9210
NEXTID	The F98701 table contains a local record of next IDs that are assigned to each workstation.

Modification Rules

This section discusses:

- Types of modifications.
- Objects that an upgrade preserves and replaces.

Types of Modifications

Because JD Edwards EnterpriseOne Development Tools are comprehensive and flexible, you can configure certain aspects of business solutions and applications without making custom modifications. This concept is referred to as *modless modifications*. Modless modifications are modifications that you can perform easily without the help of a developer. You can perform modless modifications on:

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

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

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

Developers may need to modify the JD Edwards EnterpriseOne software more extensively. To ensure that the modifications perform like modless modifications and to provide a seamless and predictable upgrade to the next release level, you should verify that any software modifications that you make comply with the recommended rules and standards.

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

The system tracks all custom modifications as you check them into the server. Before you perform an upgrade, you can run Oracle's JD Edwards EnterpriseOne Object Librarian Modifications report (R9840D) to see a list of the changed objects.

The system consists of control tables, such as menus, UDCs, versions, and the data dictionary, and transaction tables, such as the F0101 table. The system provides control tables, which contain data that you can modify, as well as transaction tables, which contain your business data.

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

Objects That an Upgrade Preserves and Replaces

Modification rules exist for these types of objects:

- Interactive applications
- Reports
- Application text changes
- Table specifications
- Control tables
- Business views
- Event rules
- Data structures
- Business functions
- Versions
- Business services

General Rules for Modification

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

- Preserves

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

- Replaces

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

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

These general modification rules apply to all objects:

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

  The system uses its own reserved system codes that enable it to categorize different applications and vertical groups. When you use system codes 55 through 59 for your custom modifications, the system does not overlay your modifications with new applications.

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

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

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

Interactive Applications

Do not delete controls, grid columns, or hyper items on existing applications. If you do not want to see them, hide or disable them. The system might use these items for calculations or as variables, and deleting them might disable major system functions.

This table describes the interactive application elements that are preserved or replaced during an upgrade.

Object	Preserved	Replaced	Comments
New applications.	X		You can either create a new application, or copy an existing application using the Copy feature in Application Design Aid. This feature enables you to copy all of the application specifications, including event rules. If you use the Copy feature to copy an existing application for some modifications, during an upgrade your new application does not receive any changes that the system might have made to the original application.
New hyper items added to existing forms.		X	
New controls added to existing forms.		X	
New grid columns added to existing forms.		X	
Style changes.		X	Style changes include fonts and colors. New controls have the standard base definitions. If you adjust the style, you need to also adjust the styles for any new controls that you added to an application.
Code-generator overrides.		X	

Object	Preserved	Replaced	Comments
Data dictionary overrides.		X	
Location and size changes.		X	In a subsequent release of the software, a new control might be placed in the same location that you have placed a custom control. In this case, the new control appears on top of your custom control. This situation does not affect the event rules or the functions of the application. After the upgrade, you can use Application Design Aid to rearrange the controls.
Sequence changes for tabs or columns.		X	The upgrade process adds new controls to the end of your custom tab sequence. You can review the tab sequence after an upgrade.
Custom forms on existing applications.		X	Instead of adding custom forms to existing applications, create a custom application using system codes 55 through 59, and then place the custom form on that custom application. You can then add to existing applications Form exits and Row exits that call your custom forms within your custom applications. System performance is not adversely affected when you call an external application from a row exit instead of from a form within the application.

Note. None of the custom modifications to the JD Edwards EnterpriseOne applications are preserved during the Batch Specification Merge process. Instead, administrators must manually retrofit the modifications from a JD Edwards EnterpriseOne workstation with the help of Oracle's JD Edwards EnterpriseOne Visual ER Compare and FDA Compare tools when the upgrade is complete.

Reports

For Oracle's JD Edwards EnterpriseOne Report Design Aid specifications, do not delete objects on existing reports. Hide the objects that you do not want to appear. The system might use these objects for calculations or as variables, and deleting them could disable major system functions.

This table describes the report elements that are preserved or replaced during an upgrade:

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

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

Application Text Changes

This table describes the application text that is preserved or replaced during an upgrade:

Object	Preserved	Replaced	Comments
Overrides performed in JD Edwards EnterpriseOne Application Design Aid.		X	Use the JD Edwards EnterpriseOne Visual Compare tools.
Overrides performed in JD Edwards EnterpriseOne Report Design Aid.	X		
Overrides performed in JD Edwards EnterpriseOne Interactive Vocabulary Override.		X	Use the JD Edwards EnterpriseOne Visual Compare tools.
Overrides performed in JD Edwards EnterpriseOne Batch Vocabulary Override.	X		

Table Specifications

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

This table describes the table specification elements that are preserved or replaced during an upgrade:

Object	Preserved	Replaced	Comments
New Tables.	X		
Custom indexes to tables.	X		
Columns added to or removed from existing tables.		X	This object includes changing field length, field type, and decimal position. Instead of adding a new column to an existing table, use a tag table with system codes 55 through 59.

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

Control Tables

Control tables contain 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 system tables, as the basis for the data merge.

This table describes the control table elements that are preserved or replaced during an upgrade:

Object	Preserved	Replaced	Comments
Data dictionary custom changes.	X		This object 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 system data items, your changes override. Create new data items using system codes 55 through 59.
UDCs.	X		The upgrade process merges any new, hard-coded values. (Oracle-owned values are stored in systems 90 and above, and H90 and above.) The process also reports any hard-coded values that conflict with your custom values.

Object	Preserved	Replaced	Comments
Menus.	X		In case of a conflict with base menus, your custom changes override.
Columns added or removed from existing control tables.		X	

Business Views

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

This table describes 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 that are added to existing business views.	X		
Columns that are removed from business views.		X	

Event Rules

During the upgrade process, the system checks for custom event rules that conflict with new event rules that the software installs. If a conflict exists, the system disables the custom event rules and appends them to the end of the new event rules.

This table describes the event rule elements that are preserved or replaced during an upgrade:

Object	Preserved	Replaced	Comments
Custom event rules for custom applications, reports, and tables.	X		
Custom event rules for custom business functions.	X		
Custom event rules on a new custom control.		X	Use the JD Edwards EnterpriseOne Visual Compare tools.
Events for system applications, reports, and tables that do not have any system event rules attached to the same event.	X		

Object	Preserved	Replaced	Comments
Events for system business functions that do not have any system event rules attached to the same event.	X		
Events for system applications, reports, and tables that have existing event rules attached to the same event.		X	Use the JD Edwards EnterpriseOne Visual Compare tools.
Events for system business functions that have event rules attached to the same event.		X	Use the JD Edwards EnterpriseOne Visual Compare tools.

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

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

Data Structures

This table describes the data structure elements that are preserved or replaced during an upgrade:

Object	Preserved	Replaced
Custom forms' data structures.	X	
Custom processing options' data structures.	X	
Custom reports' data structures.	X	
Custom business functions' data structures.	X	
Custom generic text's data structures.	X	
Modifications to existing system forms' data structures.		X
Modifications to existing system processing options' data structures.		X
Modifications to existing system reports' data structures.		X

Object	Preserved	Replaced
Modifications to existing system business functions' data structures.		X
Modifications to existing system generic text's data structures.		X

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

Business Functions

For any new custom business functions (BSFNs), create a new custom parent DLL to store your custom modifications. Always use the standard application program interface (API), jdeCallObject, to call other business functions from within a business function.

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

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

This table describes the business function elements that are preserved or replaced during an upgrade:

Object	Preserved	Replaced	Comments
New custom business function objects.	X		
Modifications made to existing system business function objects.		X	Named event rule (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.

This table describes the versions elements that are preserved or replaced during an upgrade:

Object	Preserved	Replaced
Non-Oracle versions.	X	
Version specifications.	X	
Processing option data.	X	

Object	Preserved	Replaced
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-Oracle versions that use processing option templates. A warning is issued at runtime, and some data might be lost.

Also, event rule modifications for custom versions of JD Edwards EnterpriseOne templates are not reconciled with the parent template.

Business Services

This table describes the business services elements that are preserved or replaced during an upgrade.

Object	Preserved	Replaced	Comments
New custom business service	X		Custom objects are always preserved.
New object within an existing JD Edwards EnterpriseOne business service		X	JD Edwards EnterpriseOne objects are always replaced.
Changed object within an existing JD Edwards EnterpriseOne business service		X	JD Edwards EnterpriseOne objects are always replaced.
Business services selected as web services	X		Within P9603, you select which business services will be exposed as web services. This selection is preserved.

CHAPTER 4

Assembling Packages

This chapter provides an overview of the package assembly process and discusses how to:

- Verify a path code for package assembly.
- Assemble a package using Director mode.
- Assemble a package using Express mode.
- Revise an existing package.
- Activate an assembled package.

Understanding the Package Assembly Process

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

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

Package Assembly Director

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

You can enter default information on each of the main forms of the Package Assembly Director, or you can access subforms from each of the main forms to configure the information. The steps are the same whether you are adding components for the first time or revising a previously assembled package.

You can also access any previously assembled packages and view information about these packages by clicking the plus (+) symbol of the package on the Work with Packages form. 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, and language.

When you finish adding the selected components to a package, those components appear on the specific form for that component, the Package Component Revision form, and the Work with Packages form of the Package Assembly Director.

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

Form	Function
Package Assembly Directory form	Use this form to review introductory information about the Package Assembly Director.
Package Information form	Use this form to enter the package name, description, and corresponding path code.
Package Type Selection form	Use this form to indicate whether you are creating a full 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 were creating a package to update your original package called APPL_B, you would enter APPL_B as the parent package for your update package.
Foundation Component form	Use this form to enter the location of the foundation. A foundation is the code that is required to run all applications. It is required for all full packages. If you do not specify a foundation path for full packages, the system uses the default foundation path. Update packages use the foundation for the parent package unless you specify another foundation.
Database Component form	Use this form to specify the location of the database to be included in the package. For full packages, if you do not specify a database location, the system uses the default database path. Update packages do not require a database.
Default Object Component form (for full packages only)	Use this form to verify the deployment data source. When you build a full package, the system retrieves the objects that are included in the package from the deployment data source that is associated with the path code that you specified for the package.
Object Component form (update packages only)	Use this form to specify the individual objects that you want to include in the package. You can add any of these objects: • Interactive or batch applications • Business functions • 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 an application or another feature.
Language Component form	Use this form to include in your package language specifications for a language other than English.

Form	Function
Package Component Revisions form	Use this form to review the information that you entered on the previous forms. You can modify any or all of your selections on this form.
Mobile Client Database Revisions form	Use this form to build a mobile client package and to select the mobile client databases.

Accepting Default Values

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

Forms determine the default values based on these criteria:

- Foundation

  The default foundation location is the server share path under the path code for the package.

- Database

  The default database location is the server share path under the path code for the package.

- Objects

  The default location for full packages is the deployment data source.

- Language

  The default language is English.

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

If you are building a full 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, and language. This method applies only to full packages. For an update package, if you accept the defaults but do not include any objects, the system creates an empty package.

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

Verifying a Path Code for Package Assembly

Before you assemble a package, you can verify that the path code from which the package is built is configured correctly.

This section provides an overview of the process to verify a path code and discusses how to verify a path code for package assembly.

Understanding the Process to Verify a Path Code

The verification process tests the environment, machines, and tables before a package is submitted. By verifying your environment, you eliminate the chance that your package build will fail due to configuration issues. This verification can save many hours in building a package.

During the verification process, the program verifies these items:

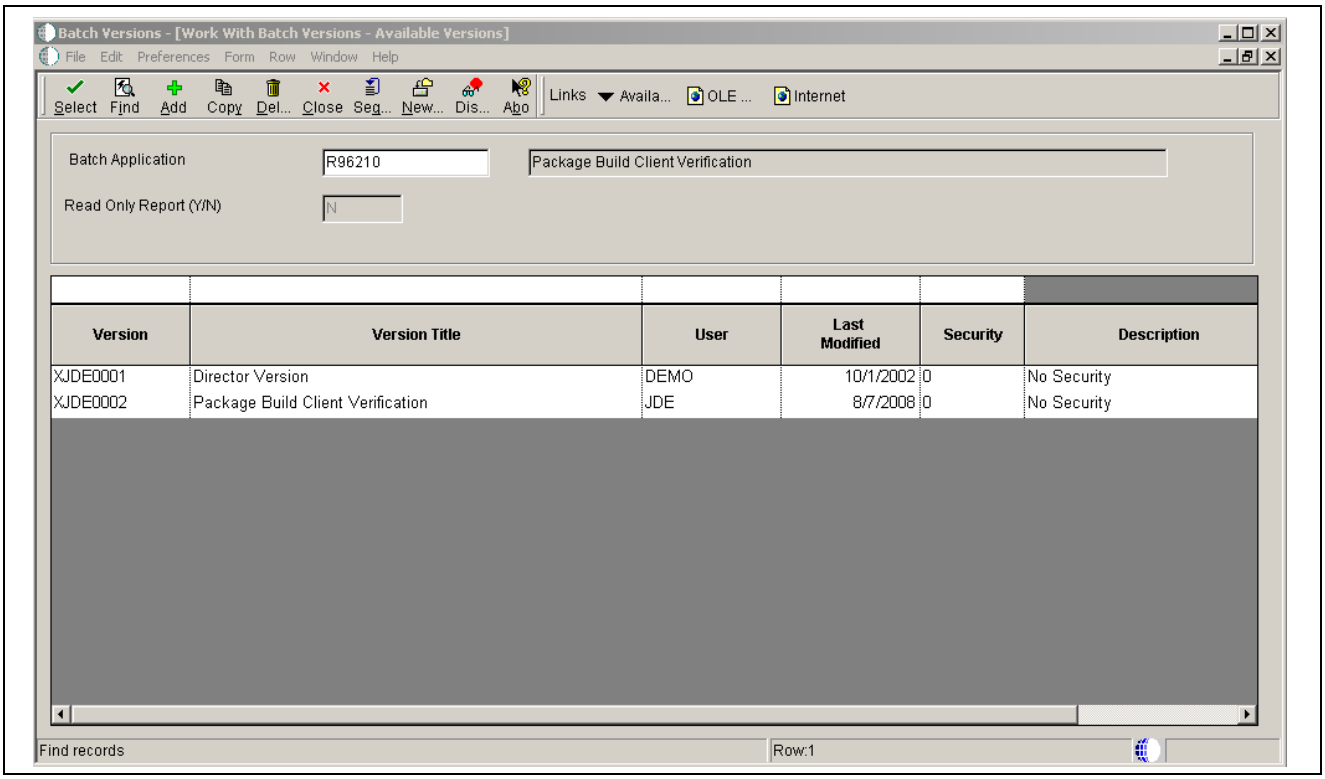
- Disk space is adequate.
- Central objects and package build tables are accessible.
- User has permissions to create directories on the deployment server and enterprise server.
- Required service pack is installed.
- Required Microsoft Data Access Components (MDAC) are installed.
- Machine tables are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

Form Used to Verify a Path Code for Package Assembly

Form Name	FormID	Navigation	Usage
Work with Batch Versions - Available Versions	W98305A	System Administration Tools, Package and Deployment Tools, Package Assembly Select Build Verification from the Form menu.	Verify the path code for package assembly.

Verifying a Path Code for Package Assembly

Access the Work with Batch Versions - Available Versions form.



Work with Batch Versions — Available Versions form

1. Select a version and click Select.
2. On Version Prompting, click Submit.
3. Complete the Processing Options fields, and click OK.
4. On Printer Selection, select the desired printer, and click OK.

Assembling a Package Using Director Mode

Assembling a package in Director mode can involve configuring additional components, such as a foundation location, database location, or features. A feature is a set of files or configuration options, such as registry settings, that is copied to a workstation or server to support an application or other function. Like objects, features are built into a package and deployed to workstations and servers.

Note. Assembling a package in Express mode, rather than Director mode, is recommended in order to simplify and speed up the assembly process. Director mode can be used if you are unfamiliar with the process and would like to walk through each form consecutively. However, Express mode is the default mode for the Package Assembly application. This default behavior can be changed with a processing option.

This section discusses how to:

- Use Director mode to assemble a new package.
- Select mobile packages.
- Add a new foundation location.

- Add a database location.
- Add features to a package.
- Review the package assembly selections.

Using Director Mode to Assemble a New Package

From System Administration Tools, select the Package and Deployment Tools menu, Package Assembly.

1. On the Work with Packages form, click Add, and then on the Package Assembly Director form, click Next.
2. On the Package Information form, complete the Package Name, Description, and Path Code fields.

   Note. The name of the package cannot be longer than eight characters.

   Note. You can build a single foundation package to deploy to all path codes by entering **ALL* in the Path Code field. This option enables you to create an update package for service packs that can be installed to any path code. If you enter **ALL* in the Path Code field, the application does not enable you to select, build, or deploy any objects (such as specs, business functions, and so on) in the package; you can only deploy a foundation. The package is built to a directory called all_packages, which is located under the release path (for example, e900/all_packages). This package can be deployed to any path code.

   Before you can use this option, you must define the *ALL path code in the Path Code Master application.

   See *JD Edwards EnterpriseOne Tools 8.98 Configurable Network Computing Implementation Guide*, "Understanding Path Codes," Setting Up Path Codes.
3. Select the Director option, and click Next.
4. On Package Type Selection, complete these fields, and click Next:

Field	Description
Full	Select to create a full package that contains all specifications and foundation code.
Update	Select to create an update package with specific items that can be deployed to specific users. If you are building a foundation package to the *ALL path code, the application automatically selects an update package.
Parent Package	Indicate the parent package that the update package is based on or related to. This information is used by the system to determine how to build business functions.

Note. If you select to build an update package, the program disables the option to build a mobile client package. However, the program will automatically build a mobile client package for the update if the parent package includes a mobile client.

5. Click Build Mobile Client Package if you want to build a mobile package, and then click Next.

   Note. The Build Mobile Client Package form is bypassed if you do not have a mobile environment in the path code for which you are building the package.

   See Chapter 4, "Assembling Packages," Selecting Mobile Packages, page 44.

6. On the Foundation Component form, perform one of these actions:
   - For full packages, accept the default location by clicking Next, or click Browse to specify another foundation location.
   - For an update, click Clear unless the package includes a foundation, and then click Next.

   See Chapter 4, "Assembling Packages," Adding a New Foundation Location, page 44.

7. On the Database Component form, perform one of these actions:
   - For full packages, accept the default location, or click Browse to specify another database location.
   - For an update, click Clear unless the package includes a foundation.

   See Chapter 4, "Assembling Packages," Adding a Database Location, page 46.

8. Complete one of these actions:
   - If you are assembling a full package, click Next.

     For a full package, the system builds your package from the deployment data source that is associated with the default object path. Verify that the correct location appears on the form, and proceed to the next step.
   - If you are assembling an update package, click Next on the Database Component form, and then proceed to the next step.

9. On the Object Component form, to add an object, click Browse.

   The Object Component form only appears when you are assembling an update package. If you are assembling a full package, the Default Object Component form appears and you cannot add objects.

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

    When you revise a previously assembled package, objects that you added earlier also appear on the Object Component Selection form.

11. On the Object Component form or Default Object Component form, click Next.

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

12. To add a feature, click Browse.

13. On the Feature Component Selection form, click Find to display a list of features, select one or more features, and then click Select to add the features that you want to include in your package.

    To select multiple features, press the CTRL or SHIFT key while clicking features, and then click Select.

14. Click Close to return to the Features Component form.

    See Chapter 4, "Assembling Packages," Adding Features to a Package, page 46.

15. On the Feature Component form, click Next.

16. On the Language Component form, click Next if English is the only language that you want to configure.
17. To add a language to the language specifications for your package, double-click its row header in the detail area, and click Next..

    If you add a language to your package, only that language will be included. For example, if you add French, English will not be included even though it is the default language. To include two languages (such as French and English), you must select the detail records for both languages.
18. Continue with the task Reviewing the Package Assembly Selections.

Selecting Mobile Packages

Access the Mobile Client Database Revisions form by clicking Next on the Package Type Selection form.

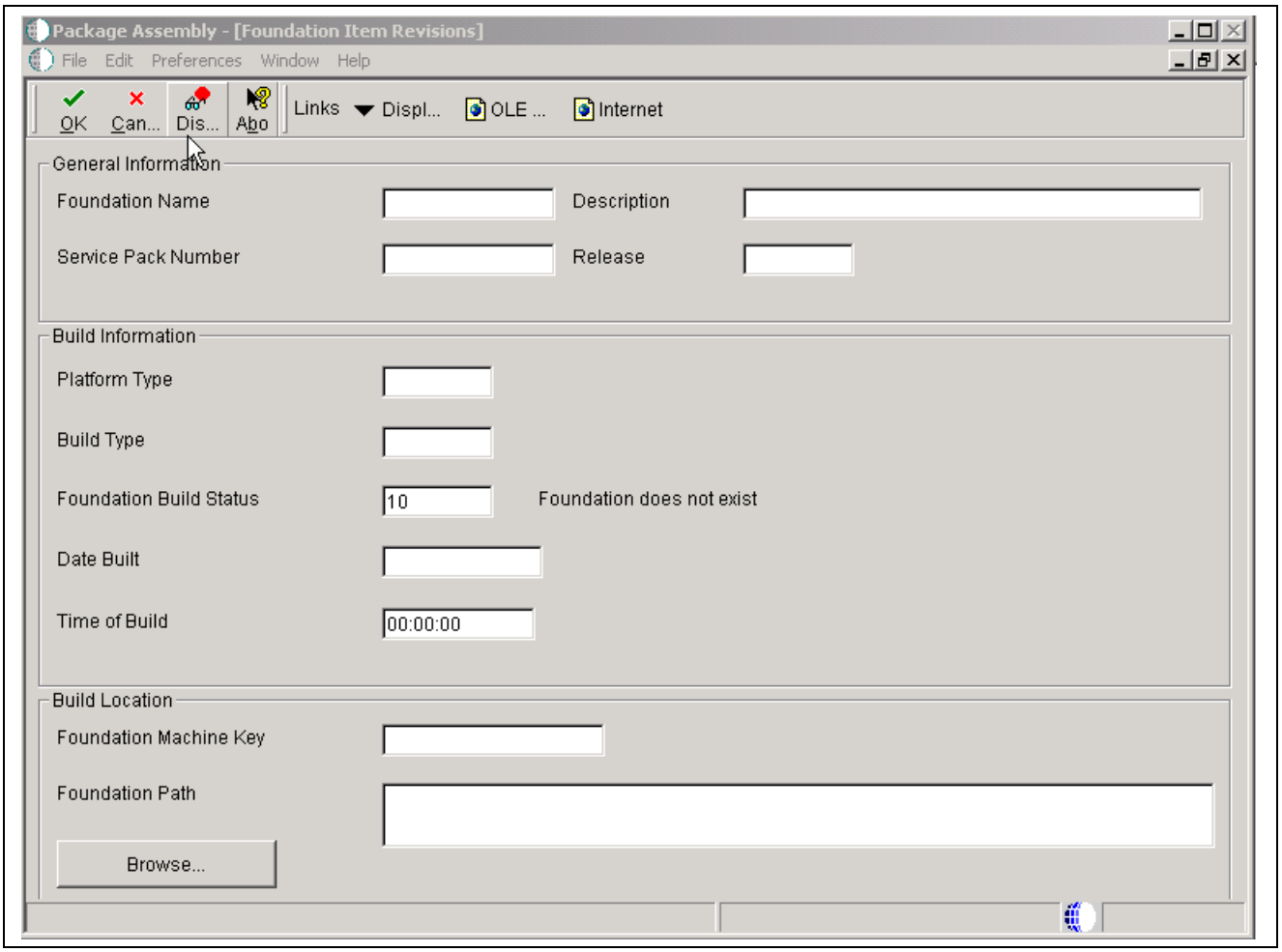
1. Select Build Mobile Client Package if you want to build a mobile package.

   If you select this option, the package build program will create a mobile package name by appending *_M* to the name of the existing package. The program also lists the database names and owners of the mobile client databases.
2. Select the MSDE databases that you want to include in the mobile package, and then click Next.

Note. You cannot change the owners of the mobile databases. Deselecting the databases is not recommended unless you are an advanced package build user.

Adding a New Foundation Location

Access the Foundation Item Revisions form by clicking Add on the Foundation Component Selection form.



Foundation Item Revisions form

1. Enter a foundation ID in the Foundation Name field.

   This is the code that is required to run all applications. A foundation ID is required for all full packages. For full packages, if you do not select a foundation, the default foundation is used. The default foundation is determined through the release that is 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.

2. Enter a service pack number in the Service Pack Number field, if appropriate.

   A service pack is an update to the foundation code that is delivered between major releases and cumulative releases of the software.

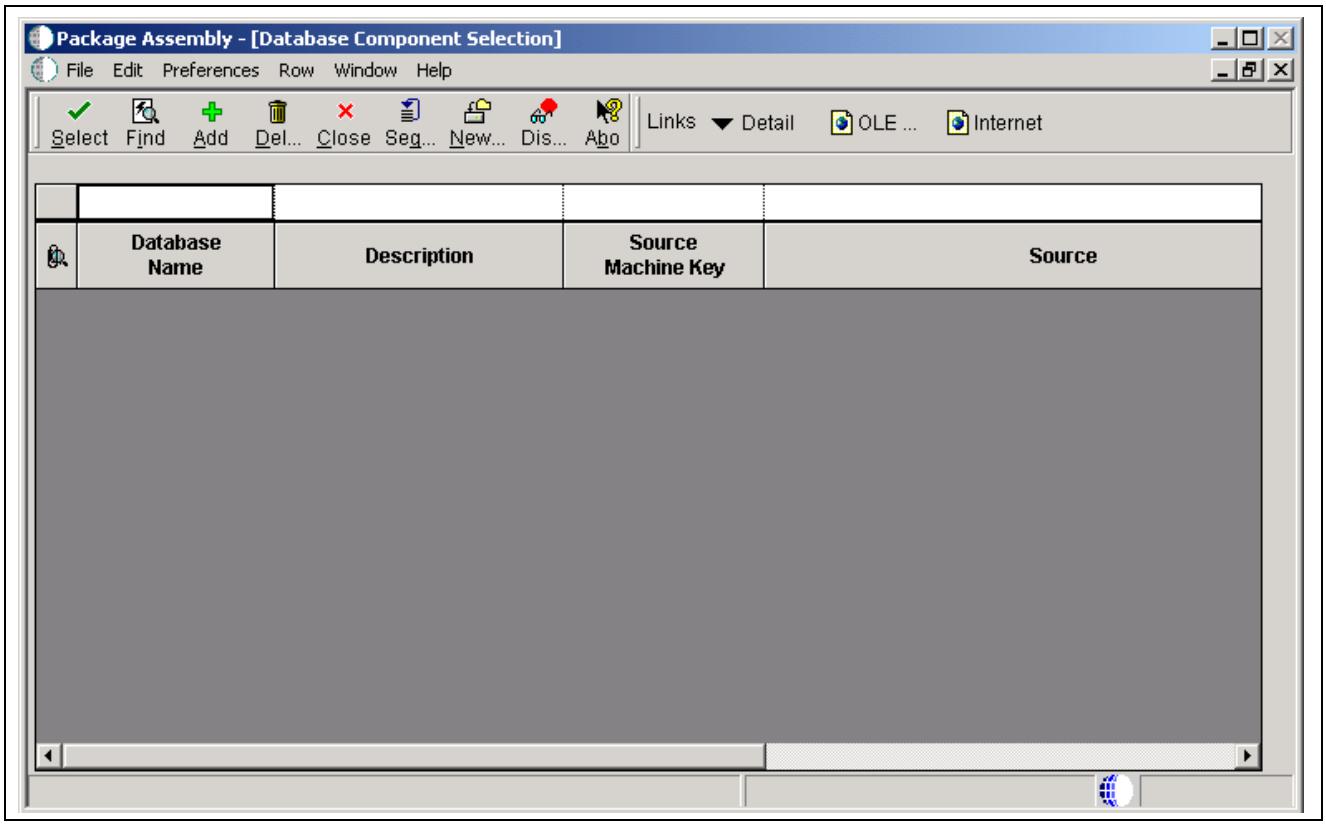
3. Enter the release number with which this foundation is associated in the Release field.
4. Enter the host machine type in the Platform Type field.
5. Enter the compiler configuration to use for the software build in the Build Type field.
6. Enter the current status of the build process for foundation in the Foundation Build Status field.
7. Enter the date that the software build finished in the Date Built field.
8. Enter the time at which the software build finished in the Time of Build field.
9. Enter the name of the deployment server where your custom foundation resides in the Foundation Machine Key field.

10. Enter the exact path from which this item should be copied in the Foundation Path field.

    All files in the last directory that is specified will be included in the package. Source Machine Key and Source are used together to define the item's location.

Adding a Database Location

Access the Database Component Selection form.



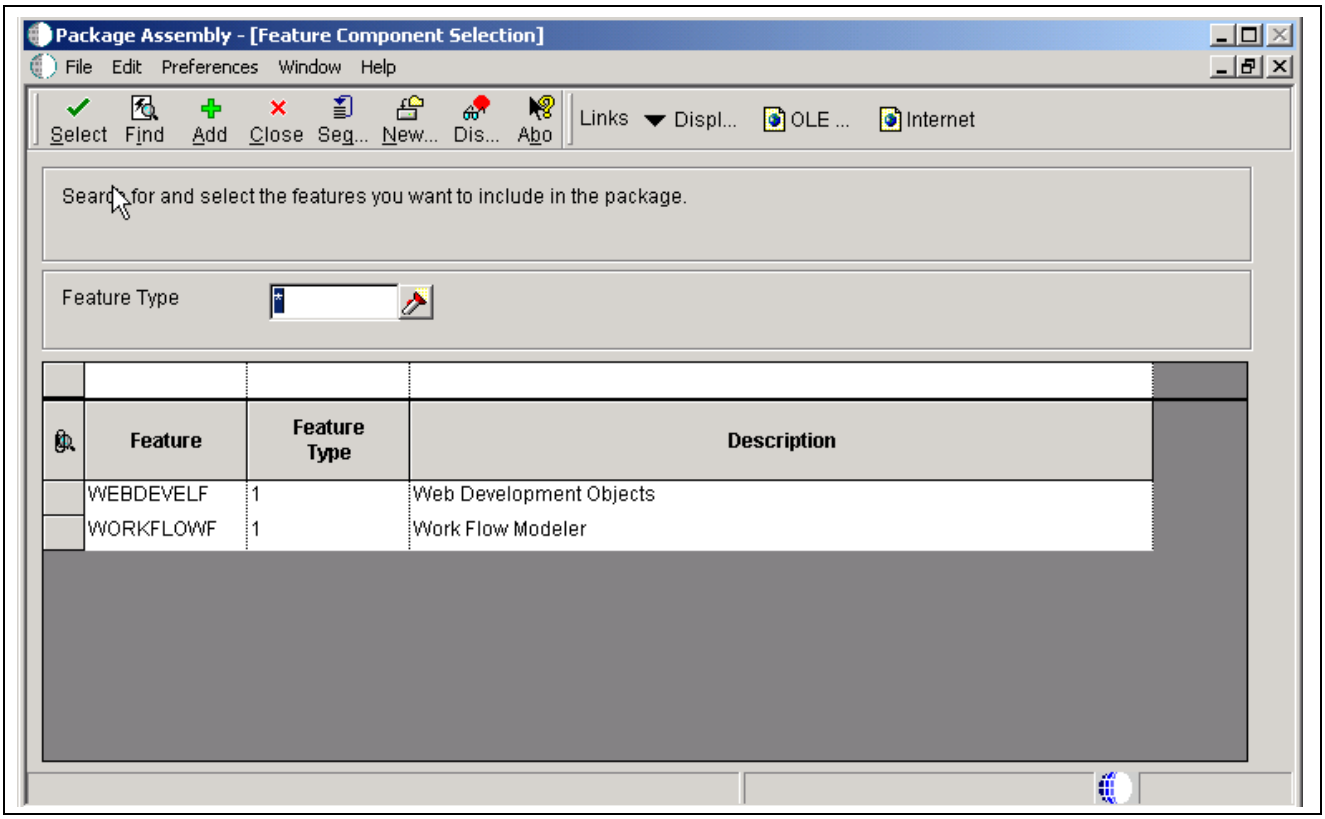
Database Component Selection form

1. Click Add to add a new database component.
2. Enter the name of the database component in the Database Name field.
3. Enter the name of the machine on the network (server or workstation) in the Source Machine Key field.
4. Enter the shared directory for this path code in the Server Share Path field.

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

Adding Features to a Package

Access the Feature Component Selection form. On the Default Object Component form, click Next.



Feature Component Selection form

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

   To select multiple features, press the CTRL or SHIFT key.

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

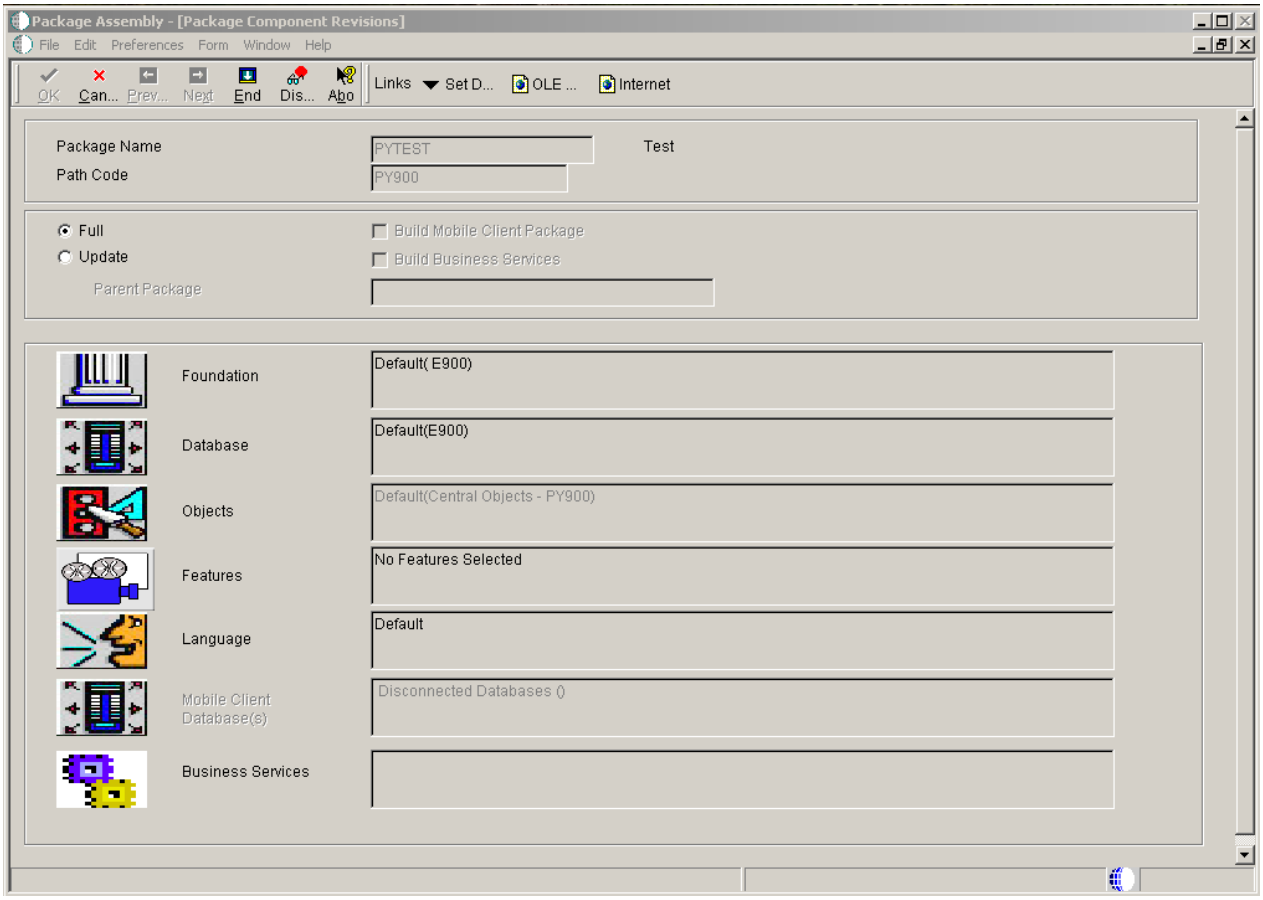
   Oracle's JD Edwards EnterpriseOne Feature Based Deployment Director launches. You can use this director to create the new feature.

   See Chapter 6, "Building Packages," Incorporating Features into Packages, page 94.

3. Repeat steps 1 and 2 until you have finished adding features to your package.
4. When you are finished, click Close to return to the form from which you accessed the Feature Component Selection form.

Reviewing the Package Assembly Selections

Access the Package Component Revisions form. On the Language Component form, click Next.



Package Component Revisions form

1. Review the current foundation, database locations, mobile databases, objects, features, languages, and business services that exist for your package.

   Note. From the Package Component Revisions form, you can select or clear the Build Mobile Client Packages option. This option is automatically disabled if the name of the package that is currently being revised violates the eight-character uniqueness rule or if a mobile package exists with the same name.

2. To change any of the package components, click the button for the component that you want to change.

   The form for that package component appears.

3. When you are finished assembling the package, click End to quit the Package Assembly Director.
4. Continue with the task Activating an Assembled Package.

Assembling a Package Using Express Mode

This section provides an overview of Express mode and lists the forms used to assemble a package using Express mode.

Understanding Express Mode

Express mode enables you to accept default values for the package assembly and then selectively choose which forms to view and modify. This may be preferable if you are familiar with the JD Edwards EnterpriseOne Package Assembly application and do not want to view and click Next through all of the Package Assembly forms.

When you select Express mode, you access the Package Component Revisions form, from which you can access the appropriate forms for the components that you want to update.

The JD Edwards EnterpriseOne Package Assembly application (P9601) is in Express mode by default. This can be changed to Director mode through a processing option.

See Chapter 4, "Assembling Packages," Using Director Mode to Assemble a New Package, page 42.

Forms Used to Assemble a Package Using Express Mode

Form Name	FormID	Navigation	Usage
Package Information	W9601F	Package and Deployment Tools (GH9083), Package Assembly Click Add, and then click Next.	Assemble a new package.
Package Component Revisions	W9601B	Select Express, and click Next on the Package Information form.	Review and revise the components in your package.
Mobile Client Database Revisions	W9601AD	Click the Mobile Client Database(s) button on the Package Component Revisions form.	Specify a mobile package and select the MSDE databases to include in the mobile package.
Foundation Component Selection	W9601H	Click the Foundation button on the Package Component Revisions form.	Add an existing foundation location to your package. Locate the existing foundation location that you want to use in the package, and click Select.
Foundation Item Revisions	W9883D	Click Add on the Foundation Component Selection form.	Add a new foundation location to the package.
Database Component Selection	W9601N	Click the Database button on the Package Component Revisions form.	Add an existing database location to the package. Locate the existing database location that you want to use in the package and click Select.
Database Item Revisions	W9883K	Click Add on the Database Component Selection form.	Add a new database location to the package.
Object Component Selection	W9601D	Click the Objects button on the Package Component Revisions form. Click Browse.	Add an object to the package.
Feature Component Selection	W9601AB	Click the Features button on the Package Component Revisions form. Click Browse.	Add a feature to the package.

Revising an Existing Package

This section provides an overview of the package revision process, lists prerequisites, and discusses how to revise an existing package.

Understanding the Package Revision Process

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

Prerequisite

Verify that the status of the package is *In Definition.* If you try to revise a package that has a status of *Assembly-Definition Complete,* the system displays an error message. To change the status of a package, select Active/Inactive from the Row menu on the Work with Packages form. You cannot revise or delete a package that has already been deployed.

Form Used to Revise an Existing Package

Form Name	FormID	Navigation	Usage
Package Component Revisions	W9601B	Package and Deployment Tools (GH9083), Package Assembly Select the package that you want to revise, and then select Package Revisions from the Row menu.	Revise an existing package.

Revising an Existing Package

Access the Package Component Revisions form.

1. Make any necessary changes to the package components.
2. When you are finished revising the package definition, click OK to return to the Work with Packages form.

   If any build information exists for the package, the system warns you that the changes will delete the existing build information.
3. Click one of these buttons:
   - OK

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

     Delete the revisions and save the existing build information.

Activating an Assembled Package

This section provides an overview of the activation process and lists the forms used to activate an assembled package.

Understanding the Activation Process

After you have assembled a package, the package status remains at Assembly. While you define the package, it is inactive. You must activate the package to change the package status to Assembly-Definition Complete. An assembled package cannot be built until the status has been changed to Assembly-Definition Complete. The Assembly-Definition Complete status indicates that you are finished assembling the package and are ready to begin the build definition process.

You can change the package status at any time until you start the build definition process. That is, even after you have changed a package status to Assembly-Definition Complete, you can change the status back to In Definition if you need to revise the assembled package. When you are ready to define the build for the package, follow the steps described in Defining Package Builds.

Form Used to Activate an Assembled Package

Form Name	FormID	Navigation	Usage
Work with Packages	W9601L	Package and Deployment Tools (GH9083), Package Assembly Select the package that you want to activate, and select Active/Inactive from the Row menu.	Activate the package. You can use this same process to change the status of a complete package back to In Definition.

CHAPTER 5

Understanding the Package Build Process

This chapter discusses:

- How the JD Edwards EnterpriseOne system builds packages.
- Server packages.
- Workstation packages.
- Files created by the build process.
- Features.

How the System Builds Packages

This section discusses:

- How the JD Edwards EnterpriseOne system builds a full client package.
- How the JD Edwards EnterpriseOne system builds an update client package.
- How the JD Edwards EnterpriseOne system builds a full mobile package.
- How the JD Edwards EnterpriseOne system builds an update mobile package.
- How the JD Edwards EnterpriseOne system builds a full server package.
- How the JD Edwards EnterpriseOne system builds an update server package.

How the System Builds a Full Client Package

This is an overview of how the JD Edwards EnterpriseOne system builds a full package. The JD Edwards EnterpriseOne system:

1. Creates the package build directories.
2. Creates the INF file.
3. Copies these directories and files from the check-in location to the package name directory:
   - res
   - source (.c files)
   - include (.h files)
   - work
   - make
   - bin32

- lib32
- obj
- mobileDB
- pack
- spec
- mobilespec
- pkgspec

Note. If you select to build business functions with the package build, the system does not copy bin32, lib32, and object (.obj) files because Oracle's JD Edwards EnterpriseOne BusBuild program creates them.

---

4. Creates an OEE or SSE database with all the central objects tables.
5. Copies all the central objects data into the created database.
6. Runs the JD Edwards EnterpriseOne BusBuild program to compile and link the business functions that create the DLLs in the bin32 directory, the objects in the obj directory, and the libraries in the lib32 directory.
7. Copies the files from the include, source, bin32, lib32, and obj folders of the package to the path code check in location.
8. Compresses the directories.

How the System Builds an Update Client Package

This is an overview of how the JD Edwards EnterpriseOne system builds an update package. The JD Edwards EnterpriseOne system:

1. Creates the package build directories.
2. Creates the INF file.
3. For each object in the Package Build History table (F96225), retrieves the information from the relational database and adds it to the TAM specification files.
4. Copies the associated .c, .h, and .hxx files for the selected objects from the check-in location to the package build area.
5. Runs the JD Edwards EnterpriseOne BusBuild program to update the DLLs in the bin32 directory, the objects in the obj directory, and the libraries in the lib32 directory.
6. Copies the specs and files in the include, source, bin32, lib32, and obj folders of the update to the parent package.

How the System Builds a Full Mobile Package

This is an overview of how the JD Edwards EnterpriseOne system builds a full mobile package. The first five steps are the same as a full client package. The JD Edwards EnterpriseOne system:

1. Creates the package build directories.
2. Creates the INF file.
3. Copies these directories and files from the check-in location to the package name directory:

- res
- source (.c files)
- include (.h files)
- work
- make
- bin32
- lib32
- obj
- mobileDB
- pack
- java
- spec
- mobile spec
- pkg spec

Note. If you select to build business functions with the package build, the system does not copy bin32, lib32, and object (.obj) files because the JD Edwards EnterpriseOne BusBuild program creates them.

---

4. Builds the local spec repository based on the central objects.
5. Runs the JD Edwards EnterpriseOne BusBuild program to compile and link the business functions that create the DLLs in the bin32 directory, the objects in the obj directory, and the libraries in the lib32 directory.
6. Builds the mobile specs.
7. Builds the mobile DB (databases).
8. Builds the mobile.inf file.
9. Compresses the package.

How the System Builds an Update Mobile Package

The process for creating a mobile update package is similar to creating a regular update package. A mobile update package is created automatically based on the parent package. If the selected parent package was marked as *mobile,* a mobile update package is automatically built for every update package.

How the System Builds a Full Server Package

This is an overview of how the JD Edwards EnterpriseOne system builds a full server package. The JD Edwards EnterpriseOne system:

1. Creates the package build directories.
2. Creates the INF file.
3. Copies these directories and files from the check-in location to the package name directory:
   - res

- source (.c files)
- include (.h files)
- work
- make
- bin32
- lib32
- obj
- mobileDB
- pack
- spec
- mobile spec
- pkg spec

Note. If you select to build business functions with the package build, the system does not copy bin32, lib32, and object (.obj) files because the JD Edwards EnterpriseOne BusBuild program creates them.

4. Builds the local spec repository based on the central objects.
5. Generates named event rules (NERs) using the JD Edwards EnterpriseOne BusBuild program.
6. Creates package build directories on the enterprise server.
7. Copies all necessary source and include files from the package location to the enterprise server.
8. Compiles business functions on the server to generate .dll, .so, .sl, or .SRVPGM files.
9. Creates the spec repository in the specified server spec data source.
10. Copies specs from the build machine/deployment server to the spec data source.
11. Compresses the .dll, .so, .sl, and .SRVPGM files.
12. Copies the compressed files from the enterprise server to the deployment server.

How the System Builds an Update Server Package

The steps for a server update package are the same as the first five steps for building a full server package. However, specs for a server update package are merged directly with the live package during package deployment.

Server Packages

This section discusses:

- A description of server packages.
- The server package build process.
- Jde.ini settings for server package builds.

- Spec.ini settings.
- Source code for Sun servers.

A Description of Server Packages

A server package is a group of specification records, source files, header files, and compiled objects that are created on the enterprise servers. A server package is essentially the same as a client workstation package, with these exceptions:

- A server package does not include a Supported Local Database.
- Foundation code is not deployed as part of a server package.
- All specs are built directly into the specified spec data source.

  The specs are copied from the database on the build machine to the spec database data source.
- Some business functions (such as client only business functions) are not built on the server, and therefore are not included in a server package.

All application development takes place on workstations. Object-related files are stored on a single deployment server, and specs are stored in the central objects database on the database server. The application development life cycle is managed by Oracle's JD Edwards EnterpriseOne Object Management Workbench. This configuration enables you to partition business applications to an enterprise server. To ensure that modifications and enhancements that are developed on the workstation are reflected on the server, you must build a server package that contains those modifications and enhancements.

You use Oracle's JD Edwards EnterpriseOne Package Assembly (P9601) and Package Build Director (P9621) applications to assemble, define, build, and deploy server packages. After defining and building a server package, you can deploy it to an enterprise, logic, or application server by using Oracle's JD Edwards EnterpriseOne Deployment Director program (P9631).

The Server Package Build Process

Although creating a server package is identical to creating a client workstation package, you create each for different purposes. Server packages are necessary to transfer specifications and business functions to the enterprise servers.

Oracle's JD Edwards EnterpriseOne Package Build program (P9621) provides these benefits for building server package builds:

- Provides complete integration with client workstation packages.
- Builds a package on multiple servers simultaneously.
- Builds individual package components simultaneously on the server.
- Builds a package on one enterprise server and deploys to another server of the same type.
- Creates history records to enable monitoring and to provide restart capabilities for packages that do not deploy successfully.
- Creates compressed files and loads them onto the deployment server for easier mastering to a CD, or for deploying to another server of the same type.
- Supports both full and update packages.
- Provides the capability to specify a central spec repository for all enterprise and web application (WAS) servers to share.

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

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

1. The system starts a batch application that copies the central objects from the RDBMS to the spec repository on the build machine. The created repository is then copied to the deployment server.
2. On the build machine, the system starts another batch application for the server package.
3. This batch application calls a business function, which in turn calls the server package build engine.
4. The build engine uses the records that the Package Assembly and Package Build Director programs create to:
   - Transfer all business function source and header files to the server.

     At this time, the build engine reads the Object Librarian Master table (F9860) to determine the DLL to which each module belongs. For the JDBTRIG library, a special function is called to direct the trigger library to which the module belongs. In this case, the system does not use the Object Librarian Master table.
   - Start a build master process on the server when the source files for all business functions are transferred.

     This build master starts one or several individual build processes simultaneously. Each DLL has its own build process. The jde.ini file indicates the number of processes that can run simultaneously on a non-iSeries platform. This function is performed by the QBATCH subsystem on an iSeries platform.
   - 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.
   - Create the spec repository in the spec database data source and copy the spec files from the build machine to the spec repository.
   - Check the status of each build piece on each server after the build process has begun on all servers.

     History records are updated as the statuses change.
   - Compress the package components and transfer the compressed files back to the deployment server when the building is complete.

     This happens only if you specify that the system compress the files when it builds the package. This process is repeated for all servers.

Jde.ini Settings for Server Package Builds

If the server package includes business functions, the Build Settings within Server Manager apply to the package. This table describes various build settings:

Setting	Value	Purpose
Build Area	/usr/jdedwards/E900/packages	Indicates the location on the server where the package will be built.

Setting	Value	Purpose
Optimization Flags	+02 (default for HP 9000) -02 (default for RS/6000 and ;Sun)	Varies, depending on the platform. The system uses these compile flags to build business functions in Release mode. You should not change these flags.
DebugFlags	-g -D_DEBUG _DJDEDEBUG (default for HP 9000) -g -qfulpath -qdbextra -D_DEBUG -DJDEDEBUG (default for RS/6000) -g -D_DEBUG -DJDEDEBUG (default for Sun)	Varies, depending on the platform. The system uses these compile flags to build business functions in Debug mode. You should not change these flags.
InliningFlags	blank (default)	Indicates whether the iSeries uses inlining. Enter *Yes* to select inlining on the iSeries. Enter *No* to turn it off. This flag is blank or ignored for non-iSeries servers.
DefineFlags	-DKERNEL -DPRODUCTION_ VERSION -DNATURAL_ ALIGNMENT -D_HPUX-SOURCE (default for HP 9000) -DKERNEL -DPRODUCTION_ VERSION -DNATURAL_ ALIGNMENT (default for RS/6000) -DKERNEL -DPRODUCTION_ VERSION -DNATURAL_ ALIGNMENT -D_SUN-SOURCE (default for Sun)	Indicates the Kernel Production Version of the source for HP, RS, and SUN.
CompilerFlags	-Aa +w1 +z -c (default for HP 9000) -qalign=natural -qflag=I:I -c (default for RS/6000) -qspill=1024 -misalign -KPIC (default for Sun)	Varies, depending on the platform. The spill flag sets the stack space when business functions are compiled. Typically, 1024 is adequate space to compile the delivered business functions.
OSReleaseLevel	+DAportable -q32 (for AIX)	Indicates the release level for which you are compiling the package. You should not change these flags.

Setting	Value	Purpose
LinkFlags	-b -z (default for HP 9000) -bl:/<your system directory>/bin32 /functlist.imp -bM:SRE -bexpall -brtl -lc -bnentry -L. L/usr/<your system directory>/lib -ljdelib -ljdekrnl -ljdenet -bloadmap:loadmap (default for RS/6000) -G -L$(ORACLE_HOME)/lib (default for Sun)	Varies, depending on the platform. The system uses these flags to link business functions. You should not change these flags.
LinkLibraries	blank (default)	Indicates the libraries to which business functions are linked. (Applies to Windows and iSeries servers only.)
SimultaneousBuilds	0 (unlimited) (default) any integer (number of simultaneous builds)	Indicates the number of DLLs that can be built at a time. Zero means that all will be built simultaneously.
Qname	*queue name*	Applies to AS400 only. Specify a queue name if you want the jobs for building dlls to go to a queue other than the default queue.
<*compiler values*>	Any compilers installed on the system will be retrieved from the registry and shown in a pull-down list. (Microsoft Windows platform)	Indicates the compiler level to use for builds. With Tools Release 8.98, the JD Edwards EnterpriseOne application release supports one of two Microsoft Visual C++ compilers. For example, JD Edwards EnterpriseOne 8.12, 8.11_SP1, and 8.11 support both Microsoft Visual C++ 2003 v7.1 and 2005 v8.0 compilers.

Spec.ini Settings

All JD Edwards EnterpriseOne servers and WIN32 clients require a spec.ini file to retrieve the package and spec data source information. This file is created by the Package Build process. It resides in the spec directory.

This table describes the settings within the SPEC LOCATIONS section of the spec.ini file:

Setting	Value
*path code*_Package	Indicates the name of the package.
*path code*_DataSource	Indicates the database data source that is used to retrieve the spec information.

Source Code for Sun Servers

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

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

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

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

See Also

*JD Edwards EnterpriseOne Tools Release 8.98 Server Manager Guide*on the Update Center.

Workstation Packages

This section discusses:

- Workstation installation.
- Building specifications and business functions.
- Defining the compiler level.
- Verifying UNICODE settings.
- Package INF files.

Workstation Installation

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 workstation configuration contains the full suite of applications, including those that the user rarely or never uses, but all applications are available immediately.

Building Specifications and Business Functions

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

```
WaitForBusbuild=Y/N
```

A *Y* means that business functions are built after all the specs are complete. The system builds the specifications and business functions sequentially instead of simultaneously.

An *N* means that specs and business functions are built simultaneously, which can speed up the build process.

Defining the Compiler Level

If you have more than one version of Microsoft Visual C++ on your machine, add one of the following settings to the [JDE_CG] section of the workstation jde.ini file on the computer that you use to build the packages:

```
VisualStudioVersion=6
VisualStudioVersion=7
VisualStudioVersion=8
```

Use the first setting if you are using the Microsoft Visual C++ v6.0 compiler, the second setting if you are using the Microsoft Visual C++ 2003 v7.1 compiler, and the third setting if you are using the Microsoft Visual C++ 2005 v8.0 compiler. If more than one supported compilers have been installed on the computer that you use to build packages and the VisualStudioVersion is not defined in the jde.ini, the highest level compiler will be used to perform the build.

Verifying UNICODE Settings

If you are upgrading from Xe and you have modified any interactive or batch applications that contain NERs and that are language-enabled, you must ensure that the following setting is in the [INSTALL] section of the workstation jde.ini on the computer that you use to build the packages:

```
Unicode Conversion Codepage=<code_page_value>
```

In this setting, *code_page_value* is a valid value for the code page of the language-enabled application that contains NERs. For example, for Korean language the value would be:

```
Unicode Conversion Codepage=ksc-5601
```

Note. If you are a language customer but have never added NERs to your applications, then you are not required to have this setting. Also note that `LocalCodeSet=<value>` is no longer used in releases subsequent to Xe.

Package INF Files

The Package INF file is essentially the interface between the package build and Oracle's JD Edwards EnterpriseOne Workstation Installation program. The INF file defines the components that are included in the package, the source and destination paths for those components, and the attributes that are needed to install the package.

The INF file is created during the package build process and is stored in its own package_inf directory, based on the release master directory. JD Edwards EnterpriseOne Workstation Installation reads the INF file for the package that it is installing to determine which components are loaded to a workstation, as well as their locations.

Here is a typical INF file for package DV900FA, which is full package A for the DV900 path code. This INF file includes these sections:

- [SrcDirs]
- [DestDirs]
- [FileSets]
- [FileSetsDescription]
- [Components]
- [Typical]

- [Compact]
- [Attributes]
- [Oracle Databases]
- [Start]
- [Desktop]
- [Environment]
- [Fonts]
- [Feature]

[SrcDirs]

The JD Edwards EnterpriseOne Workstation Installation program uses these settings to determine the source path from which information is copied. JD Edwards EnterpriseOne Package Build compresses these directories. JD Edwards EnterpriseOne Workstation Installation copies the compressed directories to workstations.

Item	Purpose
SPathcode=\\MachineName\E900\PACKAGE\ DV900FA	Indicates the location of the package that the package builds for workstation installation. The default value for this path is the path code directory over which you built the package. You can change this setting if you want to use a different package.
SSYS=\\MachineName\E900\SYSTEM	Indicates the location of the system directory that the package builds for workstation installation. The default value for this path is the system directory that is associated with the path code over which you built the package. Normally, this directory is subordinate to the release share name (E900). This item appears only when included in the package.
SPathcodeDATA=\\MachineName\E900\ DV900\PACKAGE\DATA	Indicates the location of the Supported Local Database that the package builds for workstation installation. The default value for this path is the data directory that is subordinate to the path code over which you built the package. This item appears only when included in the package.

[DestDirs]

The JD Edwards EnterpriseOne Workstation Installation program uses these settings to determine the destination paths on the workstation. The process replaces %INSTALL with the user's computer configuration, which is set up in the User Display Preferences table (F00921) and the User Defined Codes Language Status table (F00051).

Item	Purpose
DPathcode=%INSTALL\path code	Indicates the destination directory for the package.

Item	Purpose
DSYS=%INSTALL\system	Indicates the destination directory for system files. This item appears only when included in the package.
DPathcodeDATA=%INSTALL\path code\data	Indicates the destination directory for the database. This item appears only when included in the package.

[Filesets]

These settings list the various source and destination directories that are subordinate to the path code for the package. Y equals compressed, and N equals not compressed. The source and destination directory names are preceded by an *S* and *D,* respectively.

Item	Purpose
*Pathcode*1=Y, $S*pathcode*\bin32, $D*pathcode*\bin32	Indicates the bin32 directory that is subordinate to the path code.
*Pathcode*2=Y, $S*pathcode*\spec, $D*pathcode*\spec or *Pathcode*2=Y,$S*pathcode*\mobilespec, $D*pathcode*\mobilespec	Indicates the spec directory that is subordinate to the path code. Indicates the mobile spec directory that is subordinate to the path code.
*Pathcode*3=Y, $S*pathcode*\include, $D*pathcode*\include	Indicates the include directory that is subordinate to the path code.
*Pathcode*4=Y, $S*pathcode*\lib32, $D*pathcode*\lib32	Indicates the lib directory that is subordinate to the path code.
*Pathcode*5=Y, $S*pathcode*\obj, $D*pathcode*\obj	Indicates the obj directory that is subordinate to the path code.
*Pathcode*6=Y, $S*pathcode*\source, $D*pathcode*\source	Indicates the source directory that is subordinate to the path code.
*Pathcode*7=Y, $S*pathcode*\work, $D*pathcode*\work	Indicates the work directory that is subordinate to the path code.
*Pathcode*8=Y, $S*pathcode*\make, $D*pathcode*\make	Indicates the make directory that is subordinate to the path code.
*Pathcode*9=Y, $S*pathcode*\res, $D*pathcode*\res	Indicates the res directory that is subordinate to the path code.
*Pathcode*10=Y,$S*pathcode*\sbf.cab, $D*pathcode*\java	Indicates the java directory that is subordinate to the path code.
SYS=Y,$SSYS\System.cab, $DSYS	Indicates the compressed system database that the package build generates.
*Pathcode*DATA=Y, $S*pathcode*DATA\data.CAB, $D*pathcode*DATA	Indicates the compressed data database that the package build generates.

[FileSetsDescription]

This section provides a text description for each fileset as shown in this example:

```
[FileSetsDescription]
DV9001=Business Function DLL Files
DV9002=Specification Files
DV9003=Include Files
DV9004=Library Files
DV9005=Object Files
DV9006=Source Files
DV9007=Work Files
DV9008=Make Files
DV9009=Resource Files
DV90010=SBF Source Files
SYS=Foundation Files
DV900DATA=Data Files
```

[Components]

These settings indicate the location of the foundation, production, and development objects, as well as the database files.

Item	Purpose
Production Objects=APPL_PKG1, APPL_PKG2, APPL_PKG3	Indicates the location of production objects.
Development Objects=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.

[Typical]

This section describes the setting for a typical development user.

Item	Purpose
Name=Development	Indicates that the package is for a development user.
Description=Install the development objects	Indicates the package description.
Components=ProdObj, DevObj, Foundation, Data	Indicates that the package should contain both production and development objects, as well as the database and foundation.

[Compact]

This section describes settings for a typical production user.

Item	Purpose
Name=Production	Indicates that the package is for a production user.
Description=Install the production objects only	Indicates the package description.
Components=Production Objects, Foundation, Data	Indicates that the package should contain only production objects, as well as the database and foundation.

[Attributes]

This section contains information about the current release, global tables, specification and help files, and the jde.ini file.

Item	Purpose
PackageName=DV900FA	Indicates the name of the package.
PathCode=DV900	Indicates the path code for which the package is being built.
Built=Build Completed Successfully	Indicates the package status. A status of 50 or 70 means that the package is ready for installation.
PackageType=Full	Indicates the package type: full or update.
SPEC_FORMAT=XML	Indicates the format for the specifications.
Release=E900	Indicates the current release, which determines the 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.
BaseRelease=B9	Indicates the current base release.
SystemBuildType=RELEASE	Indicates the type of build: DEBUG or RELEASE. This option is retrieved from owver.dll.
MFCVersion=6	Indicates the version of the MFC compiler.
SpecFilesAvailable=Y	Indicates that specification files are available to merge or copy. This option is always set to Y for full packages. For update packages, this option is set to Y only if objects are included in the package.
DelGlbTbl=Y	Indicates whether to delete global tables when installing. This option is set to Y for full packages. For update packages, this option is set to Y only if the objects include a table object.
ReplaceIni=Y	Indicates whether to delete the existing jde.ini file when installing, and then create a new one. This option is set to Y for full packages. For update packages, the user must specify during package build definition whether to replace the jde.ini file.

Item	Purpose
AppBuildDate=Mon Jul 20 11:22:22 2008	Indicates the date and time when the package was built. Full packages always have this date. This option is blank when no objects are included in the package.
FoundationBuildDate=Wed Jun 03 15:08:34 2008	Indicates the date and time when the foundation was built.The date is retrieved from owver.dll. Full packages always have this date. This option is blank when no foundation location is specified in the package.
DataBuildDate=Wed Jun 03 15:08:34 2008	Indicates the date and time when the database file was built. Full packages always have this date. This option is blank when no database location is specified in the package.
DeploymentServerName=DENMLSAN222	Indicates the name of the deployment server.
Location=DENVER	Indicates the location of the deployment server.
DeploymentStatus=Approved	Indicates the package deployment status.
PackageDescription=Development full package A	Indicates the package description.
Icon Description=JDEdwards	Describes the desktop icon.
Default Environment=DV900	Indicates the default environment.
AllPathCodes=Y	Indicates that a package for *ALL path codes exists when set to Y.
Mobile=	Indicates whether the INF file is for a mobile package . This option is set to Y when the INF is for a mobile package.
Spec=	Indicates the format of the specifications with these values: • XML_RDBMS Use this setting when XML specs are in an RDBMS (full package). • XML_TAM Use this setting when XML specs are in TAM format (update package).

[Oracle Databases]

This section contains information about the Oracle databases.

Item	Purpose
JDELocal_DV900=ORACLE	Indicates the local Oracle database.
SPEC_DV900FA=ORACLE	Indicates the Oracle database.

Each of the databases listed, has its own section in the INF file as shown in this example:

```
[JDELocal_DV900]
SourceTableSpace=JDELocal
Server=127.0.0.1
UserID=SYSTEM
DataFileDestDir=$DDV900DATA\JDELocal_DV900.dbf
DumpFileDestDir=$DDV900DATA\JDELocal_DV900.dmp

[SPEC_DV900FA]
SourceTableSpace=MASTER
Server=127.0.0.1
DataFileDestDir=$DDV900\Spec\SPEC_DV900FA.dbf
DumpFileDestDir=$DDV900\Spec\SPEC_DV900FA.dmp
```

[START]

This section contains startup information:

```
[START]
ProgramGroupName=JDEdwards
Item1=System\Bin32\activConsole.exe, JDEdwards Solution Explorer, appl_pgf\res\One⇒
World.ico
```

[Desktop]

This section contains desktop information:

```
[Desktop]
Item1=System\Bin32\activConsole.exe, JDEdwards Solution Explorer, appl_pgf, res⇒⇒
\OneWorld.ico
```

[Environment]

This section contains environment information:

```
PathDV900=%INSTALL\DV900\bin32;
PathSys=%INSTALL\system\bin32;
```

[Fonts]

This section contains font information:

```
[Fonts]
Arial=Font\arial.ttf
```

[Feature]

This section describes information for any features that are included in the package. A feature is a set of files or configuration options that is included in a package for deployment to a workstation or server. This example provides an example of some features that might be included:

```
[Feature]
WEBDEVELF=\\DENMLSAN246\E900\Package_inf\Feature_inf\WEBDEVELF_1.INF
WEBDEVOC4J=\\DENMLSAN246\E900\Package_inf\Feature_inf\WEBDEVOC4J_1.INF
WORKFLOWF=\\DENMLSAN246\E900\Package_inf\Feature_inf\WORKFLOWF_1.INF
```


Files Created by the Build Process

This section discusses:

- Workstation package build
- Server package build
- UNIX server build
- Windows server build
- iSeries server build

Workstation Package Build

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

The production environment PD900/bin32 directory contains the DLLs that are created on the workstation. All of the business function source files are in the PD900/source directory.

Files Created by a Business Function Build

When you build a single business function through Oracle's JD Edwards EnterpriseOne Object Librarian, the Business Function Builder program uses the make (*.mak) file that is generated at runtime, and creates or copies these files and builds the business functions into their respective DLLs:

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

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

These files are created for NER business functions:

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

These files are created for table event rules:

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

Server Package Build

Server package builds are used to move path code objects from the deployment server to enterprise server platforms. Server package builds are initiated when you create either full or update packages during package assembly. After you have assembled the package, you must select the server option during package definition, and select the relevant servers from the list of available servers in the screen that follows. When package definition is complete and the package has been activated, highlight the package and select Submit Build from the Row menu to start the server package build. When the server package build has finished successfully, you can deploy the server package.

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

UNIX Server Build

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

Files Created by a Business Function Build

When you are building business functions, these groups of source files are actually compiled:

- NER business functions.
- Table event rules.
- C business function event rules.

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

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

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

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

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

Where Business Functions Are Stored

On a UNIX platform, related business functions are grouped into shared libraries. This grouping limits the size and number of procedures that are contained in each shared library. Grouping prevents memory allocation errors and avoids platform-specific limitations in the number of procedures that you can export per shared library.

The exact location of the package is determined by the Build Settings within Server Manager.

Subordinate to the package directory (PD900FA) 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 this example where the top directory represents the package name:

```
PD900FA
   source
     CAEC
     CALLBSFN
     CCORE
     CDESIGN
     CDIST
     CFIN
     CHRM
     CMFG
     JDBTRIG
```

Each subdirectory contains the business function source files that belong to the shared library. All shared libraries are installed in the PD900/bin32 directory. The naming convention for the shared libraries is lib, followed by the name of the shared library subdirectory, followed by .sl (for HPUX) or .so (for AIX). An example is libccore.sl.

Specification Files

JD Edwards EnterpriseOne specifications are stored in an RDBMS. The database data source for this database is specified in the spec.ini file or is selected by the package build administrator during the server package build definition process. Package Build copies the specs directly from the build machine to the specified spec database. However, local cache (GLBLTBL, DDDICT, and DDTEXT) specification files are still created in TAM format. The contents in these files are destroyed when a new package is deployed.

Windows Server Build

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

Files Created by a Business Function Build

When you are building business functions, these groups of source files are actually compiled:

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

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

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

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

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

- DLLs (.dll)

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

Where Business Functions Are Stored

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

The exact location of the package is determined by the Build Settings within Server Manager.

Subordinate to the package directory (PD900FA) is a source directory. This source directory contains subdirectories for each DLL that is created on the enterprise server.

The directory structure looks like this example where the top directory is the package name:

```
PD900

   source

     CAEC

     CALLBSFN

     CCORE

     CDESIGN

     CDIST

     CFIN

     CHRM

     CMFG

     JDBTRIG
```

Each subdirectory contains all of the business function source files that belong to the DLL. All DLLs are installed in the PD900\bin32 directory. They have the same name as the DLL subdirectories, except that they have the .dll suffix.

Specification Files

JD Edwards EnterpriseOne specifications are stored in an RDBMS. The database data source for this database is specified in the spec.ini file or is selected by the package build administrator during the server package build definition process. Package Build copies the specs directly from the build machine to the specified spec database. However, local cache (GLBLTBL, DDDICT, and DDTEXT) specification files are still created in TAM format. The contents in these files are destroyed when a new package is deployed.

iSeries Server Build

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

Files Created by a Business Function Build

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

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

Where Business Function Source Members Are Stored

iSeries business function source and headers are now transferred to the Integrated File System (IFS). Server package build transfers objects to these subdirectories under the server package directory in the IFS for the iSeries.

The exact location of the package is determined by the Build Settings within Server Manager.

Subordinate to the package directory (PD900FA) is a source directory. This source directory contains subdirectories for each DLL that is created on the enterprise server.

The directory structure looks like this example where the top directory is the package name:

```
PD900FA
   include
   pack
   source
     CAEC
     CALLBSFN
     CCORE
     CDESIGN
     CDIST
     CFIN
     CHRM
     CMFG
     JDBTRIG
   spec
   text
```

This table describes the files that are found in the directories:

Directory	Description
PD900\include	This is the location where .h and .hxx source files are located. These objects are taken from the server and built on.
PD900\pack	This folder is no longer used but is created for backwards compatibility.
PD900\source	This directory contains subdirectories that include the business function DLL names. Each subdirectory contains .c source for the business functions that are compiled and linked into the DLL.
PD900\spec	This folder is no longer used but is created for backwards compatibility.
PD900\text	This directory contains build text, status files and log files (.txt, .sts, .log) for business function DLLs and specification files. The text files contain information that is needed for the server package build. The text files also contain build directives for creating business function DLLs. The status files for specification files indicate whether a server package build was successful in converting pack files into spec files. The status files for business function DLLs indicate which .c source files were successfully compiled and linked. The log files created exclusively for business function DLLs contain the compiler commands used to build and link business functions. For the Microsoft Windows platform, the beginning of this file identifies the compiler used to perform the build (for example, "Using Microsoft Visual Studio Version 8").

Note. After an upgrade, existing iSeries server path codes must be rebuilt with the server package build to avoid problems building server package updates and manually re-linking business functions using the LINKBSFN program.

Specification Files

JD Edwards EnterpriseOne specifications are stored in an RDBMS. The database data source for this database is specified in the spec.ini file or is selected by the package build administrator during the server package build definition process. Package Build copies the specs directly from the build machine to the specified spec database. However, local cache (GLBLTBL, DDDICT, and DDTEXT) specification files are still created in TAM format. The contents in these files are destroyed when a new package is deployed.

Features

This section discusses:

- Defining features
- Feature INF files

Defining Features

In addition to objects, you can also add a *feature* to a package. A feature is a set of files or configuration options that must be copied to a workstation or server to support an application or another function. Like objects, features are included in a package and deployed to the workstations and servers that require the feature components.

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

You define a feature by using the JD Edwards EnterpriseOne Package Assembly program (P9601). You can then add the feature to a package by using the JD Edwards EnterpriseOne Package Assembly program (P9601) and Package Build Director program (P9621).

Feature INF Files

When a package contains features, a section called [Features] in the Package INF file includes both the feature name and a pointer to the Feature INF file that is created for each feature in the package. These Feature INF files provide specifications that tell the installation program the actions to perform during the installation.

The Feature INF file can include these sections:

- [Header]
- [Registry]
- [INI]
- [FileSets]
- [Shortcut]
- [ThirdPartyApps]
- [ODBCDataSources]

This is a typical Feature INF file for which the sections contain specifications for each feature component.

[Header]

The header section contains general information about the feature and specifies the installation options for the feature.

Item	Purpose
Feature=	Name of the feature.
FeatureType=	Type of feature.
Description=	Text description of the feature.
Required=	A setting that indicates whether installation of the feature is required.
InitialChoice=	A setting that specifies the default selections for features that the user can install.

The Required and InitialChoice entries are set using the three Feature Installation option settings (Required, Selected, Deselected) on the Feature Information form. When you select one of these three options, the system writes these values into the Required and InitialChoice entries in the feature INF file.

Feature Installation Option	Required	InitialChoice
Required	Y	Both
Selected	N	Both
Deselected	N	Custom

[Registry]

This section contains information about how the feature affects the Windows registry.

The settings for this section are displayed in this order:

Registry_no.=Root[value], Key, [prefix]Name, [prefix]Value

The following table contains a description of each variable:

Item	Purpose
Root	Describes the root in the registry with these values: • 0 means root • 1 means current user • 2 means local machine location • 3 means users
Key	Indicates the key for the registry value.
Name	The registry value name. Name prefixes are: • + means that the name is created (if it does not already exist) when the feature is installed • – means that the name is deleted with all subkeys when the feature is uninstalled • * means that the name is created (if it does not already exist) when the feature is installed, and it is removed with all subkeys when the feature is uninstalled
Value	The name of the registry value. Value prefixes are: • #x means that the value is stored as a hexadecimal value • #% means that the value is stored as an expandable string • # means that the value is stored as an integer • #$ means that the value is stored as a string

[INI]

This section contains information about how the feature affects the jde.ini file.

The settings for this section are displayed in this order:

Ini_no.=FileName, Directory, Section, Key, Value, Action

The following table contains a description of each variable:

Item	Purpose
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.

The settings for this section are displayed in this order:

Fileset_no.=Compression, SourceDirectory, FileName, TargetDirectory

The following table contains a description of each variable:

Item	Purpose
Compression	An option that indicates whether the fileset is compressed.
Source Directory	The source location of the fileset.
FileName	The name of the CAB file for the fileset.
Target Directory	The target location into which the fileset will be placed.

[Shortcut]

This section contains information about shortcuts that appear on the Windows desktop as part of the feature installation.

The settings for this section are displayed in this order:

Shortcut_no.=Directory, Name, Target, Arguments, Description, HotKey, Icon, IconIndex, ShowCmd, WKDir

The following table contains a description of each variable:

Item	Purpose
Directory	The directory where the shortcut is created.
Name	The name of the link file for the shortcut.
Target	The name of the executable file for the shortcut.
Arguments	Any command line arguments for the shortcut.
Description	A description of the shortcut.
HotKey	A hot key that launches the shortcut.
Icon	The shortcut icon and location.
IconIndex	An index of the icon if the icon is inside an image list.
ShowCmd	A command for the application window, with these value options: • 0 means show the window normal-sized. • 3 means show the window maximized. • 7 means show the window minimized; not active.
WkDir	The working directory for the shortcut.

[ThirdPartyApps]

This section contains information about third-party products that are installed with the feature.

The settings for this section are displayed in this order:

ThirdPartyApp_no.=Source Directory, Description, Synchronous/Asynchronous, Execute Before/After, FileName

The following table contains a description of each variable:

Item	Purpose
Source Directory	Source location of the executable for running the third-party application.
Description	Description of the third-party application.
Synchronous/Asynchronous	An option that indicates whether the third-party application can be installed in parallel (synchronous) or must be installed serially (asynchronous).

Item	Purpose
Execute Before/After	An option that indicates whether the third-party application installation is run before or after JD Edwards EnterpriseOne is installed.
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 two sections in the feature.inf. One section contains header information and the other contains the detail information. The feature.inf contains one header section listing all data source components that are included in the feature. For each data source that is listed in the header, a corresponding detail section exists. Only the header section is described in this table. For information about the detail section, see the documentation for the selected ODBC Driver.

The settings for this section are displayed in this order: *DataSourceName=DataSourceDriver*

Item	Purpose
DataSource Name	The name of the ODBC data source.
DataSource Driver	The driver that is used for the data source.

CHAPTER 6

Building Packages

This chapter provides an overview of the Package Build process and discusses how to:

- Build a package.
- Incorporate features into packages.
- View package build records and resubmit builds.

Understanding the Package Build Process

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

This section discusses:

- Directory structure for packages.
- Package build tasks.
- JD Edwards EnterpriseOne Package Build Definition Director.
- Business function builds during package build.
- Package compression.
- Verification of a package build.

Directory Structure for Packages

When a package is built, a directory structure with the name of the package is created within the appropriate path code directory. This directory contains the package information.

Example: JD Edwards EnterpriseOne E900 Directory Structure

The E900 (release name) directory structure looks similar to this example.

This is an example of the directory structure for PD900 (path code name):

```
Package
   PackageA (package name)
    bin32
    include
```

```
    java
   lib32
    make
    obj
    res
    source
    spec
    work
    mobilespec
    mobiledb
    pkgspec
bin32
include
java
lib32
make
obj
res
source
work
```

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

Package Build Tasks

The process that you perform to build a package might take several hours. For this reason, it is recommended that you initiate the actual package build at the end of the working day, if possible. Complete these tasks when you build a package:

- Transfer objects.

  Ensure that all of the objects that you want to include in the build have been transferred to the appropriate path code.
- Ensure that the database for the package has the most current replicated data.
- Build a package.

  Build a package using the path code to which objects were transferred.
- (Optional) Perform a cross-reference build.

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

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

Package Build Definition Director

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

Like Oracle's JD Edwards EnterpriseOne Package Assembly application, Oracle's JD Edwards EnterpriseOne Package Build application defaults to Express mode. You can use a processing option to switch the JD Edwards EnterpriseOne Package Build application to Director mode.

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

Form	Description
Package Build Definition Director	Use this form to review introductory information about the Package Build Definition Director.
Package Selection	Use this form to select the defined package that you want to build. The status of the package must be *Assembly-Definition Complete.*
Package Build Location	Use this form to specify whether you want to build the package for the client workstation, one or more servers, or both clients and servers. For server packages, also specify the shared specifications data source.
Server Selection	Use this form to specify the server location. (The server location is required when you build a package for a server.)
Build Specification Options	Use this form to specify whether you want to include all specification tables or only selected tables in the package. The option to build individual specifications is useful if a build fails and the package error log indicates that an individual specification file needs to be rebuilt.
Individual Specification Selection	Use this form to include only selected specifications in the package.
Business Function Options	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.

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

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

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

Viewing Package Build History and Resubmitting Builds

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

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

See Also

Chapter 6, "Building Packages," Viewing the Package Build History, page 119

Business Function Builds During Package Build

When you build business functions as part of the package build, the system performs the same process as if you had manually run Oracle's JD Edwards EnterpriseOne BusBuild program (selected Build from the Global Build menu) after you built the package.

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

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

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

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

Package Compression

You can compress a server package or client package.

Compressing Server Packages

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

```
DoCompression=1
```

This setting compresses packages that you built on the server and deploys them to other servers of the same type, such as all AIX UNIX servers or NT servers. If you plan to deploy a package to an enterprise server, you must build the package on the same type of server, with compression selected. The directories are compressed into file types that are compatible with the type of enterprise server that you are using. For NT servers, the file extension is .cab, for UNIX the file extension is .z, and for AS400, the file has no extension.

When the server builds a compressed package, it stores the compressed files in subdirectories, such as \bin32, that are subordinate to this 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 program also creates .cab files in the *package name* directory, although these files are not used when you deploy to a client workstation.

The compression process creates a new file called compressed.inf in the *server type* directory. This file includes the information that the system needs to deploy the compressed files. This table shows the type of compressed file that the process creates for each type of server:

NT	UNIX	iSeries
.cab	.z	SAVF

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

Specs are no longer compressed. They are deployed either by being copied directly from the database on the deployment server or by setting the spec.ini file setting.

Compressing Server Update Packages

To compress an update package for a server and deploy it to other servers of the same type, enter *1* for processing option number 2 on the JD Edwards EnterpriseOne Package Build Director application (P9621).

This option displays a field to compress an update package when defining that package. When you select this option and build the package, the program creates a compressed file in the *package name**server type*\bin32 directory. Specifications are not compressed. They are copied directly from the deployment server to the specification database data source during package deployment.

Compressing Client Packages

When you compress directories, the application objects that are included in the package are automatically compressed. The system also creates an entry in the package INF file that indicates whether the foundation, data, and application objects are compressed.

Verification of a Package Build

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

During the verification process, the program verifies that:

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

Note. You cannot verify the specification database data source. Only enterprise servers and clients can be verified.

Building a Package

This section lists prerequisites and discusses how to:

- Set processing options for the Package Build Definition Director.
- Define a package build.
- Review package build selections.
- Build a package.

Prerequisites

Before you complete the tasks in this section:

- Verify that the User ID that is used to perform the package build has drop table and create table rights.

If you are using a Oracle, SQL, or DB2 UDB for Microsoft Windows/Unix, and you are running with security server turned on, you must add a security override so that the package build process can create the metadata repository table in central objects.

See Appendix A, "Adding a Security Override for Package Build," page 189.

- For customers adopting Microsoft Visual C++ 2005:

  All JD Edwards EnterpriseOne Windows machines receiving application foundation packages built with Microsoft Visual C++ 2005 require the runtime libraries to be installed.

  See Appendix B, "Using the Microsoft Visual C++ 2005 Compiler," page 191.

- Assemble the package and verify that the status of the assembled package is *Assembly-Definition Complete.*
- Verify that Oracle's JD Edwards EnterpriseOne Object Configuration Manager (OCM) mappings are correctly set for the JD Edwards EnterpriseOne Package Build (R9621) and Server Package Build (R9622) programs, which the system generates as part of the package build process.

  For example, if you want the programs to run locally, ensure that the OCM mappings point to Local for the environment in which the package build is running.

- Verify that logging is turned off during the package build.

  When the jdeproperties.log file is set for logging in the \system\classes folder and a package build is submitted, the build process is slowed down.

See Also

JD Edwards EnterpriseOne Tools 8.98 Configurable Network Computing Implementation Guide, "Working with Object Configuration Manager"

Forms Used to Build a Package

Form Name	FormID	Navigation	Usage
Package Selection	W9621C	Package and Deployment Tools (GH9083), Package Build. Click Add, and then click Next.	Select a package to define a build.
Package Build Revisions	W9621B	Package and Deployment Tools menu (GH9083), Package Build. Select a package, and deactivate it by selecting Active/Inactive from the Row menu. Select Build Revisions from the Row menu.	Review and change package build options.
Work With Package Build Definition	W9621L	Package and Deployment Tools (GH9083), Package Build	Add a package build definition. Build a defined package.

Setting Processing Options for the Package Build Definition Director (P9621)

Processing options enable you to specify the default processing for programs and reports.

For programs, you can specify options such as the default values for specific transactions, whether fields appear on a form, and the version of the program that you want to run.

For reports, processing options enable you to specify the information that appears on reports. For example, you set a processing option to include the fiscal year or the number of aging days on a report.

Do not modify JD Edwards EnterpriseOne demo versions, which are identified by ZJDE or XJDE prefixes. Copy these versions or create new versions to change any values, including the version number, version title, prompting options, security, and processing options.

Processing Tab

Although processing options are set up during JD Edwards EnterpriseOne implementation, you can change processing options each time you run a program.

1. Changes

Enter a value to determine how changes will occur.

<*Blank*>means that changes will only be allowed at the package level and will apply to all servers selected.

Enter *1* to enable changes to the build definitions by individual server.

2. Mastering

Mark this processing option with a *1* if this process is for Mastering purposes. If the process is for all users, mark this processing option with <*Blank*>.

3. Build Verification

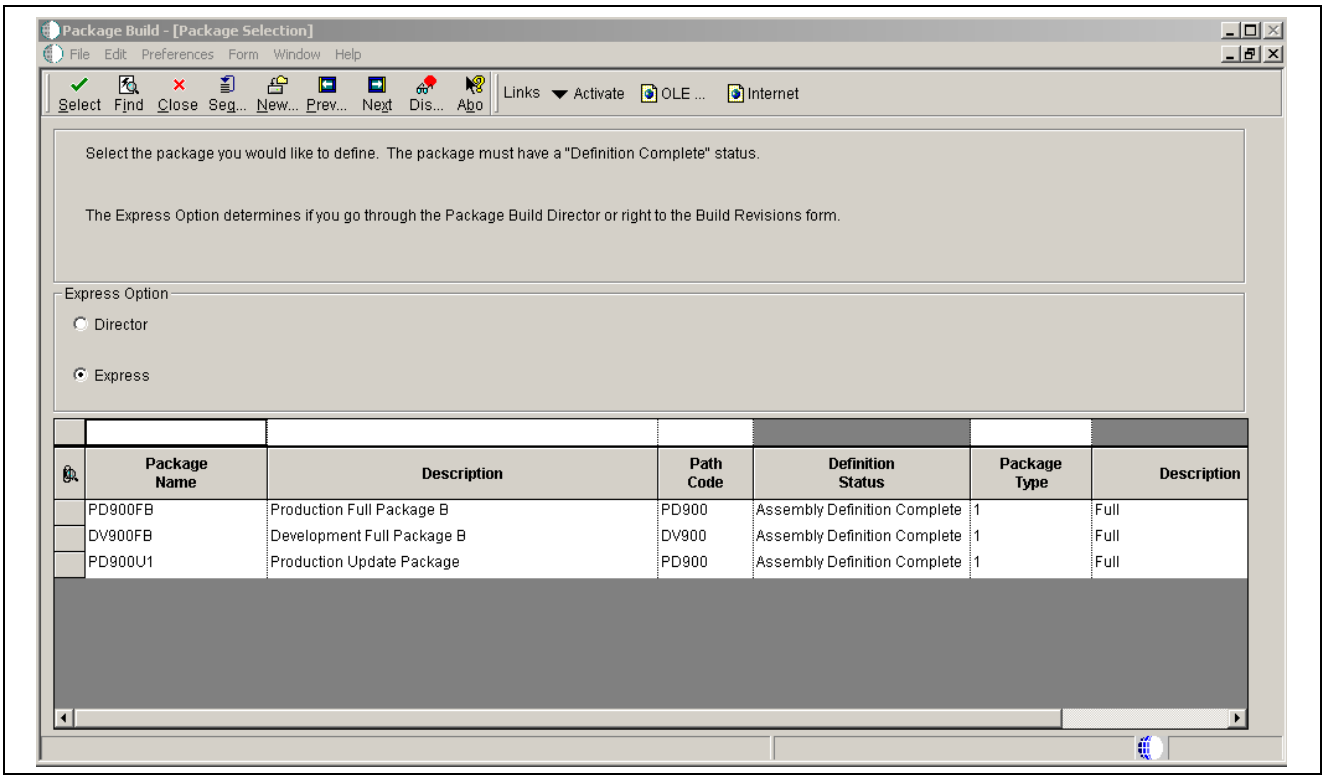
Mark this processing option with a *1* if the Build Verification UBE is to be run prior to building all packages. If the build verification fails, the package build UBE will not be run. Leave this processing option <*Blank*> if you do not want to run Build Verification.

4. Director or Express Mode

Use this processing option to switch between Director and Express modes.

Defining a Package Build

Access the Package Selection form.



Package Selection form

1. Find and select the defined package that you want to build.
2. If the package definition has a status of *In Definition,* you must change the status to *Assembly-Definition Complete* before you build the package. To change the status, select the package and select Activate from the Row menu.
3. On the Package Selection form, in the Express Option pane, select one of these options:

Option	Description
Director	Select this option if you want to configure the package build. Director enables you to navigate the package build definition forms.
Express	Select this option if you want to accept the default build parameters. Express enables you to accept the default options for the package build and skip the package build definition forms.

4. If you selected the Express option, skip to the Reviewing Package Selections task. If you selected the Director option, continue with the next task.
5. On the Package Build Location form, select one or both of these options:

Option	Description
Client	Select to indicate that the package is being built for installation on client workstations. If the package includes a build for a mobile client, the Client option is automatically selected.
Server(s)	Select to indicate that the package is being built for installation on one or more servers.

6. If you are building a package for client workstations only, click Next and proceed to step 10.
7. If you are building a server package, you can specify the Shared Location for the shared spec database and click Next.

   Note. The default shared spec database is always the central objects data source for the package path code.

8. To select a server on the Server Selection form, and then double-click the row header for the server.

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

   Note. Servers are automatically selected for an update package. They are selected based on the server selection of the parent package.

9. Click Next.
10. On the Build Specification Options form, 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.
11. Complete these fields and click Next:

Field	Description
All Specification Tables	Select this option if you want to build all specification tables into the package.
Individual Specification Tables	Select this option if you would like to select individual tables to include in the package. All of the tables listed on the Individual Specifications Selection form will be included in the package.
Stop Build Option	Indicate the point at which the system should stop the build. You can continue building on all errors, stop building on specification errors, stop building on business function errors, or avoid compressing when errors exist.
Replace jde.ini	For update packages, indicate if you want a new jde.ini file delivered with the package. Leave this unchecked unless the jde.ini file has changed. For example, the jde.ini may change when you perform upgrades or when you re-configure in release master.

12. If you chose to build individual specification tables, the Individual Specification Selection form appears.
13. To indicate that you do not want to build a specification table, clear its option.

You can clear multiple options.

14. Click Next.

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

15. Complete these fields and click Next:

Field	Description
Build Mode	Specify the build mode, such as debug or optimize.
Stop-Build Option	Specify what action to take if errors occur while building business functions.
Build BSFN Documentation	Specify whether you want to build the documentation for the functions.
Clear Output Destination First	Indicate if you want the destination directory for the functions to be cleared before the build.

16. On the Compression Options form, select the Compress Package option.

    - Select this option to compress the applications included in the package, and to specify options for the compression process.

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

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

17. If you are compressing the package, select from these options:

Option	Description
All Directories	Select to compress all of the directories listed on the Individual Directory Selection form.
Individual Directories	Select to compress only certain directories which you specify.
Compress Data	Indicate whether to compress the data in a package after the package is created. Compress Data compresses the Supported Local Database that is associated with this package.
Compress Foundation	Indicate whether to compress the foundation files in the package after the package is created. Compress Foundation compresses the foundation that is associated with the package.

Note. Verify that the DoCompression setting is set to *1* to enable compression. If this setting is not set to 1, the system does not compress the server package.

See Chapter 5, "Understanding the Package Build Process," Jde.ini Settings for Server Package Builds, page 58.

18. To compress a package, click Compress Options and select All Directories.

    This selection compresses the client and server packages. The system compresses the client package to the deployment server. It compresses the server package on the enterprise servers and copies the files to the deployment server.

19. Click Next.

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

20. On the Individual Directory Selection form, indicate that you want to compress a directory by clicking its option to select it and click Next.

    You can select multiple options.

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

22. On the Build Features form, if you want to build a feature.inf file with the package, select the Build Feature INFs option.

    When you select this option, the Compress and Build options become available.

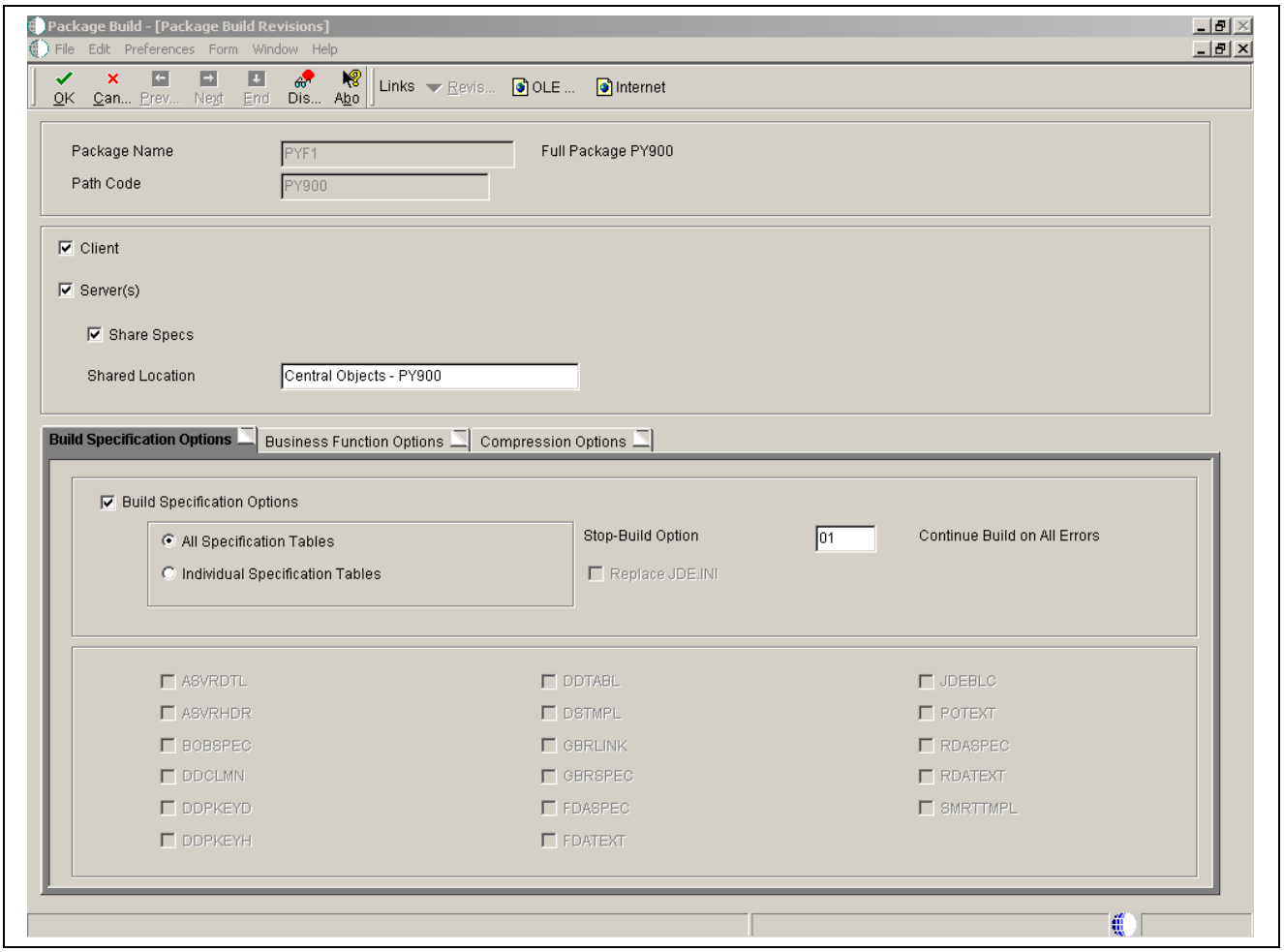
    See Chapter 6, "Building Packages," Configuring Features During the Package Build Definition, page 114.

23. Click Next.

24. Review the package build selections and click End.

Reviewing Package Build Selections

Access the Package Build Revisions form.



Package Build Revisions form

1. Review current build options, business function options, compression options, and feature options that you specified for the package.
2. Click the tab for the type of option that you want to change, and make any changes.

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

   See Chapter 6, "Building Packages," Defining a Package Build, page 88.
3. When you are finished reviewing or changing the build options, click End to exit the Package Build Definition Director, or click OK to accept changes to an existing package.
4. On Work With Package Build Definition, activate the package by choosing Active/Inactive from the Row menu.

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

Building a Package

Access the Work With Package Build Definition form.

1. Select Active/Inactive from the Row menu to activate the package.

2. Select Submit Build from the Row menu when you are ready to initiate the package build.
3. Select one of these options and click OK.
   - On Screen
   - To Printer

   The form closes and the system begins building the package. Build time varies, depending on the number and size of the items in the package. A build could take five minutes for a small package, or several hours for a full package that contains all applications. When the build finishes, the report either appears on the screen or prints, depending on the destination that you specified.
4. Review the report to make sure that all components in the package were built successfully.

   If the report indicates any errors, review the error logs for more detail.

   If the package build finishes successfully, you can schedule the package for deployment.

Incorporating Features into Packages

This section provides overviews of the feature build and deployment process and the JD Edwards EnterpriseOne Feature Based Deployment Director and discusses how to:

- Create a feature.
- Define a file set.
- Define a registry setting.
- Define a shortcut.
- Define additional package build processes.
- Define additional install processes.
- Define an initialization file.
- Define a new open database connectivity (ODBC) data source.
- Import an existing ODBC data source.
- Review feature components.
- Copy features.
- Add a feature to a package.
- Configure features during the package build definition.
- Configure features for an existing package build definition.

Understanding the Feature Build and Deployment Process

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

Note. Oracle's JD Edwards EnterpriseOne web development clients require a specific feature component to develop web-based objects. This feature is also required for mobile packages, because mobile clients require a Web Development client in order to operate.

See *JD Edwards EnterpriseOne Tools 8.98 Web Development Client Installation Guide*

Note. Oracle's JD Edwards EnterpriseOne development clients require a specific feature component to develop integration points. This feature is the Web Services Gateway Development feature.

See *JD Edwards EnterpriseOne 8.98 Web Services Gateway Installation and Setup Guide*

You might also want to include any of these features when you build a package:

- ActiveX controls.

  The Application Design Aid tool enables you to include ActiveX controls in applications. If ActiveX controls are delivered with the software, you need a way to copy these controls to the workstation.

- Open Data Access (ODA) data sources.

  ODA requires that additional ODBC data sources be created on any workstation or server that uses ODA.

- Sales Force Automation databases.

  The Sales Force Automation feature requires that you install a separate Supported Local Database on the workstation so that it can be disconnected from the network during offline operation. You must also write a registry setting that indicates that the machine is used offline.

- BMC Patrol, GenCorba, GenCom, and other third-party interfaces or products.

  Each of these products and interfaces requires additional components on the workstation and server in order to function. As functionality expands to support additional third-party products and interfaces, these products will each have their own set of supporting files.

For software releases prior to JD Edwards EnterpriseOne 8.10, custom programming was required to add feature components to the workstation and server. You can now use familiar tools such as the JD Edwards EnterpriseOne Package Assembly Director and Package Build Definition Director to create a package that contains the feature, and then you can deploy it using the JD Edwards EnterpriseOne Package Deployment Director or multitier deployment.

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

Feature Definition

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

The forms in the JD Edwards EnterpriseOne Feature Based Deployment Director enable you to:

- Create a file set.
- Define registry settings.
- Define a Microsoft Windows shortcut.
- Enter initialization file information.

- Add ODBC data sources.
- Specify the feature build sequence.
- Enter information for third-party products.

Feature Selection During Package Assembly

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

Feature Configuration 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 select the file sets that will be compressed within the package, and to specify the processes that will be run before and after the feature is built.

Package Deployment

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

Workstation Installation and Deployment Server Installation

After you have deployed the package to workstations and deployment servers, use Oracle's JD Edwards EnterpriseOne Workstation Installation and Deployment Server Installation applications to install the package.

Feature Entries in the Package.inf File

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

Installation of Packages Containing Features

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

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

See Also

Chapter 5, "Understanding the Package Build Process," Features, page 74

Chapter 4, "Assembling Packages," Adding Features to a Package, page 46

Chapter 5, "Understanding the Package Build Process," Workstation Packages, page 61

JD Edwards EnterpriseOne Application Release 9.0 Installation Guide

Understanding the Feature Based Deployment Director

The JD Edwards EnterpriseOne Feature Based Deployment Director enables you to define the feature so that it can be included in a package and then deployed to workstations and servers. The forms in the director enable you to specify the name and type of the feature, as well as the different feature components. The Feature Information form enables you to select the types of components to include in the feature, and determines the subsequent forms that appear in the JD Edwards EnterpriseOne Feature Based Deployment Director.

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

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

Copying a Feature Definition

The JD Edwards EnterpriseOne 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.

Forms Used to Incorporate Features into Packages

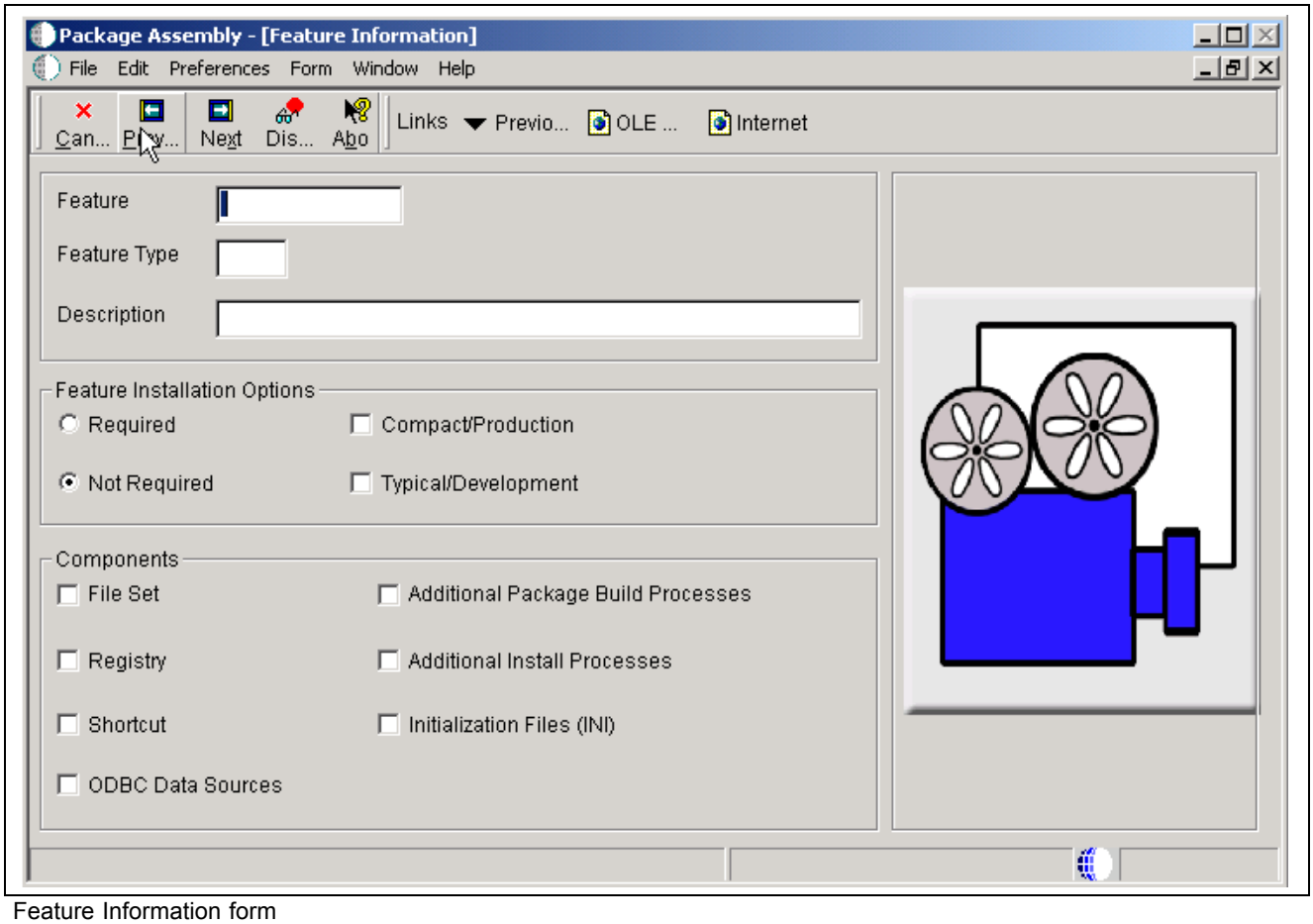
Form Name	FormID	Navigation	Usage
Feature Information	W9326C	Package and Deployment Tools (GH9083), Package Assembly Select Features from the Form menu. Click Add. Click Next.	Define a feature and add one or more components to the feature.
File Set Definition	W9326J	From the Feature Information form, select File Set and click Next.	Enter information about any file sets that must be installed on the workstation or server for the feature to function properly. A file set is a collection of files that must be installed on the workstation or deployment server for the feature to function correctly.
Registry Definition	W9326D	From the Feature Information form, select Registry and click Next until the Registry Definition form appears.	Enter information that should be added to the Microsoft Windows registry as part of the feature installation. Registry information that you enter on this form will be delivered in the package that contains the feature.

Page Name	Definition Name	Navigation	Usage
Shortcut Definition	W9326G	From the Feature Information form, select Shortcut and click Next until the Shortcut Definition form appears.	Use this form to add a shortcut for the feature to the Windows desktop. The system creates a shortcut on the desktop after the feature is installed.
Shortcut Advanced Options	W9326P	From the Shortcut Definition form, select Advanced from the Form menu.	Enter advanced shortcut options.
Additional Package Build Processes	W9326H	From the Feature Information form, select Additional Package Build Processes and click Next until the Additional Package Build Processes form appears.	Specify a batch application or executable program to run either before or after the package that contains the feature is installed.
Additional Install Processes	W9326K	From the Feature Information form, select Additional Install Processes and click Next until the Additional Install Processes form appears.	Enter information about third-party applications that should be run when the package is installed.
Initialization File (INI) Definition	W9326I	From the Feature Information form, select Initialization Files (INI) and click Next until the Initialization File (INI) Definition form appears.	Enter information that should be written to an initialization file (such as jde.ini) as part of the feature installation. The INI file is automatically updated when the package is installed.
ODBC Data Source Definition	W9326N	From the Feature Information form, select ODBC Data Sources and click Next until the ODBC Data Source Definition form appears.	Enter information for any ODBC data sources that must be added to support the feature.
Local Data Sources	W9326O	On ODBC Data Source Definition, select Import from the Form menu.	Select previously created data sources that reside locally on your machine.
Features Summary	W9326L	From the Feature Information form, click Next and add the each feature component. After you add all the components, the wizard displays the Features Summary form.	Review and modify information that you entered on any of the Feature Based Deployments forms.

Page Name	Definition Name	Navigation	Usage
Feature Copy	W9326M	Package and Deployment Tools (GH9083), Package Assembly Select Features from the Form menu. Select the feature from which to copy the definition, and click Copy.	Copy an existing feature and rename it as a new feature. This function is useful if you want to create a feature definition that closely matches an existing feature definition.
Feature Component Selection	W9601AB	Package and Deployment Tools (GH9083), Package Assembly Click Add to create a new package. Enter the forms in the Package Assembly Directory until the Features Component form appears. Click Browse. Package and Deployment Tools (GH9083), Package Assembly Select a package and then select Package Revisions from the Row menu. On Package Component Revisions, click the Features button. To add a feature, click Browse.	Add defined features to a new package. Add defined features to an existing package that is open for revision.
Build Features	W9621B	Package and Deployment Tools (GH9083), Package Build Click Add to launch the Package Build Definition Director. Click Next and complete the screens until you come to the Build Features form. Package and Deployment Tools menu (GH9083), Package Build Find and select the package that contains features. Select Build Revisions from the Row menu. Click the Build Features tab.	Enables you to specify whether the system builds feature INF files for the features in the package. If you defined a fileset component in the feature, you can select 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.

Creating a Feature

Access the Feature Information form.



Feature Information form

Feature Enter a name for the feature.

Feature Type Enter the feature type, if applicable.

Description Enter a description of the feature.

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

Not Required Select this option if the installation of this feature is optional. Whether the feature is installed depends on the options that you select (Compact/Production and Typical/Development). Inclusion of the feature can be overridden when the package is installed.

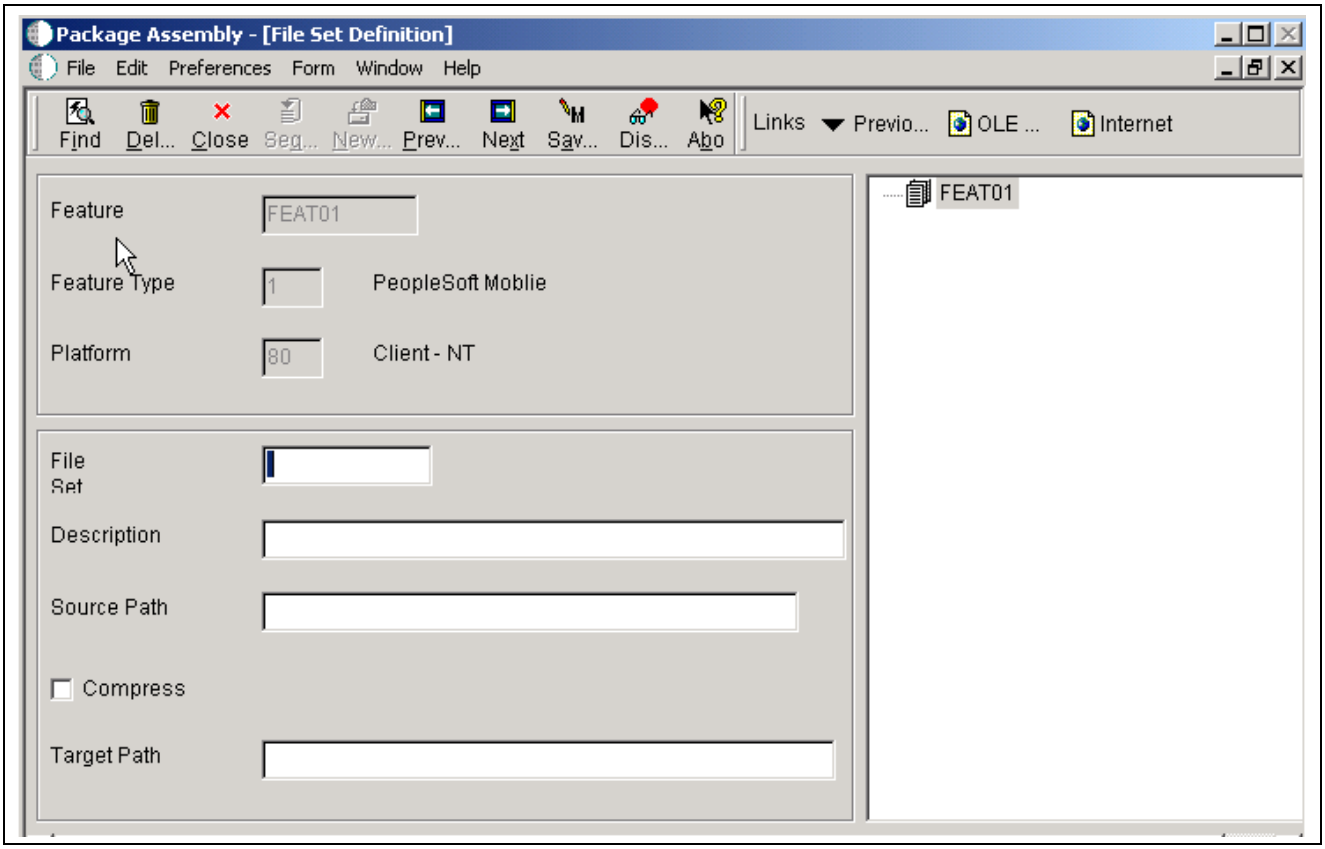
Compact/Production Select this option if this feature is to be included in a Compact/Production install by default. This option can be overridden when the package is installed if Not Required is also selected.

Typical/Development Select this option if this feature is to be included in a Typical/Development install by default. This option can be overridden when the package is installed if Not Required is also selected.

File Set	Select this option if the feature is a file set.
Registry	Select this option if the feature is a registry setting.
Shortcut	Select this option if the feature is a shortcut.
ODBC Data Sources	Select this option if the feature is an ODBC Data Source.
Additional Package Build Processes	Select this option if the feature is an additional process to be performed during the package build.
Additional Install Processes	Select this option if the feature is an additional process to be performed during the install process.
Initialization Files (INI)	Select this option if the feature is an initialization file.

Defining a File Set

Access the File Set Definition form.



FileSet Definition form

File Set	Enter comments or memoranda in this free-form text field.
File Set Description	Enter a description for the group of files.
Source Path	Enter the path that identifies the source location of the file set.
Compress	Select this option to compress the file.
Target Path	Enter a path to identify the target location of the file set.

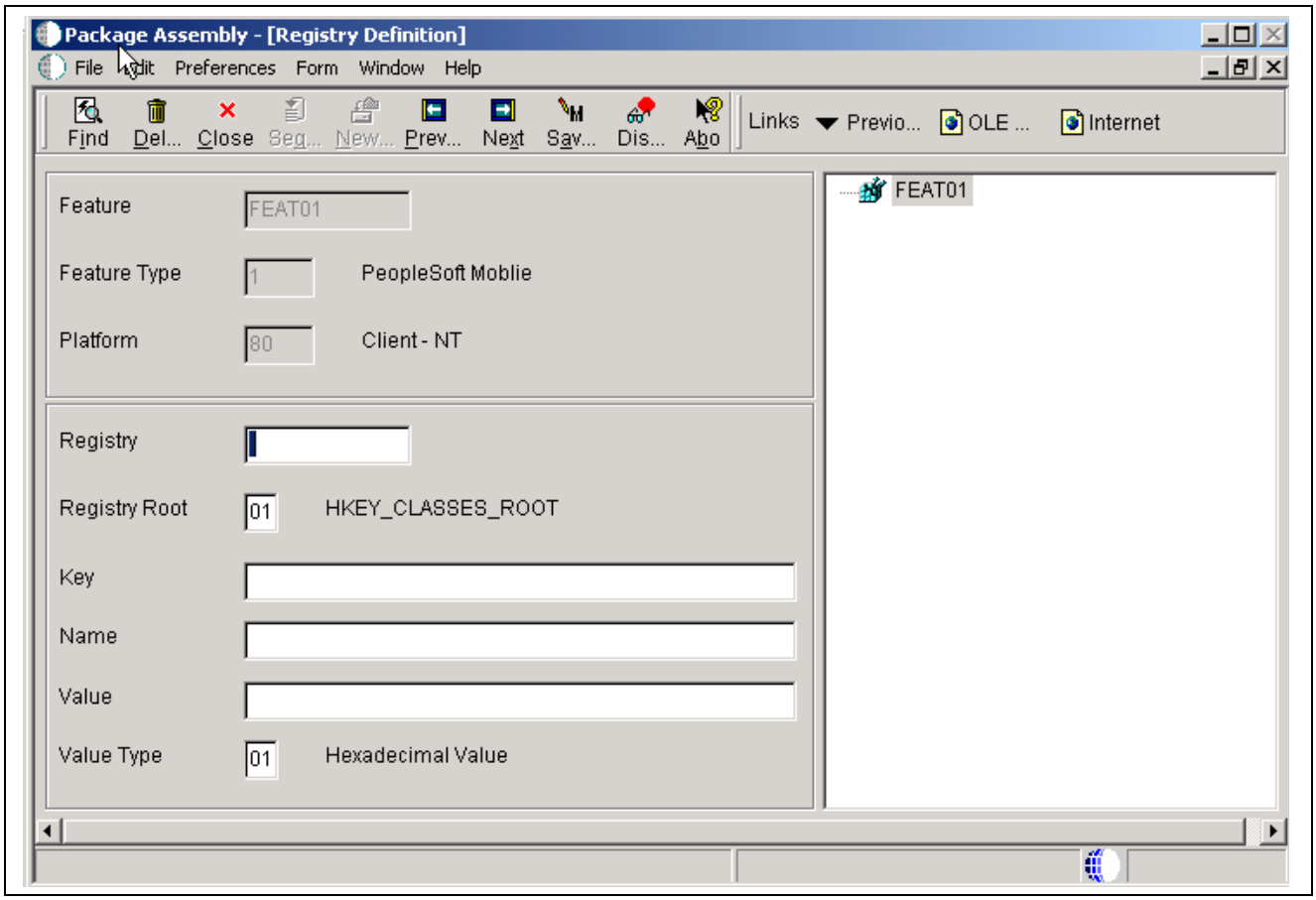
The source path tells the system where to find the file set to be copied into the package, and the target path indicates the location to which the file set should be copied when the package is installed. Although a feature can have an unlimited number of file sets, each file set can have only one target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

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

Always select Save Node from the Form menu when you are finished adding file set information.

Defining a Registry Setting

Access the Registry Definition form.



Registry Definition form

Registry Enter an identifier for the registry modification.

Registry Root Enter the root key in the registry.

Key Enter the key for a registry value.

Name Enter the registry value name.

Value Enter the registry value.

Value Type Enter the data type in which the value is stored in the registry.

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

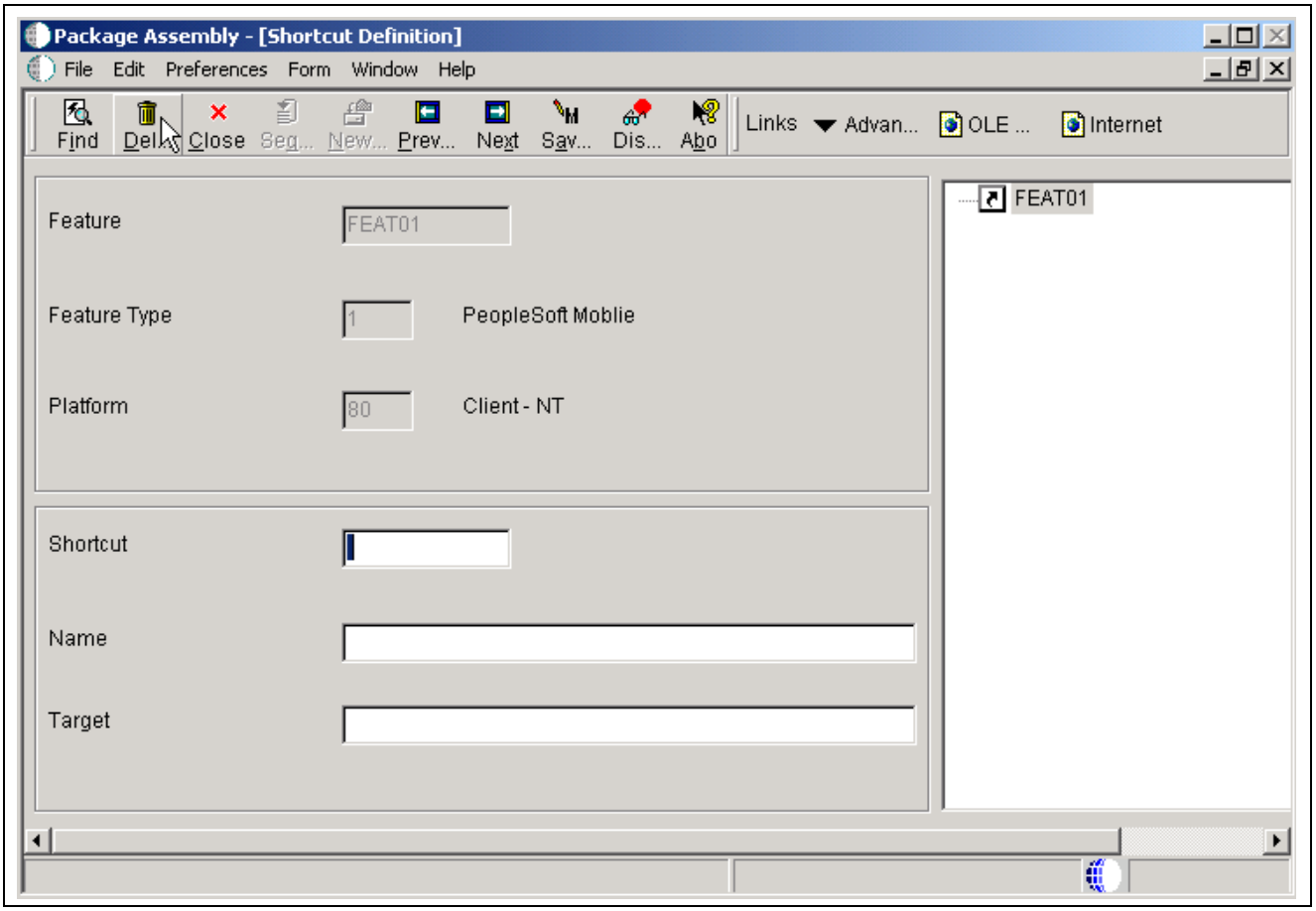
Always select Save Node from the Form menu when you are finished entering registry information.

Defining a Shortcut

To define a shortcut component, you enter a shortcut definition, and then you can enter advanced shortcut options.

Entering a Simple Shortcut Definition

Access the Shortcut Definition form.



Shortcut Definition form

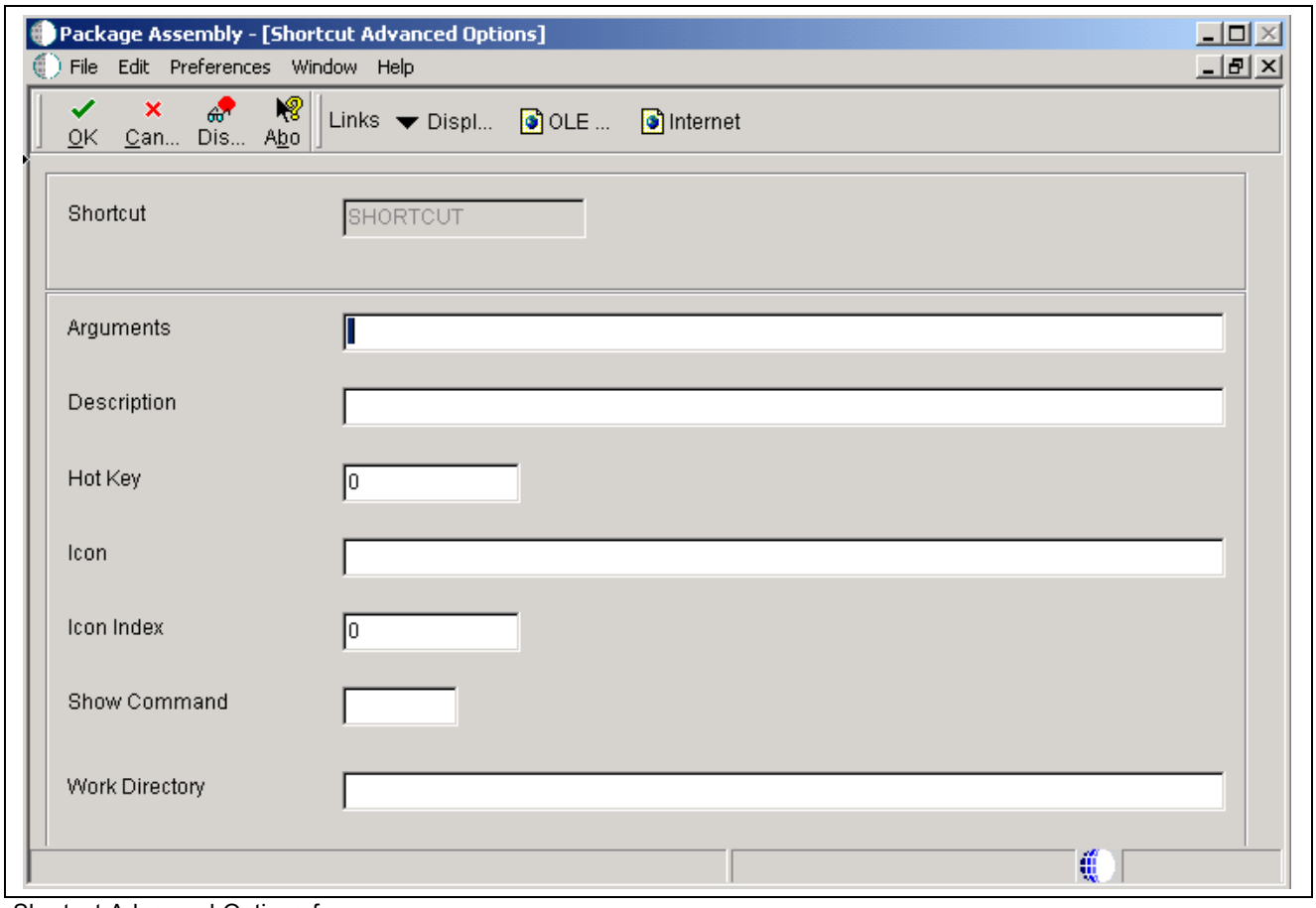
Shortcut Enter a name that identifies a unique shortcut to a user's computer.

Name Enter the name of the shortcut.

Target Enter the path and file name of a target file.

Entering Advanced Shortcut Options

Access the Shortcut Advanced Options form.

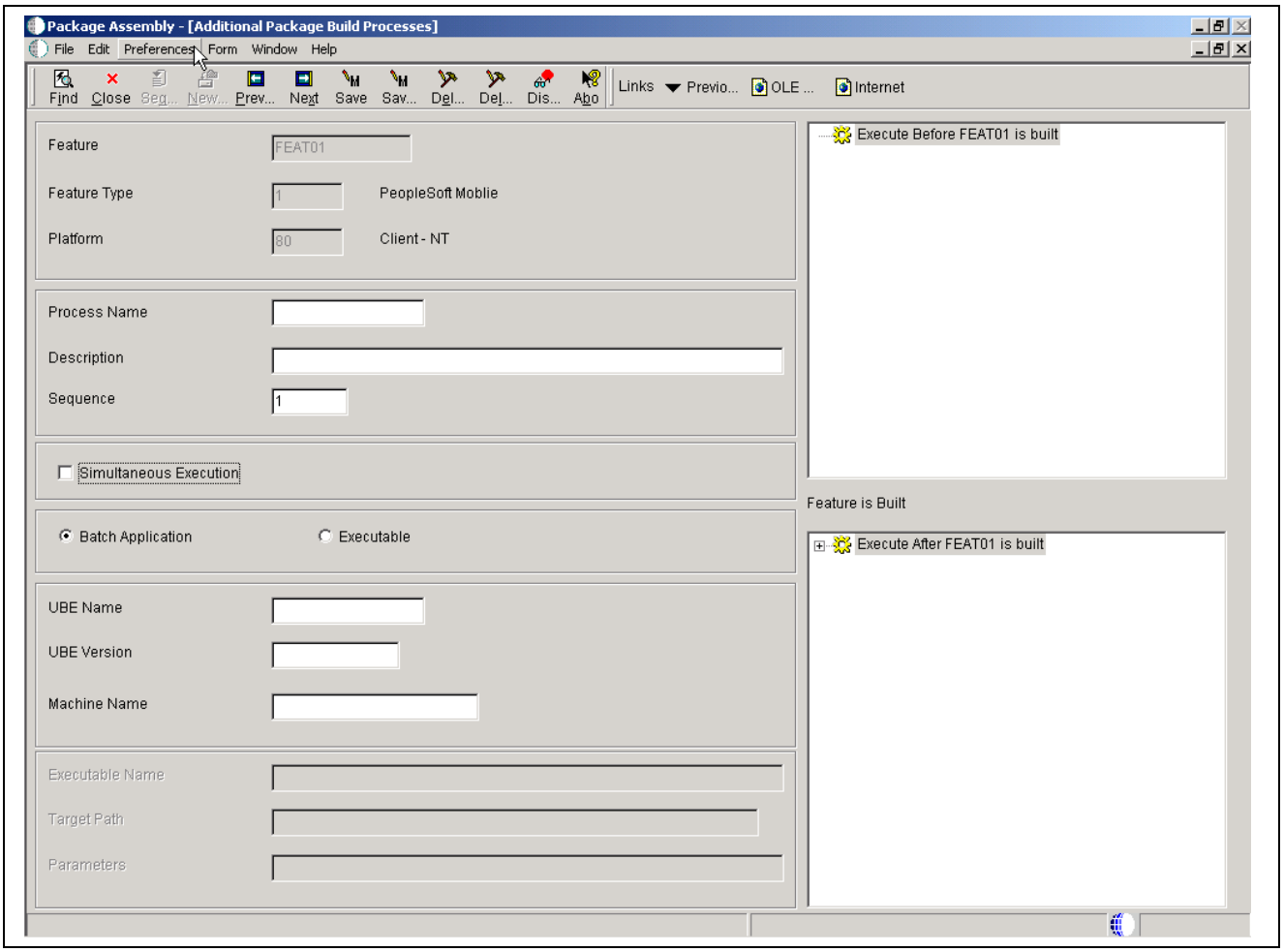


Shortcut Advanced Options form

Arguments	Enter the parameters that are entered at the command line for the shortcut.
Description	Enter a description of the shortcut.
Hot Key	Enter a key sequence that, when pressed, automatically launches the shortcut.
Icon	Enter the path and name of the icon file, based on a relative target path.
Icon Index	Enter the icon index for a shortcut.
Show Command	Specify the size of the window after the shortcut is launched. For example, the window might be minimized or maximized.
Work Directory	Enter the identifier of the directory path or the working directory of a shortcut.

Defining Additional Package Build Processes

Access the Additional Package Build Processes form.



Additional Package Build Processes form

Process Name Enter the name of the build process.

Description Enter a description of the build process.

Sequence Enter a number to identify the order in which the process will be run relative to the other processes that run during the package build.

Synchronous Execution Select this option to indicate whether the package build job waits for the process to finish before it continues.

Batch Application or Executable Specify whether the process is an application or an executable.

UBE Name Enter the name of the batch application. Only applies if batch application was selected.

UBE Version Enter the version of the batch application. Only applies if batch application was selected.

Machine Name Enter the name of the server or workstation on which the batch application will run. Only applies if batch application was selected.

Executable Name Enter the name of the executable program that the system launches to install the third-party software. Only applies if executable program was selected.

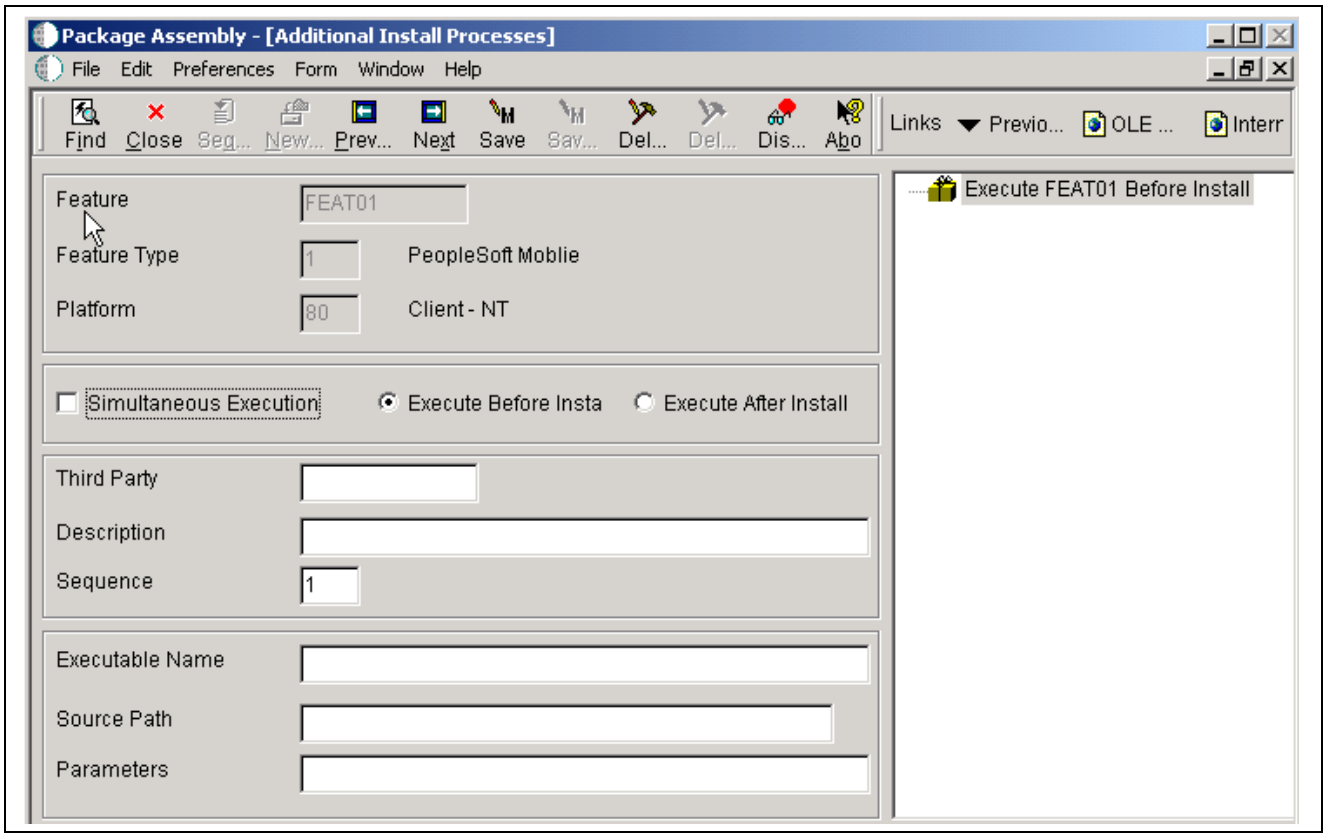
Target Path Enter the path and file name of a target file. Only applies if executable program was selected.

Parameters Enter the executable parameters that the setup program uses to install the third-party software. Only applies if executable program was selected.

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

Defining Additional Install Processes

Access the Additional Install Processes form.



Additional Install Processes form

Third Party Enter the name of the third-party component.

Description Enter a description of third-party software.

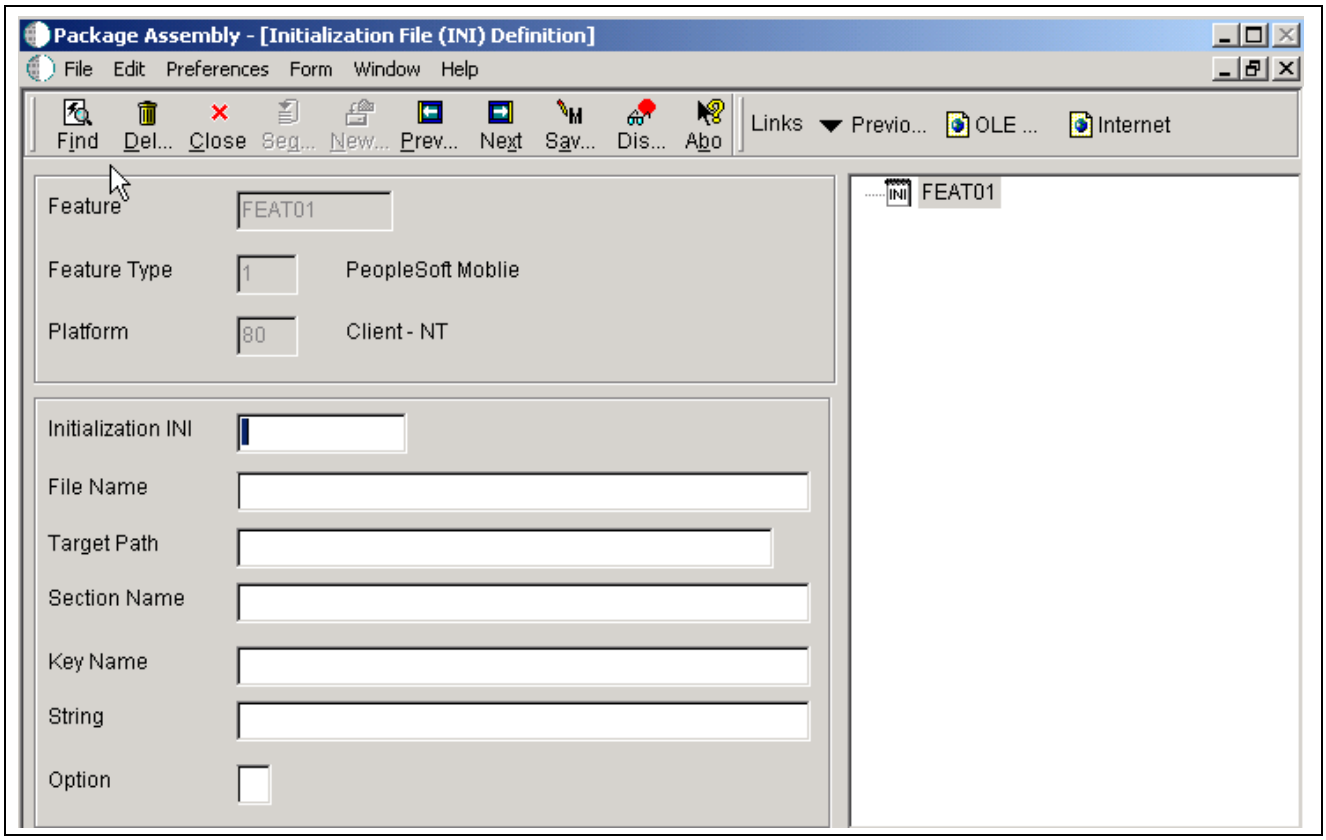
Sequence Enter a number to identify the order in which this process will run relative to the other additional install processes.

Synchronous and Execute After Install	Clear the Simultaneous Execution option and select the Execute After Install option. The third-party process waits for the JD Edwards EnterpriseOne client install to finish before running.
Synchronous and Execute Before Install	Clear the Simultaneous Execution option and select the Execute Before Install option. The JD Edwards EnterpriseOne client install will run the third-party process and wait until it finishes before installing the client.
Asynchronous and Execute After Install	Select the Simultaneous Execution option and the Execute After Install option. The JD Edwards EnterpriseOne client install finishes, and then starts the third-party process. Neither process waits for the other to finish before proceeding.
Asynchronous and Execute Before Install	Select the Simultaneous Execution option and the Execute Before Install option. The JD Edwards EnterpriseOne client install begins, and then immediately starts the third-party process and resumes the client install without waiting for the third-party process to finish.
Executable Name	Enter the name of the program that launches the third-party software.
Target Path	Enter the path to the executable file. Do not include the name of the file.
Parameters	Enter the executable parameters that the system passes to the third-party program.

Note. Select Save from the Form menu when you finish adding third-party product information.

Defining an Initialization File

Access the Initialization File (INI) Definition form.



Initialization File (INI) Definition form

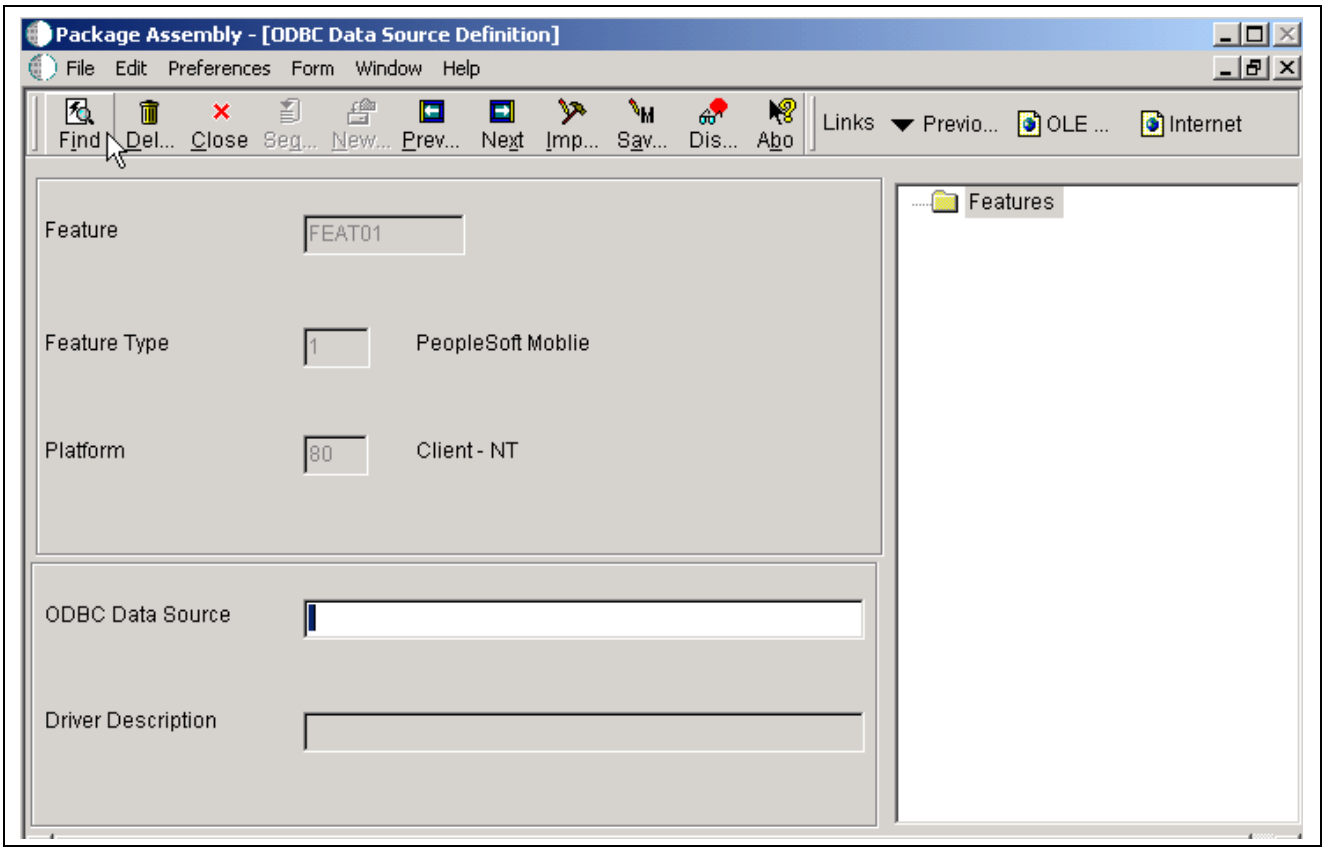
Initialization INI	Enter the identifier of an initialization file component.
File Name	Enter the name of the initialization file.
Target Path	Enter the path of the INI file.
Section Name	Enter the name of the application section in an initialization file.
Key Name	Enter a key in the initialization file that is to be added, modified, or removed.
String	Enter the value of the key in an initialization file.
Option	Enter the option that identifies the action associated with the key in the initialization file.

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

When you finish adding initialization information, select Save Node from the Form menu.

Defining a New ODBC Data Source

Access the ODBC Data Source Definition form.



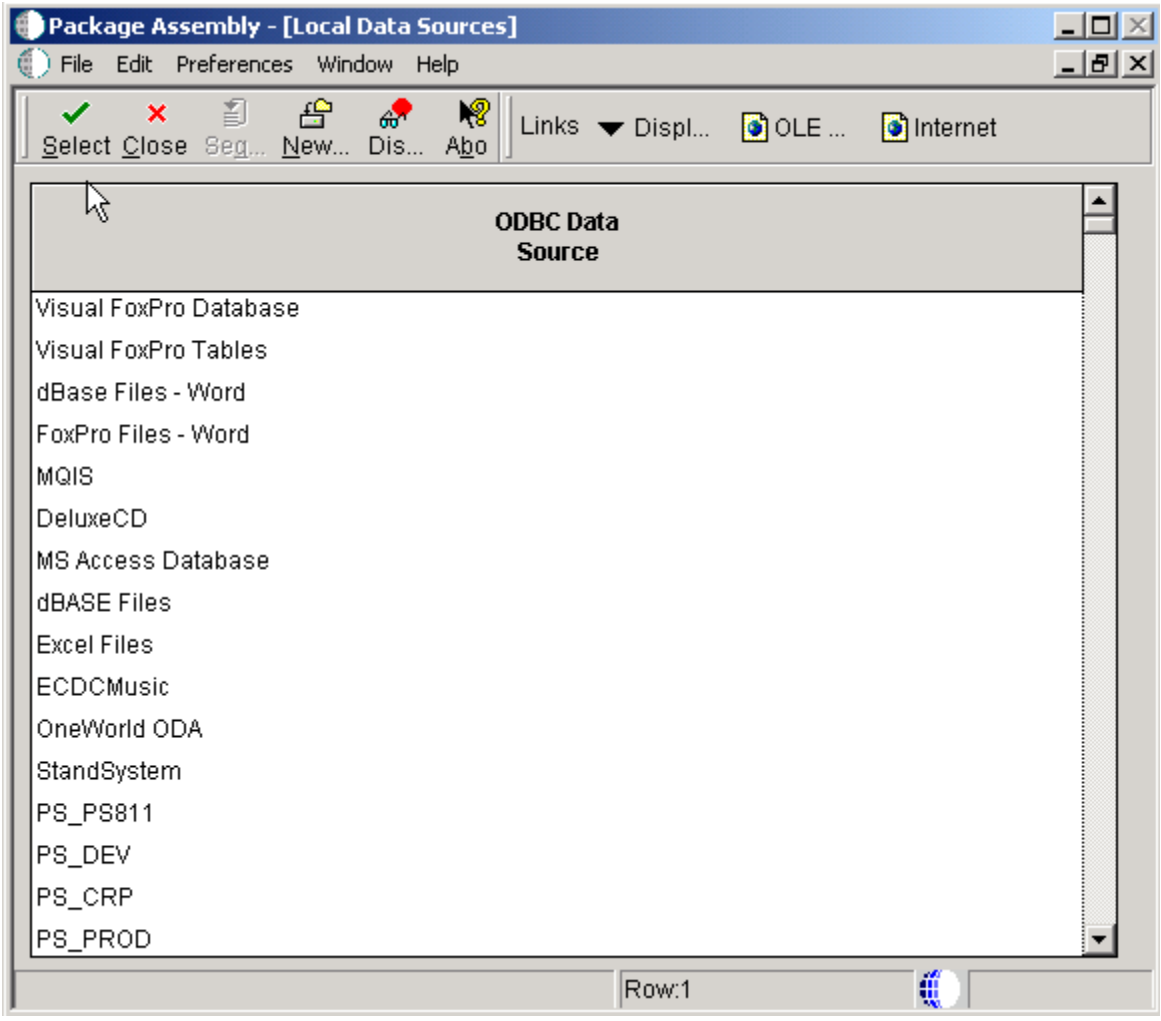
ODBC Data Source Definition form

ODBC Data Source Enter the name of the data source.

Note. When you select Save Node from the Form menu, the system activates the Microsoft Windows control panel applet that displays the ODBC Data Source forms where you can enter the data source information.

Importing an Existing ODBC Data Source

Access the Local Data Sources form.



Local Data Sources form

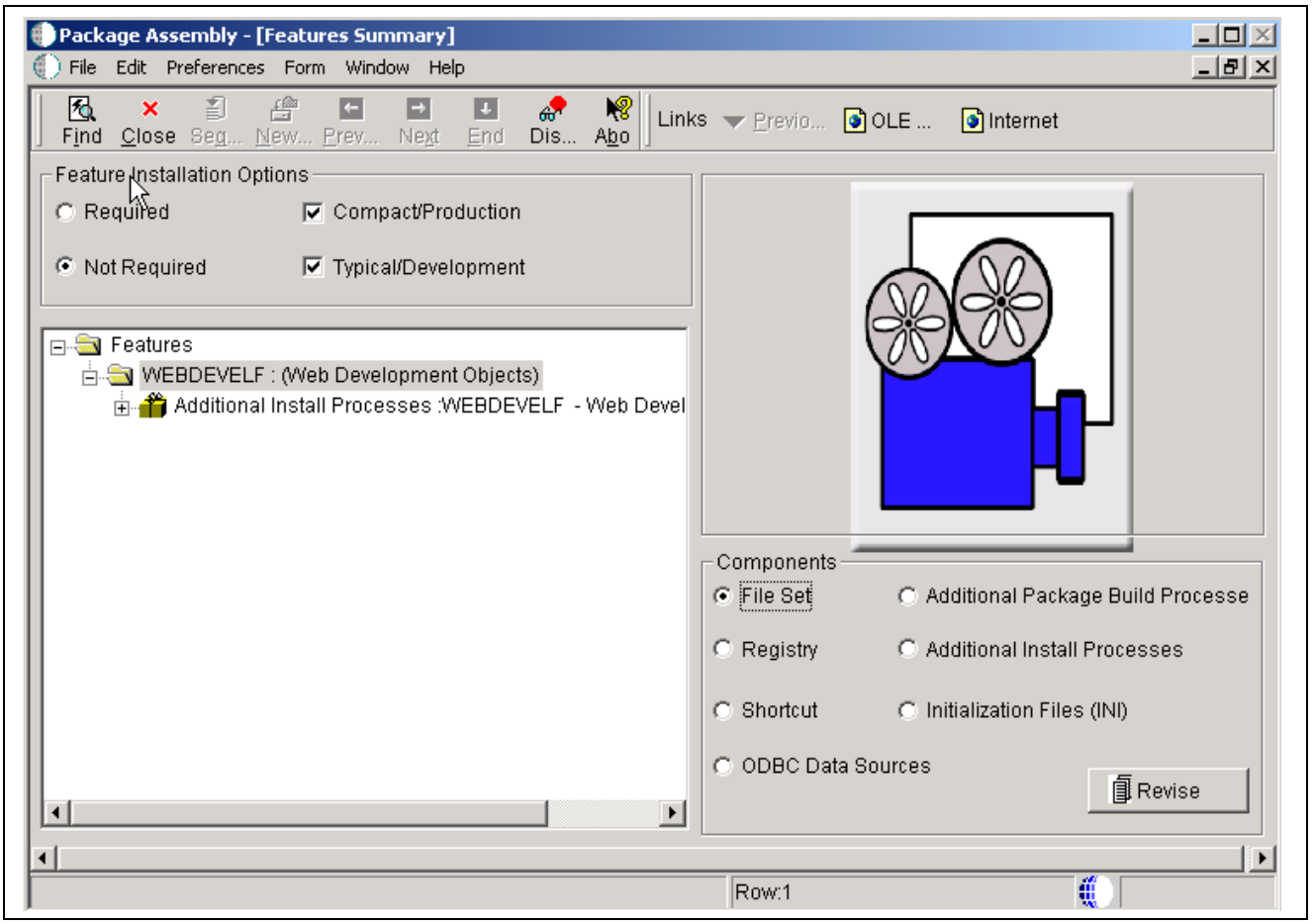
1. Press the CTRL or SHIFT key to select one or several data sources, and click Select to add the data sources to the feature.

   The ODBC Data Source Definition form reappears.

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

Reviewing Feature Components

Access the Features Summary form.



Features Summary form

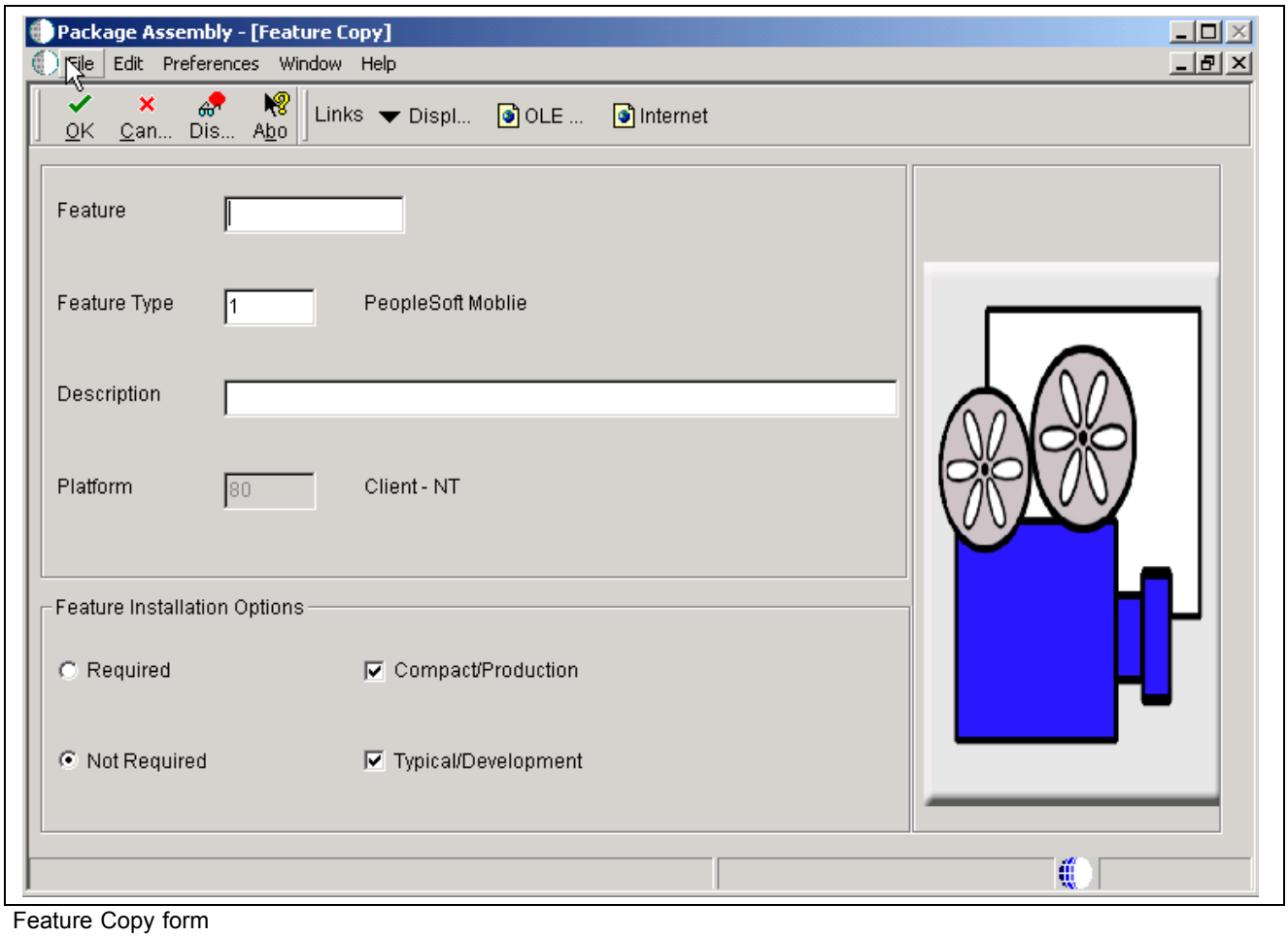
1. Select a component in the right pane and click the Revise button to review the information for that component.
2. If needed, change the field values for the selected component and click Save.
3. Repeat the previous steps to modify other components.
4. When you are finished defining the feature, click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

See Also

Chapter 4, "Assembling Packages," Revising an Existing Package, page 50

Copying Features

Access the Feature Copy form.



Feature Copy form

1. Complete these fields:
   - Feature
   - Feature Type
   - Description
2. Select one of these options:

Option	Description
Required	The installation of this feature is mandatory for both Compact/Production and Typical/Development installs. Inclusion of this feature cannot be overridden when the package is installed.
Not Required	The installation of this feature is optional. Whether the feature is installed depends on the options that you select (Compact/Production and Typical/Development). Inclusion of the feature can be overridden when the package is installed.

3. Select one or both of the options that follow.

   If you chose Required, both of these options are automatically selected.

- Compact/Production

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

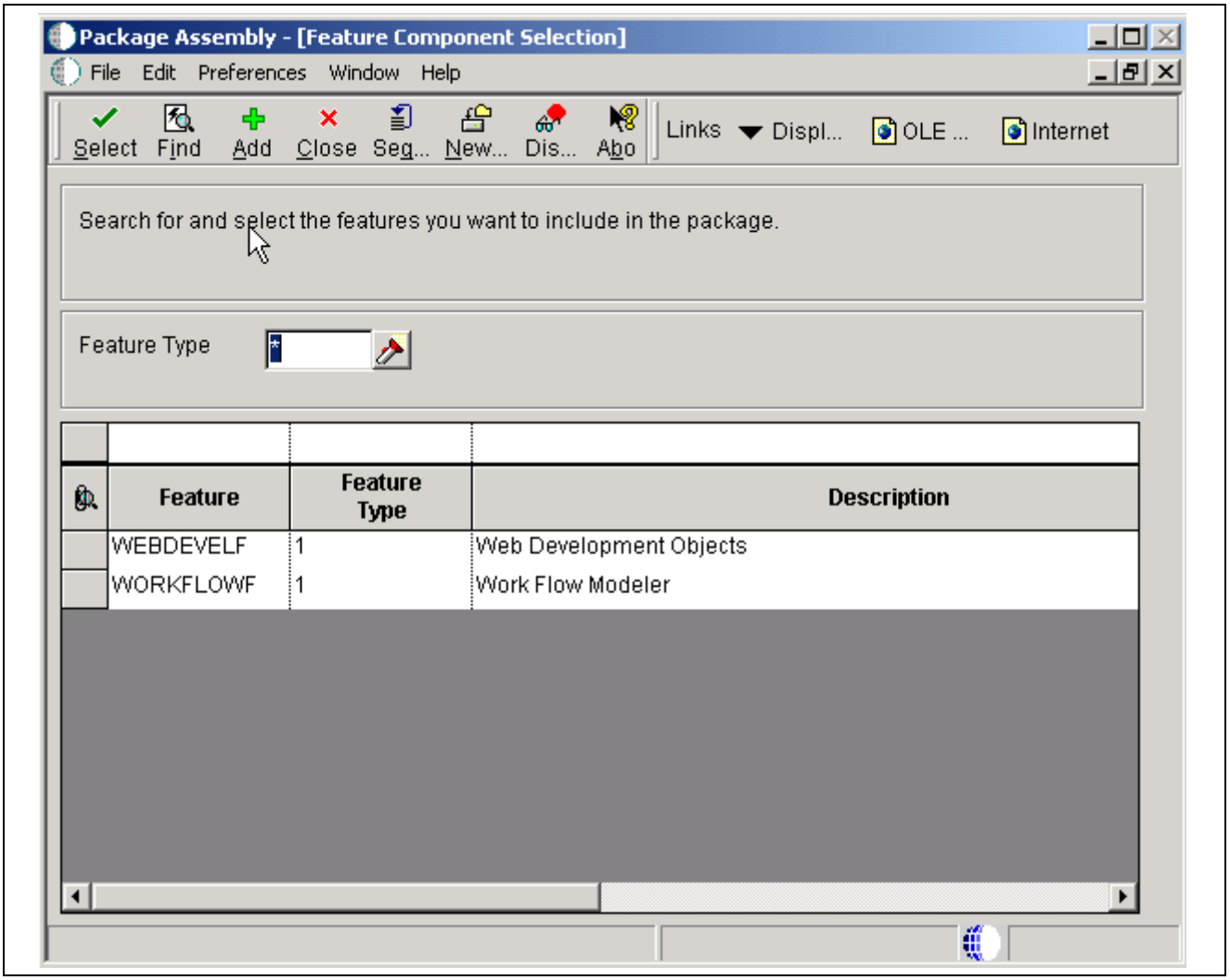
- Typical/Development

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

4. Click OK.
5. To revise the new feature definition, select the feature and select Revise Feature from the Form menu.

Adding a Feature to a Package

Access the Feature Component Selection form.



Feature Component Selection form

1. Click Find to display the list of available features.

Note. Before a feature is available for inclusion in the package, you must first define the feature.

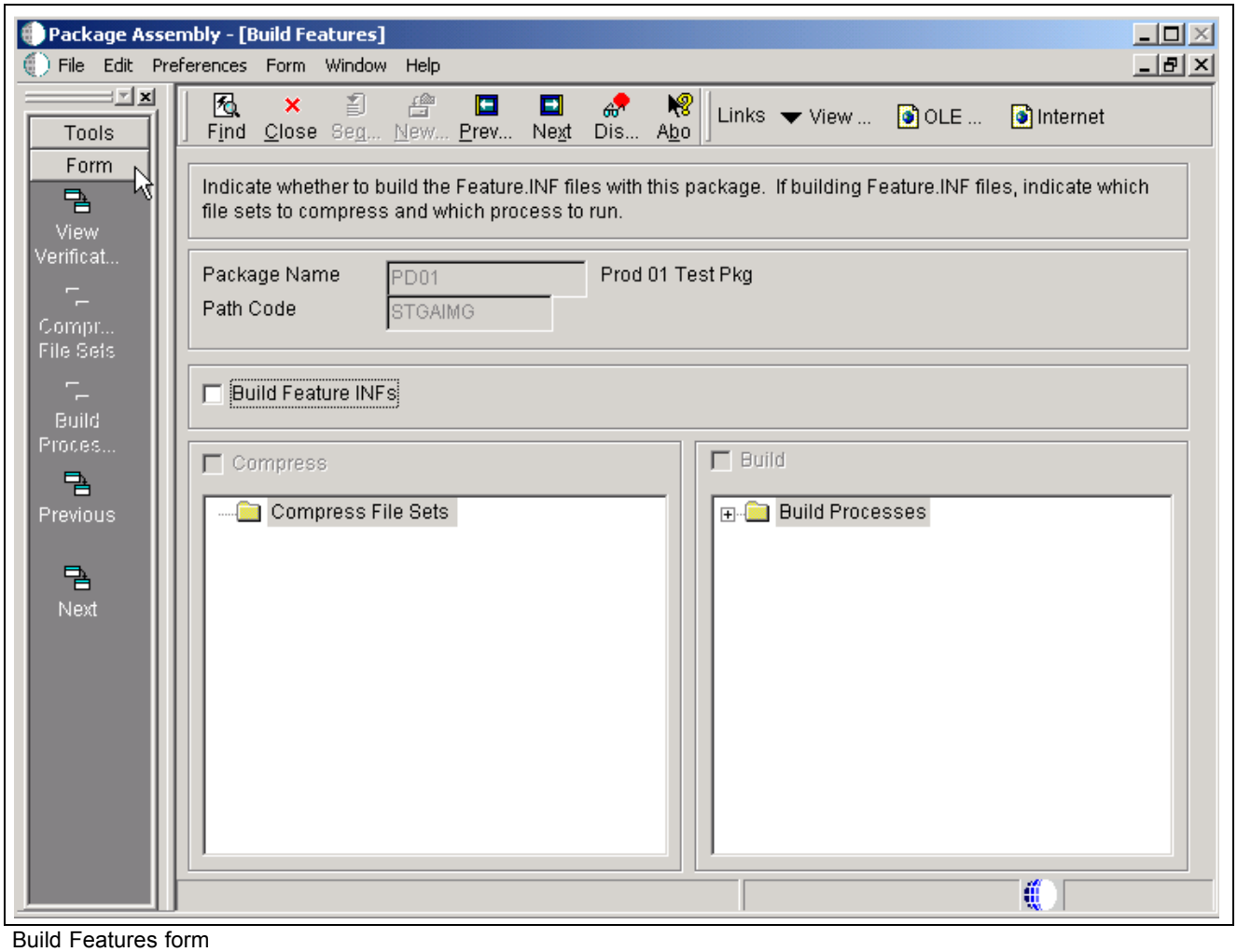
2. Use one of these methods to select one or more features to include in the package:
   - Select a feature and click the Select button.

     (Press the CTRL or SHIFT key to select multiple features.)
   - Double-click each feature.
3. When you are finished adding features, click Close to return to the Features Component form. The selected features appear.
4. Click Next and complete the remaining forms to finish assembling the package.

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

Configuring Features During the Package Build Definition

Access the Build Features form.



Build Features form

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

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

2. Continue with one or both of these tasks:
   - To compress file sets
   - To build processes

1. Select Compress, and then select Compress File Sets from the Form menu.
2. On the File Set Selection form, select each feature that you want to include by choosing a file set and clicking Select.
3. When you are finished selecting file sets, click Close.
4. Continue either by performing the next steps, or by clicking Next and completing the remaining forms to finish defining the package build.
5. To build processes, select Build, and then click Select Build Processes.
6. On the Build Processes Selection form, select each process that you want to build by choosing a process and clicking Select.
7. When you are finished selecting processes to build, click Close.
8. From the Form menu, select Build Processes and manually select each process to run during the package build.

   You must complete this step or none of the processes will run, even though they are included in the feature.
9. Click Next and complete the remaining forms to finish defining the package build.

Configuring Features for an Existing Package Build Definition

Access the Build Features form.

1. Modify or add to any of these existing build feature settings:
   - Build Feature INFs
   - Compress
   - Build
2. If you select Compress, select Revise File Sets from the Form menu to modify file sets.
3. When you are finished modifying file sets, click Close.
4. If you chose Build, click Revise Processes to modify processes.
5. When you are finished modifying processes, click Close.
6. If you selected Build, from the Form menu, select Build Processes and manually select each process to run during package build.

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

Viewing Package Build Records and Resubmitting Builds

This section provides overviews of package build history and the build status and discusses how to:

- View the package build history.
- View log files.
- Resubmit a package build.
- Change the build status.
- Reset the specification build and package build statuses.

Understanding Package Build History

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

- Package name.
- Path code.
- Date and time built.
- Name of the server for which the package was built.
- Current build status and status description.
- Current status of selected specification tables.
- Number of specifications written.
- Package records written and read.

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

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

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

F96225 Table

The system maintains a history of the package build in the F96225 table. This table contains details about the package build statuses of any package components.

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

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

Logs

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

Where to Find the Error Logs

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

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

In these examples, PD900FA is used as the package name. To determine the actual directory, substitute the package name for PD900FA.

- Package statistics: \PD900FA\work\buildreport.log
- Client package build log: \PD900FA\clientpkgbuild.log
- Server package build log: \PD900FA\svrpkgbuild.log
- Mobile object list log: \PD900FA\work\mobileobjectlist.txt
- Business function errors log: \PD900FA\work\buildlog.txt
- Missing business function source log: \PD900FA\work\NoSource.txt

Package Statistics Log

The package statistics log summarizes the outcome of the package build, showing statistics for the directories in the package, including the size and file count of each directory. This log displays a complete build that you can use to review the build directories. The report shows a breakdown of the files in the specifications directory and the size of each spec file, as well as the total count and size. You can use this log to verify that the package built successfully.

Client Package Build Log

The client package build log lists the steps completed in building the client package, as well as any errors that occurred during the process. The first page of the build log will identify the compiler version used by the package (for example, "Microsoft Visual Studio Version being used: 8"). The final page tells you whether the package was built successfully. This log file is created for a client-only or client/server package.

Server Package Build Log

The server package build log lists the steps completed in building the server package, as well as any errors that occurred during the process. The final page tells you whether the package was successfully built and deployed. This log is created for a server-only or client/server package.

Mobile Object List Log

The mobile object list log lists all the mobile objects whose specifications are included as part of the mobilespec database. It lists all mobile applications, client-only named event rules (NERs), and mobile Universal Batch Engines (UBEs) that are part of the mobilespec repository.

Business Functions Errors Log

The business functions errors log enables you to view any errors that occur while business functions are being built. The final page of the log describes whether the business functions were successfully built or were built with errors. Business functions that appear on this report might be business functions that are still in development and have not yet been checked in. Business functions that have never been checked in do not have source, and therefore, are listed in the missing business function source errors log.

Missing Business Function Source Errors Log

The missing business function source errors log describes any business functions in the package that are defined in the Object Librarian and have a record, but could not be built because no source existed.

Server Logs

All compile logs for the enterprise server are located on the server itself in the source directory of the DLL in which the object belongs. For example, suppose that you want to see the log for the Sales Order Entry Master Business Function (B4200310) in the package PACKAGE1 on an HP 9000 for which the BuildArea is /u02/jdedwards/packages. The system creates a file called /u02/jdedwards/packages/PACKAGE1/CompileLogs/CDIST/b4200310.log (or b4200310.err if there are errors) because B4200310 is in the CDIST.DLL.

If the system could not link the CDIST.DLL (shared library) on the HP 9000, it would create a file called /u02/jdedwards/packages/PACKAGE1/obj/CDIST/CDIST.log.

On the iSeries, logs for business functions that failed to compile are members in a file called FAILED in the package library. Using the previous example, you would review member B4200310 of the FAILED file in library PACKAGE1.

Understanding the Build Status

In some cases, you might need to rebuild the package rather than resume the build from the point at which the build failed. Before you can do so, you must change the status of the package build from *Build Started* to *Build Definition Complete.*

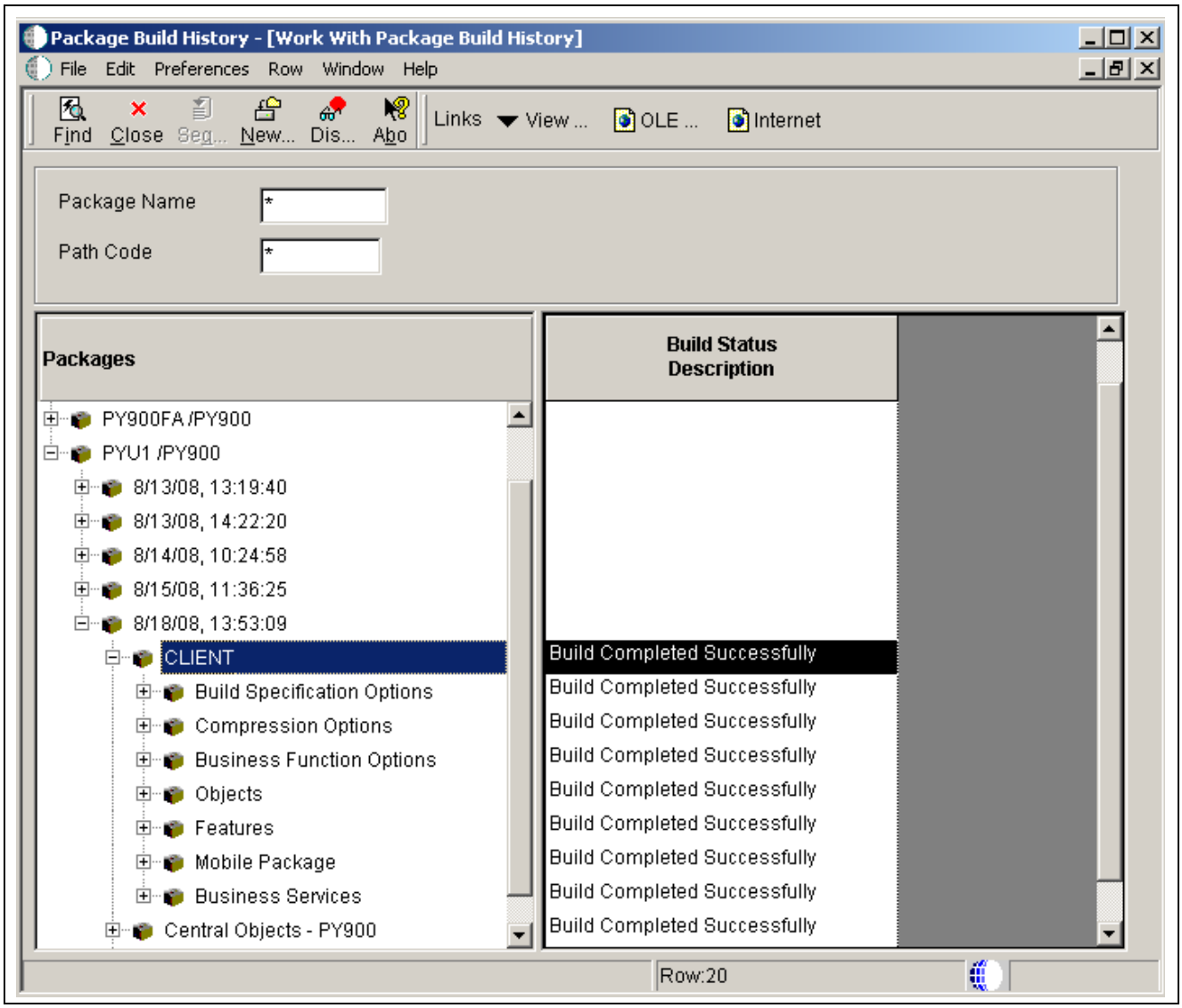
When you reset the status of the package build, you can reset the status for the server only or for all servers and client workstations for which you want to build the package.

Forms Used to View Package Build History and Logs

Form Name	FormID	Navigation	Usage
Work With Package Build History	W9622A	Package and Deployment Tools (GH9083), Package Build History	Display information about the current build status and build options for selected computers.
View Logs	W9622B	Package and Deployment Tools (GH9083), Package Build History Select View Logs from the Form menu.	Check logs for errors that occurred during the build process.
Work with Package Build Definition	W9621L	Package and Deployment Tools (GH9083), Package Build W9621L	Change the package build status.
Reset Build Status	W9622C	From Work with Package Build History, find the package for which you want to reset the statuses, expand the package, and select an individual item. Select Reset Status from the Row menu.	Reset the spec status and pack status for a package to the statuses that you specify.

Viewing the Package Build History

Access the Work With Package Build History form.



Work with Package Build History form

1. Select CLIENT or the server or the spec data source to display information about the current build status for those computers.

   You can also expand the tree to view this information:

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

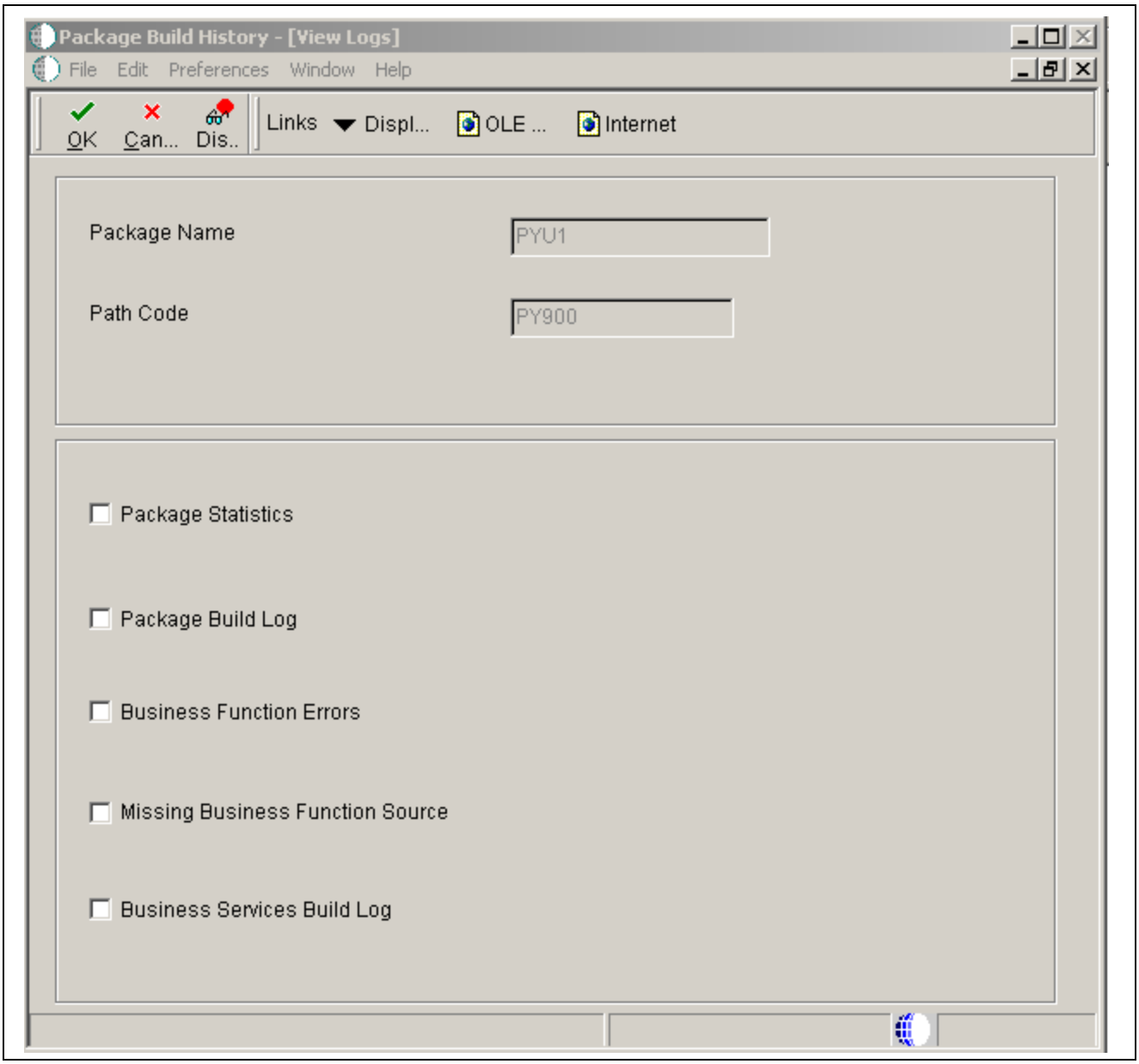
   These options and objects are those that you specified when you created the build definition for the package. For example, if you chose to build only selected specifications, you can determine the status for each specification, as well as other pertinent information.

   Note. Only specification options are built in the spec data source.

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

Viewing Log Files

Access the View Logs form.



View Logs form

Package Statistics Select this option to be able to view count and size statistics for the package directories that were built.

Package Build Log Select this option to be able 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.

Business Function Errors Select this option to be able 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	Select this option to see a list of 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 F9860 table. 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.
Business Services Build Log	Select this option to be able to view the results of the business services build for this package.

Resubmitting a Package Build

Access the Work With Package Build History form.

1. Select one of these options to find the package you want to resubmit:
   - Select a specific server to resubmit only the builds for that server.
   - Select the CLIENT heading to resubmit only the workstation builds.
2. Select Resubmit Build from the Row menu.

   If you generated NERs when you initially submitted the build, the system displays a window that asks whether you want to regenerate the NERs.
3. Click OK to regenerate NERs, or click Cancel to skip this process.

   Note. If you do not want to regenerate NERs, you can prevent this window from appearing by entering *2* in the Generate NER processing option for the Package Build History program.
4. Select one of these destinations for the build report, and click OK:
   - On Screen
   - To Printer

   The form closes, and the system begins to build the package. Build time varies, depending on the number and size of the items in the package. When the build is finished, the report either appears on the screen or prints, depending on the destination you specified.
5. Review the report to verify that the system successfully generated all components in the package.

   If the report indicates any errors, review the error logs for more detail.

See Also

Chapter 7, "Deploying Packages," page 125

Changing the Build Status

Access the Work with Package Build Definition form.

1. Find the package for which you want to reset the status.

   Below the package name, select the server or servers and client workstation for which you want to build the package.
2. From the Row menu, select Advanced.

3. On the Advanced Revisions form, click Reset to change the status of the package build from *Build Started* to *Build Definition Complete*.
4. Click OK.
5. If desired, select the package name and select Submit Build from the Row menu.
6. The program asks whether you want to delete the current build or to continue without deleting it; select one.

Resetting the Specification Build and Package Build Statuses

Access the Reset Build Status form.

1. Enter the desired statuses in the Spec Build Status and Pack Build Status fields.

   Both of these fields have a visual assist feature to help you determine the available statuses.

   Note. The values of these two fields are dependent on each other. If you change one value, be sure you understand the dependency on the other value.

   ---

2. Click Reset.
3. Click OK.

CHAPTER 7

Deploying Packages

This chapter provides an overview of package deployment and discusses how to:

- Define deployment parameters.
- Work with package deployment.
- Deploy server packages.
- Use push installation.
- Install workstations from CD.

Understanding Package Deployment

After you assemble and build a package, you can select from several methods of deploying the package to workstations and servers throughout the enterprise. For workstations, the method that you select depends on whether Oracle's JD Edwards EnterpriseOne is already installed on the workstation.

This section discusses:

- Deploying to workstations without JD Edwards EnterpriseOne.
- Deploying to workstations with JD Edwards EnterpriseOne already installed.
- Deploying to servers.
- Deploying to tiered locations.
- Deploying to workstations from CD.

Deploying to Workstations Without JD Edwards EnterpriseOne

If JD Edwards EnterpriseOne is not currently installed on a workstation, you can deploy the package through Oracle's JD Edwards EnterpriseOne Workstation Installation program. You can use JD Edwards EnterpriseOne Workstation Installation to deploy full packages, but you cannot use JD Edwards EnterpriseOne Workstation Installation to deploy an update package to a workstation on which JD Edwards EnterpriseOne is not installed.

JD Edwards EnterpriseOne Workstation Installation retrieves items that are specified in the package. A package is like a bill of materials with instructions that describe from where the system retrieves all of the necessary components that the JD Edwards EnterpriseOne Workstation Installation program deploys to the local workstation. This program can be run interactively (initiated by a person at a workstation) or in silent mode and scheduled through the push installation feature.

If you use the push installation feature, you can use Oracle's JD Edwards EnterpriseOne Package Deployment to deploy the package. Push installation enables the administrator to initiate the installation of a package from the deployment server to workstations without any user interaction. To use this feature, the push installation *listener* application must be installed on the workstation, and the machine must be defined through Oracle's JD Edwards EnterpriseOne Deployment Locations application (P9654A).

See Also

JD Edwards EnterpriseOne Application Release 9.0 Installation Guide

Deploying to Workstations with JD Edwards EnterpriseOne Already Installed

To reload a new package on workstations on which JD Edwards EnterpriseOne is already installed, use one of two methods:

- JD Edwards EnterpriseOne Workstation Installation (for full packages).
- Oracle's JD Edwards EnterpriseOne Deployment Director (P9631) (for full and update packages).

After you assemble and build a package, use JD Edwards EnterpriseOne Deployment Director to schedule the package for deployment to individual workstations or to selected groups. On the specified deployment date, when the users who are scheduled to receive the package sign in, they are given the opportunity to load the package.

Unless you are using the Push Installation feature, JD Edwards EnterpriseOne Deployment Director requires that JD Edwards EnterpriseOne be already loaded on the workstation. You can schedule a new full package to replace the existing package, or an update package to be merged with the existing package on the workstation.

Both deployment methods have advantages. JD Edwards EnterpriseOne Workstation Installation is a good method to use when you want to install a package immediately or soon after it is built, without having to schedule the package. Alternatively, JD Edwards EnterpriseOne Deployment Director is useful if you need to control when the package becomes available, if you want to make the package installation mandatory, or if you want to deploy the package to servers as well as to workstations.

Deploying to Servers

Servers receive the same package that you build for the workstation, but in a different format. When you assemble the package and create the package build definition, you can specify the servers to which you want to build and deploy the package. To deploy the package, you use the JD Edwards EnterpriseOne Deployment Director application (P9631), which uses the same scheduling mechanism to deploy packages to workstations. In fact, you can easily schedule deployment to both client workstations and servers on the same form. You cannot use the Push Installation feature to deploy to servers.

Deploying to Tiered Locations

Multitier deployment enables 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 software installations per day, or when workstation installations over your wide area network (WAN) are too slow.

Deploying to Workstations from CD

If your system has a CD writer, you can define the CD writer as a deployment location. Essentially, you define the CD writer as a pseudo deployment server from which you can copy a package onto a blank CD. You can then use this CD to install the software on workstations by using the JD Edwards EnterpriseOne Workstation Installation program that is included on the CD.

Defining Deployment Parameters

This section provides an overview of deployment parameters, provides prerequisites, and discusses how to:

- Define machines.
- Define locations.
- Define package deployment groups.
- Revise package deployment groups.

Understanding Deployment Parameters

Before you deploy packages, you must identify the workstations, servers, groups, or locations that will receive the package. Identifying these ensures that, when you are ready to schedule packages using the JD Edwards EnterpriseOne Deployment Director, the machines, groups, or locations that you want to receive the package will be available as package recipients.

A deployment group is a group of workstations that are classified by a criterion such as job function, team, or any other grouping that you specify. For example, you might have a software development group, a testing group, a production group, and so on. Oracle's JD Edwards EnterpriseOne Package Deployment Groups Revisions program (P9652A) enables you to define or revise groups that include several workstations.

A location is a group of workstations and servers that corresponds to a physical location. For example, you might have locations for Corporate and Branch, or for Building 5 and Building 7. Locations are also useful if you use multitier deployment or deploy across a WAN. In this case, you might define a location for each of your geographic locations. The JD Edwards EnterpriseOne Deployment Locations Application program (P9654A) enables you to define or revise machines and locations in your enterprise.

Both of these applications simplify the deployment process when you need to deploy a package to several users. Rather than requiring you to schedule deployment to each workstation or server, you can schedule deployment according to location or group.

When you enter a machine definition, you are really defining its usage in the configuration. For example, you can use a deployment server as a data server. When you enter machine definitions, consider these recommendations:

- 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.

Locations

In some cases, an enterprise might span several buildings, cities, or countries. In these situations, you might deploy a package to a location rather than to individual workstations and servers. Then, a secondary deployment server at each location can deploy the package to the workstations and servers at that location.

The larger your enterprise, the more you can benefit from creating and deploying to locations. If you use multitier deployment to deploy packages to remote locations, the concept of locations is crucial.

In JD Edwards EnterpriseOne, a *location* is essentially a user-defined group of machines, databases, and environments. In some cases, the location is an actual physical location that is connected by a WAN, such as when you have remote offices that are geographically separate from your main office. For example, a location might be a floor in your office building, a separate building on the corporate campus, a branch office across town, or a facility in another city.

After you create a new location, you can add workstations and servers for that location by defining the machine names that are associated with that location.

The topmost location that appears when you launch the JD Edwards EnterpriseOne Deployment Locations Application program (P9654A) is the base location. You cannot change or remove this base location, but you can create or revise locations that are subordinate to it.

When you create a location that is subordinate to another location, the original location is the parent location, and its subordinate location is the child location. For example, if you have a location called Seattle and then create a location called Redmond that is subordinate to Seattle, Seattle is the parent location and Redmond is the child location.

Deployment Groups

You can create a deployment group based on department, team, or function. For example, you might have an administration group, a testing group, a production group, and so on.

Package deployment groups are particularly useful in large enterprises in which scheduling a package for deployment to several individual workstations is very time consuming. In these environments, you can deploy packages much more quickly when you use deployment groups.

A group can contain a subgroup (a group within a group). For example, you might have a group called Quality Assurance that is a subgroup of the larger Development group.

You can help the person who builds and schedules packages by creating easily identifiable names for deployment groups. For example, for a group that includes quality assurance specialists who are responsible for testing, name the group Testing, rather than Green Team.

See Also

Chapter 9, "Setting Up Multitier Deployment," page 173

Prerequisites

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

Select one of these ways to populate the F9650 table:

- Manually

For a machine that no user has ever used to sign in to JD Edwards EnterpriseOne, use the JD Edwards EnterpriseOne Deployment Locations application (P9654A) to manually enter a record in the F9650 table.

- Automatically

  The system automatically creates a record in the F9650 table when a user on a new machine signs in to JD Edwards EnterpriseOne for the first time. (The system also automatically updates existing records in the F9650 table each time a user signs in to the workstation.)

The simplest way to populate the F9650 table is to have all users on new machines sign in. In cases in which you need to deploy a package before the users can sign in, you must manually enter machine information. The JD Edwards EnterpriseOne Deployment Locations application enables you to perform this task.

In addition to defining workstations, you can also use the JD Edwards EnterpriseOne Deployment Locations application to enter or revise definitions for these machines:

- Deployment Server
- Enterprise Server
- Data Server
- Java Application Server
- Windows Terminal Server
- Crystal Enterprise Web Server
- Crystal Enterprise CMS
- Business Services Server

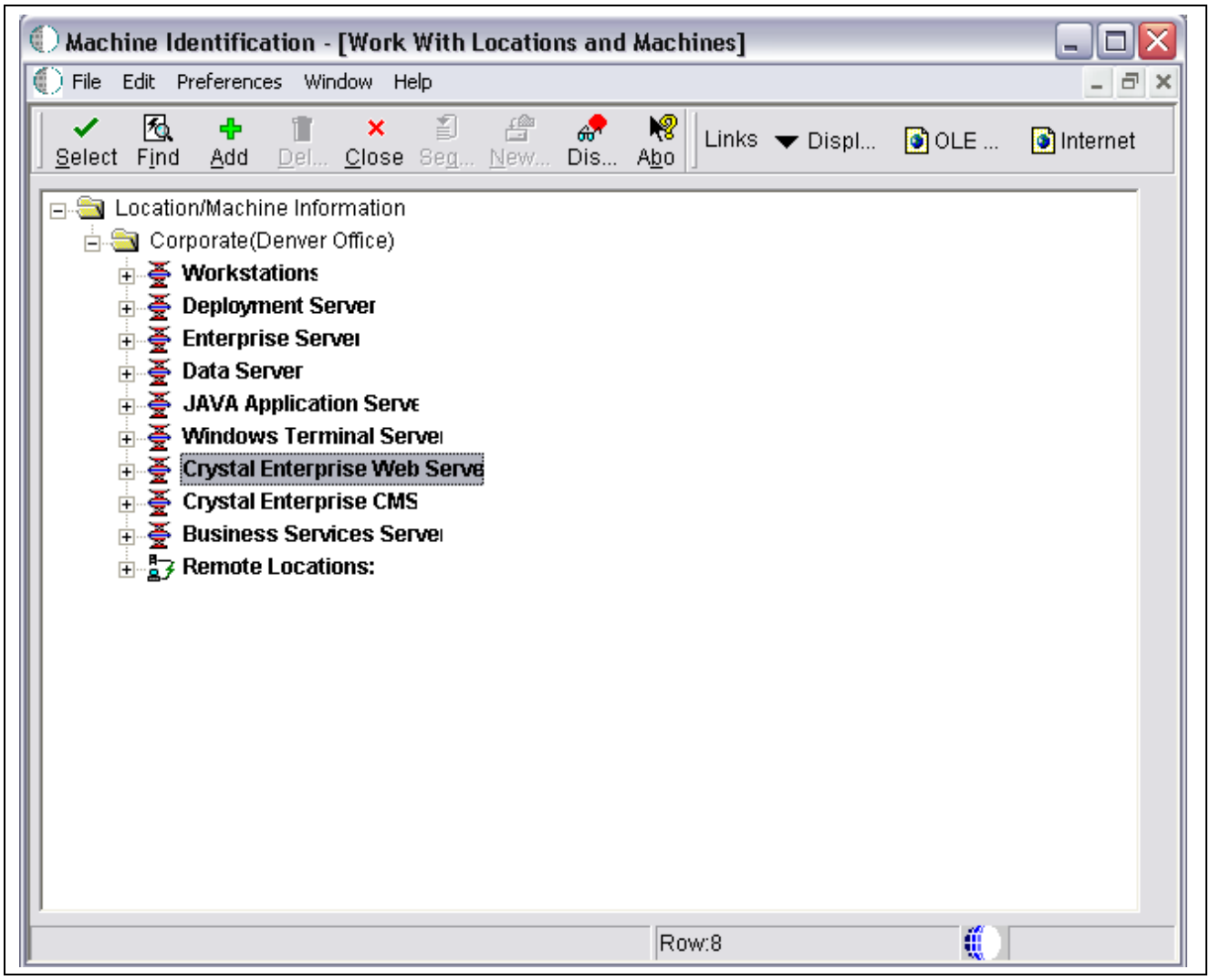
You can enter or revise definitions for these machines in multiple locations, including remote locations.

Forms Used to Define Deployment Parameters

Form Name	FormID	Navigation	Usage
Work with Locations and Machines	W9654AA	Package and Deployment Tools (GH9083), Machine Identification Click Add to add a new location or machine, or click Select to revise an existing location or machine.	Create or revise deployment locations or machines.
Deployment Group Revisions form	W9652AB	Package and Deployment Tools (GH9083), Machine Group Identification Click Add to add a new deployment group, or click Select to revise an existing group.	Create or revise package deployment groups.

Defining Machines

Access the Work with Locations and Machines form.



Work With Locations and Machines Form

1. Highlight the type of machine you would like to create.
2. Click Add.
3. Enter the appropriate values for the type of machine you are creating.

Workstation

Deployment Server Name

Enter the name of the specific server that is being used for deployment.

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

Deployment Server

Primary Deployment Server

Specify whether a deployment server is the primary deployment server for a specific location.

If you have set up a primary deployment server, you cannot access the Primary Deployment Server field when you define a new deployment server. You can

change the value in this field only when you revise the primary deployment server definition or when you change the primary deployment server to a secondary server. In this case, you can specify a different server as the primary deployment server.

Server Share Path Enter the shared directory for this path code. The objects that are stored on a file server will be found in this path.

Enterprise Server

Port Number Identify the port for a given instance of JD Edwards EnterpriseOne. Because the jde.ini file controls the port to which a workstation will connect, for workstations this port number is for reference only.

Logical Machine Name Enter the logical machine name that is assigned to this unique machine and port. A machine can be a workstation. Because you can have more than one instance of JD Edwards EnterpriseOne 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.

The logical machine name should represent the release and purpose of the machine, such as Financial Data Server-E900 or Distribution Logic Server-E900.

Database Type Enter the type of database.

Server Map Data Source Enter the name that identifies the data source.

Installation Path Enter the path on which JD Edwards EnterpriseOne is installed.

Deployment Server Name Enter the name of the specific server that is being used for deployment.

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

Server Availability This field is visible only in Update mode. Use this field to reset the enterprise server status if a package deployment failed.

Data Server

Data Source Type Enter the type of database.

HTML Server

Protocol Specify the method of communication (for example, http).

Server URL Enter the URL to the web server.

Http Port Enter the port number of the web server.

Default Login Enter the login path.

Installation Path Enter the path on which JD Edwards EnterpriseOne is installed.

Deployment Server Name Enter the name of the specific server that is being used for deployment.

Crystal Enterprise Web Server

Port Number Enter the port number for the Crystal Enterprise web server.

Crystal Enterprise CMS

User ID Enter the user name with which JD Edwards EnterpriseOne will connect to Crystal Enterprise for the purpose of running scheduled reports. This user, along with the associated password, must identify a valid Crystal Enterprise user with the necessary authority to execute reports.

Password Enter the password that JD Edwards EnterpriseOne will use to connect to Crystal Enterprise for the purpose of running scheduled reports. This password, along with the associated user ID, must identify a valid Crystal Enterprise user with the necessary authority to execute reports.

Password -Verify Reenter the password.

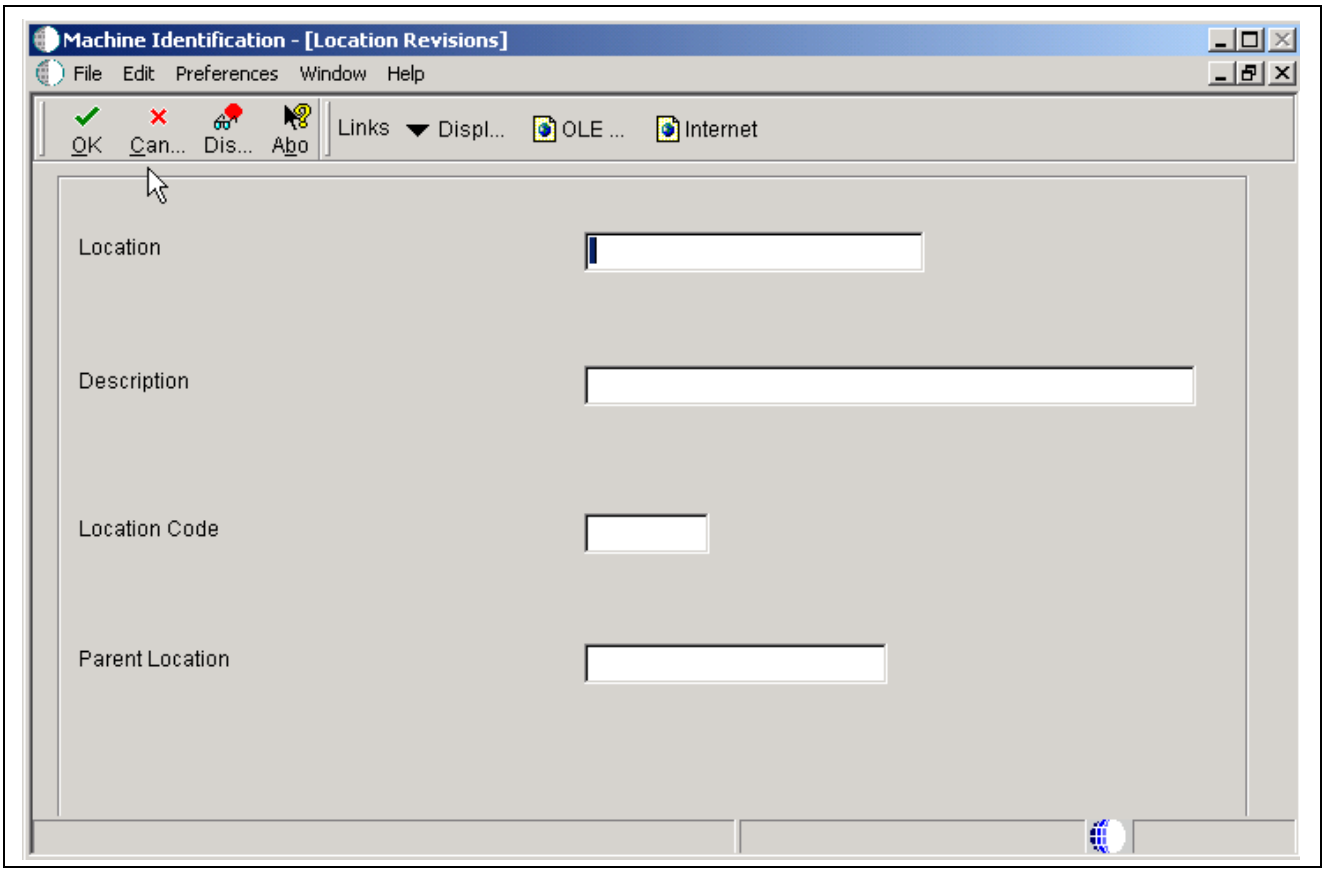
Business Services Server

The Business Services Server cannot be added through this application. You must use Server Manager to add a new Business Services Server.

See the *JD Edwards EnterpriseOne Tools Release 8.98 Server Manager Guide* on the Update Center.

Defining Locations

Access the Location Revisions form by clicking Add on the Work with Locations and Machines form.

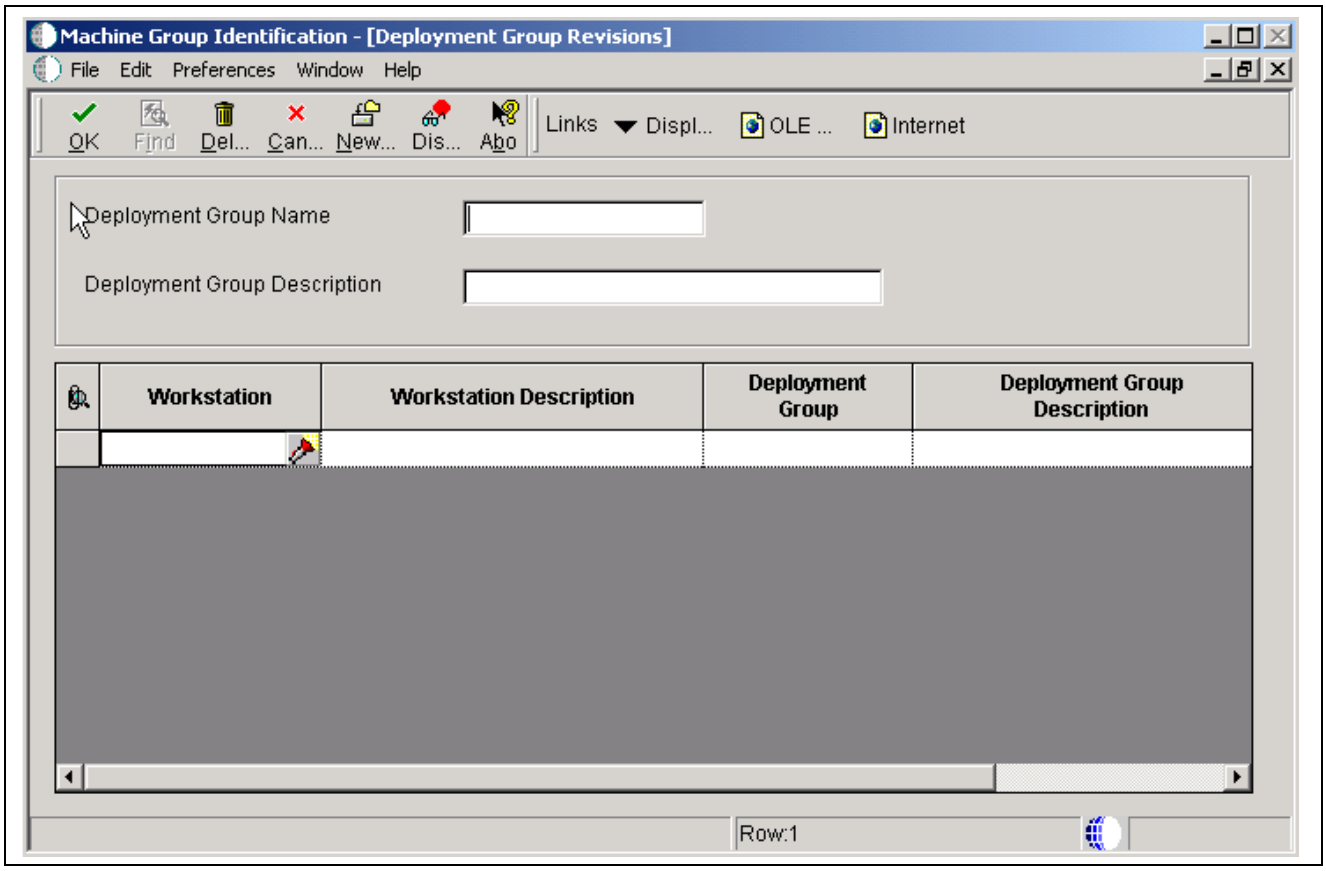


Location Revisions form

Location	Enter the name of the deployment location.
Location Code	Represent the current location for system deployment.
Parent Location	Enter the name of the parent location.

Defining Package Deployment Groups

Access the Deployment Group Revisions form.



Deployment Group Revisions form

Deployment Group Name Enter a profile to use to classify users into groups for system security purposes. You use group profiles to give the members of a group access to specific programs.

Some rules for creating a profile for a user class or group are:

- The name of the user class or group must begin with an asterisk (*) so that it does not conflict with any system profiles.
- The User Class/Group field must be blank when you enter a new group profile.

Deployment Group Description Enter the description for the selected deployment group.

Workstation Specify the name of the machine on the network .

Workstation Description Enter a user-defined name or remark.

Deployment Group Specify a group that is defined to be part of a parent group.

Revising Package Deployment Groups

Access the Deployment Group Revisions form.

Note. When you revise an existing group, you cannot change the group name, but you can change the description.

1. To add to the group, select the last row (the empty one) and enter the name of the workstation or deployment group to which you want to add members.
2. Type the name in the Workstation field or the Deployment Group field, or use the search button for those fields.

   When you use the search button for the Workstation field, the Machine Select form appears. When you use the search button for the Deployment Group field, the Deployment Group Search form appears.

Working with Package Deployment

This section provides an overview of the JD Edwards EnterpriseOne Deployment Director and discusses how to:

- Schedule a package for deployment.
- Revise deployment options.
- Activate a scheduled package.
- Install a scheduled package.

Understanding the Deployment Director

After you define and build a package, use the JD Edwards EnterpriseOne Deployment Director program (P9631) to schedule the package for deployment to individual workstations, deployment servers, or enterprise servers. On the specified deployment date, users who are scheduled to receive the package can load the package when they sign in to JD Edwards EnterpriseOne.

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.

The JD Edwards EnterpriseOne Deployment Director program (P9631) simplifies and expedites the process of scheduling and deploying built packages to workstations and servers. The director displays a series of forms that enable you to specify the package that you want to deploy, the deployment destinations, and the deployment time.

After specifying the package that you want to deploy, you specify any of these 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 select one of these methods.

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

In addition, you can specify a *push installation,* which means that the package can be deployed from the deployment server to the workstations that you specify, without requiring any interaction from the user.

Note. Mandatory and Push Installation options are applicable to client packages only.

The JD Edwards EnterpriseOne Deployment Director requires that JD Edwards EnterpriseOne already be loaded on the workstation, unless you are using push installation. You can schedule a new full package to replace the existing package, or an update package to be merged with the existing package on the workstation.

The JD Edwards EnterpriseOne Deployment Director uses these tables:

- F9650
- F9651
- F9652
- F9653
- F9654
- F98825
- F988251
- F98826
- F9603
- F96031

This table summarizes the function of each form in the JD Edwards EnterpriseOne Deployment Director:

Package Deployment Director form	View this form for a description of the JD Edwards EnterpriseOne Deployment Director.
Package Selection form	Use this form to find and select the package that you want to deploy.
Package Deployment Targets form	Use this form to specify the destination for the package. You can select individual client workstations, deployment servers, and enterprise servers, or you can deploy the package to a deployment group or location.
Package Deployment Attributes form	Use this form to enter the date and time that you want to deploy the package. Also specify whether the package is mandatory (that is, it must be installed by every package recipient) and whether you want to use push installation to deploy the package.
Deployment Client Workstation Selection form	Use this form to select each of the client workstations that will receive the package.
Deployment Server Selection form	Use this form to select each of the deployment servers that will receive the package.
Enterprise Server Selection form	Use this form to select each of the enterprise servers that will receive the package.
Deployment Location Selection form	Use this form to specify the deployment location that will receive the package.

Deployment Groups Selection form	Use this form to specify the deployment groups whose members will receive the package.
Build Selection form	For multitier deployment, use this form to specify the server or client package that you want to deploy to the destination deployment server.
Work with Package Deployment form	Use this form to review and revise the locations and package recipients that you entered through the JD Edwards EnterpriseOne Deployment Director.

Using the Deployment Director

After you have assembled and built the package, defined all machines, and verified the deployment groups, use the JD Edwards EnterpriseOne Deployment Director to specify package recipients and schedule the package for deployment.

Throughout the deployment process, you can select to either proceed to the next form or return to the previous form. Also, regardless of where you are in the process, you can cancel it.

When you schedule a package for deployment to a machine rather than a deployment group or location, you can schedule to deploy the package to client workstations, deployment servers, enterprise servers, or a combination. The forms that appear vary depending on your selection. For example, if you indicate that you want to schedule a package for deployment to client workstations and a deployment server, the forms for selecting specific workstations and deployment servers appear. If you schedule a package for deployment only to client workstations, the server selection form does not appear.

When you access the JD Edwards EnterpriseOne Deployment Director, the Work with Package Deployment form enables you to view deployed package information by either machines, deployment groups, locations, or packages.

Depending on your display selection, the tree displays different information when you expand it. This list describes the information that appears as you expand the tree level by level:

- Machines

  Level One: Client Workstation, Deployment Server, Enterprise Server, and Business Service Application Server headings.

  Level Two: Specific machines under each of these three headings.

  Level Three: Specific packages that are deployed to the machine, if any.

- Deployment Groups

  Level One: Specific groups.

  Level Two: Members of those groups.

  Level Three: Specific packages that are deployed to the group member.

- Locations

  Level One: Specific locations.

  Level Two: Client Workstation, Deployment Server, Enterprise Server, Business Service Application 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 that are 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 that are deployed to each machine, if any.

- Packages

  Level One: Package names.

  Level Two: Client Workstation, Deployment Server, Enterprise Server, and Business Service Application 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.

Activating Scheduled Packages

After you successfully define a package deployment, you must activate the package so that it is available for installation using the JD Edwards EnterpriseOne 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 JD Edwards EnterpriseOne Workstation Installation program.

In some situations, you might need to control which packages are available for installation. If, for example, you have a package that is for the testing group only, you would want to make that package inactive so that it is not available for installation through the JD Edwards EnterpriseOne Workstation Installation program. Instead, you can use JD Edwards EnterpriseOne Deployment Director program (P9631) to schedule this package for deployment to the members of the testing group.

Installing a Scheduled Package

When users receive a package, they can select to install it when they sign in to JD Edwards EnterpriseOne on or after the scheduled deployment date.

If the package is mandatory, users cannot access the system until they load the package.

If the package is optional, users can decline the package or postpone the installation until later. If they decide to postpone the installation, the software launches, and they will be given the opportunity to install the package the next time that they sign in.

If a package that is scheduled for push installation fails to load for some reason (such as if the power to the workstation was turned off during the time that the package was scheduled to deploy), that package will be included in the list of available packages when the user signs in.

See Also

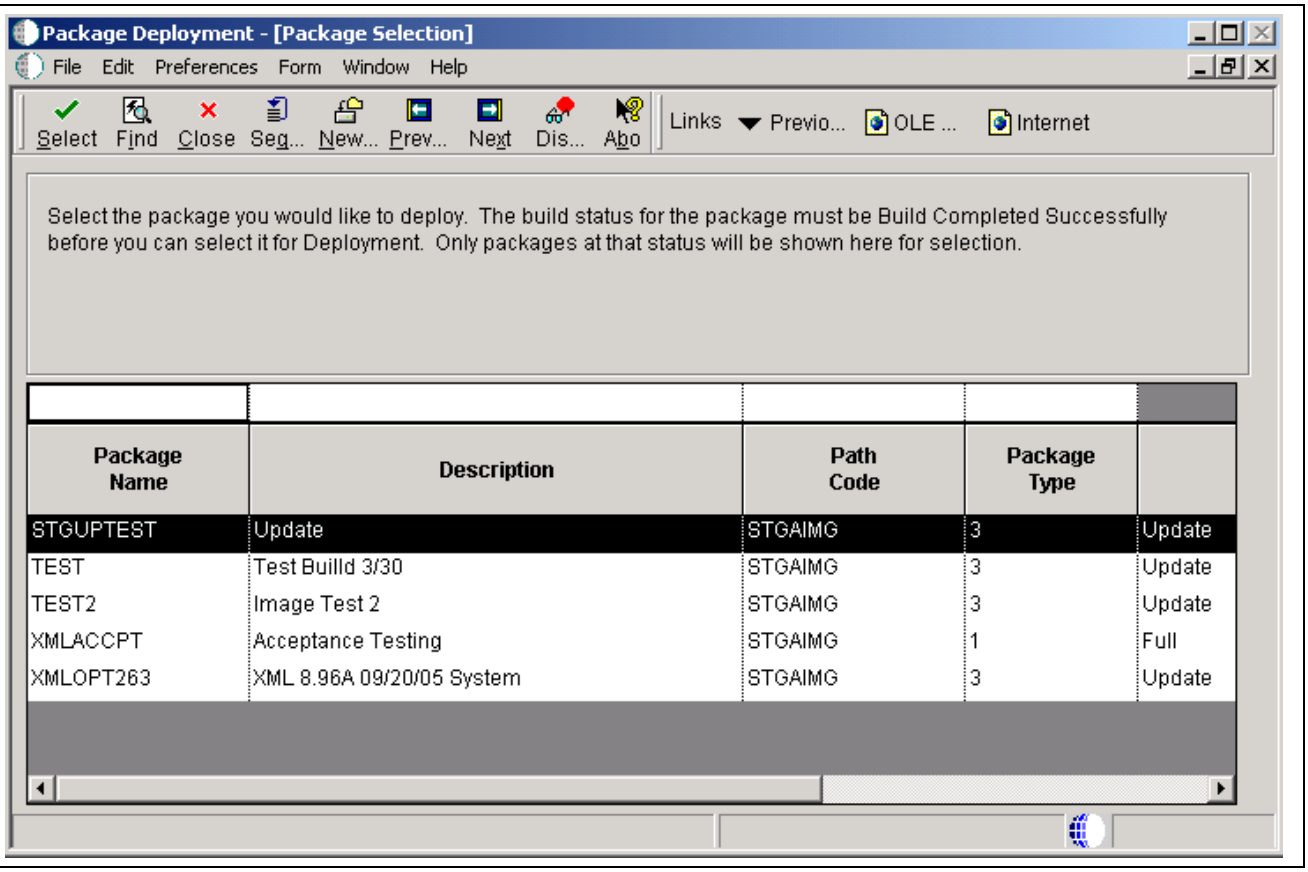
Chapter 7, "Deploying Packages," Scheduling the JD Edwards EnterpriseOne Push Installation Batch Application, page 155

Forms Used to Work with Package Deployment

Form Name	FormID	Navigation	Usage
Work with Package Deployment	W9631J	Package and Deployment Tools (GH9083), Package Deployment	Select the package to deploy and review your selections.
Package Selection	W9631C	Package and Deployment Tools (GH9083), Package Deployment. Click Add on the Work with Package Deployment form to launch the Deployment Director. Click Next.	Select the package to deploy.
Package Deployment Targets	W9631B	On Package Selection, click Next.	Select the types of machines on which to deploy the package.
Package Deployment Attributes	W9631D	On Package Deployment Targets, click Next.	Select the type of installation and the time.
Deployment Client Workstation Selection	W9631F	On Package Deployment Targets, select Client Workstation and click Next until the Deployment Client Workstation Selection form appears.	Select the workstations to which the package will be deployed.
Deployment Server Selection	W9631G	On Package Deployment Targets, select Deployment Server and click Next until the Deployment Server Selection form appears.	Select the Deployment Servers to which the package will be deployed.
Build Selection	W9631N	On Package Deployment Targets, click Next until the Build Selection form appears.	Select the server package build to deploy to the destination deployment server.
Enterprise Server Selection	W9631E	On Package Deployment Targets, select Enterprise Server and click Next until the Deployment Server Selection form appears.	Select the Enterprise Servers to which the package will be deployed.
Server Package Deployment Properties Revisions	W9631M	Package and Deployment Tools (GH9083), Package Deployment. Select Machines, and click Find to display information according to machine name. Find and select the deployed package for which you want to modify the options, and then select Properties from the Row menu.	Revise server package deployment options.

Scheduling a Package for Deployment

Access the Package Selection form.



Package Selection form

1. Select the package that you want to deploy, and then click Next.
2. On the Package Deployment Targets form, select any of these options to indicate the type of machines to which you want to deploy the package, and then click Next:
   - Client Workstation
   - Deployment Server
   - Enterprise Server
   - Business Services Server

     See Chapter 8, "Working with Packages for Business Services," page 163.
   - Deployment Group
   - Locations
3. On Package Deployment Attributes, complete these fields:
   - Mandatory Installation
   - Enable Push Installation
   - Date/Time

4. If you want to deploy the package using push installation, which pushes the package to workstations from the deployment server, select the Enable Push Installation option, and then click Next.

   If you are deploying to workstations, the Deployment Client Workstation Selection form appears. If you are not deploying to workstations, bypass the next step.

5. Find and select the workstations to which you want to deploy the package, and then click Next.

   Select a workstation by double-clicking in its row header. A check mark appears in the row header for each workstation that you select.

   If you are deploying to a deployment server, the Deployment Server Selection form appears. If you are not deploying to a deployment server, bypass the next step.

6. Find the deployment server to which you want to deploy the package, and then click Next.

   Select a server by double-clicking in its row header. A check mark appears next to each server that you select.

7. On the Build Selection form, select the server package build that you want to deploy to the destination deployment server, and then click Close.

8. 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, bypass the next step.

9. Find and select the enterprise server to which you want to deploy the package, and then click Next.

   Select a server by double-clicking in its row header.

   Note. You can deploy an update package only to servers that have the full parent package deployed.

10. On Work with Package Deployment, review your deployment selections.

11. To change any of the selections, click Prev to return to the appropriate previous form.

12. When you are finished reviewing and changing the deployment selections, click End.

13. If you are deploying a server package, find and select the server package on the Work with Package Deployment form, and then select Deploy from the Row menu.

After you schedule the package for deployment, at the specified time on the date that you specified, the package deploys to workstations. This package becomes available to the user when the user signs in.

If you are using push installation, the package automatically installs at the time that you specify in Oracle's JD Edwards EnterpriseOne Schedule Jobs program (P91300).

To schedule a package for deployment to a deployment group or location:

1. On the Package Selection form, select the package that you want to deploy, and then click Next.

2. On the Package Deployment Targets form, select either Deployment Group or Locations, and then click Next.

3. On the Package Deployment Attributes form, complete these fields:

   - Mandatory Installation
   - Enable Push Installation
   - Date/Time

4. If you want to deploy the package using push installation, which pushes the package to workstations from the deployment server, select the Enable Push Installation option.

5. Click Next.

   If you are deploying to a deployment group, the Deployment Groups Selection form appears. If you are deploying to a location, bypass the next step.

6. Find and select the deployment group that you want to receive the package, and then click Next.

   Select a group by double-clicking its row header.

7. If you are deploying to a location, the Deployment Location Selection form appears. Bypass the next step if you are deploying to a deployment group.

8. Find and select the deployment location that you want to receive the package, and then click Next.

   To select a location, double-click the row header.

9. On the Work with Package Deployment form, review the deployment selections.

10. To change any of the selections, click Prev to return to the appropriate previous form.

11. When you are finished reviewing or changing the deployment selections, click End.

12. If you are deploying a server package, find and select the server package on the Work with Package Deployment form, and then select Deploy from the Row menu.

After you schedule the package for deployment, at the specified time on the date that you specified, the package deploys to workstations. This package becomes available to the user when the user signs in.

If you are using push installation, the package automatically installs at the time that you specify in the JD Edwards EnterpriseOne Schedule Jobs program (P91300).

See Also

See the following topics in the JD Edwards EnterpriseOne Package Management Guide:

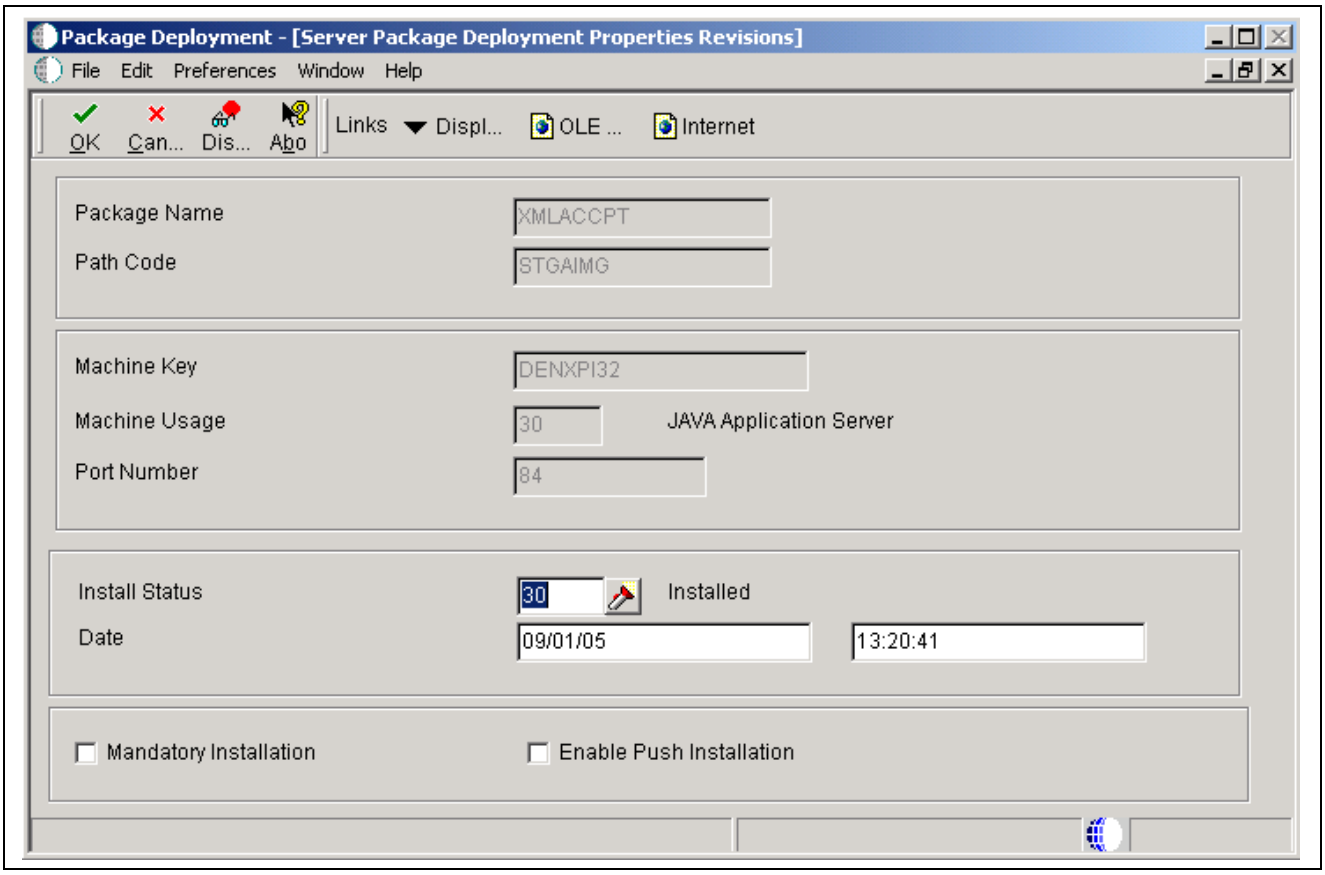
Chapter 7, "Deploying Packages," Scheduling the JD Edwards EnterpriseOne Push Installation Batch Application, page 155

Chapter 7, "Deploying Packages," Scheduling a Package for Push Installation, page 154

Chapter 7, "Deploying Packages," Understanding the Deployment Director, page 135

Revising Deployment Options

Access the Server Package Deployment Properties Revisions form.



Server Package Deployment Properties Revisions form

Package Name

Enter a name for the package.

A package describes the location on the server where components that you want to deploy to workstations or servers reside. Two package types are available:

Full: Contains the full suite of applications (all specifications).

Update: Objects contained in this type of package are loaded after the workstation or server receives the package and the user signs in to the system. 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 that are specified by the system administrator.

With the exception of just-in-time applications that are included in an Update package, all packages are a snapshot at a point in time of the central objects for a particular path code. Just-in-time applications are dynamic, not built.

Path Code

Enter the path code.

The path code is a pointer to a set of objects and is used to keep track of sets of objects and their locations.

Mandatory Installation

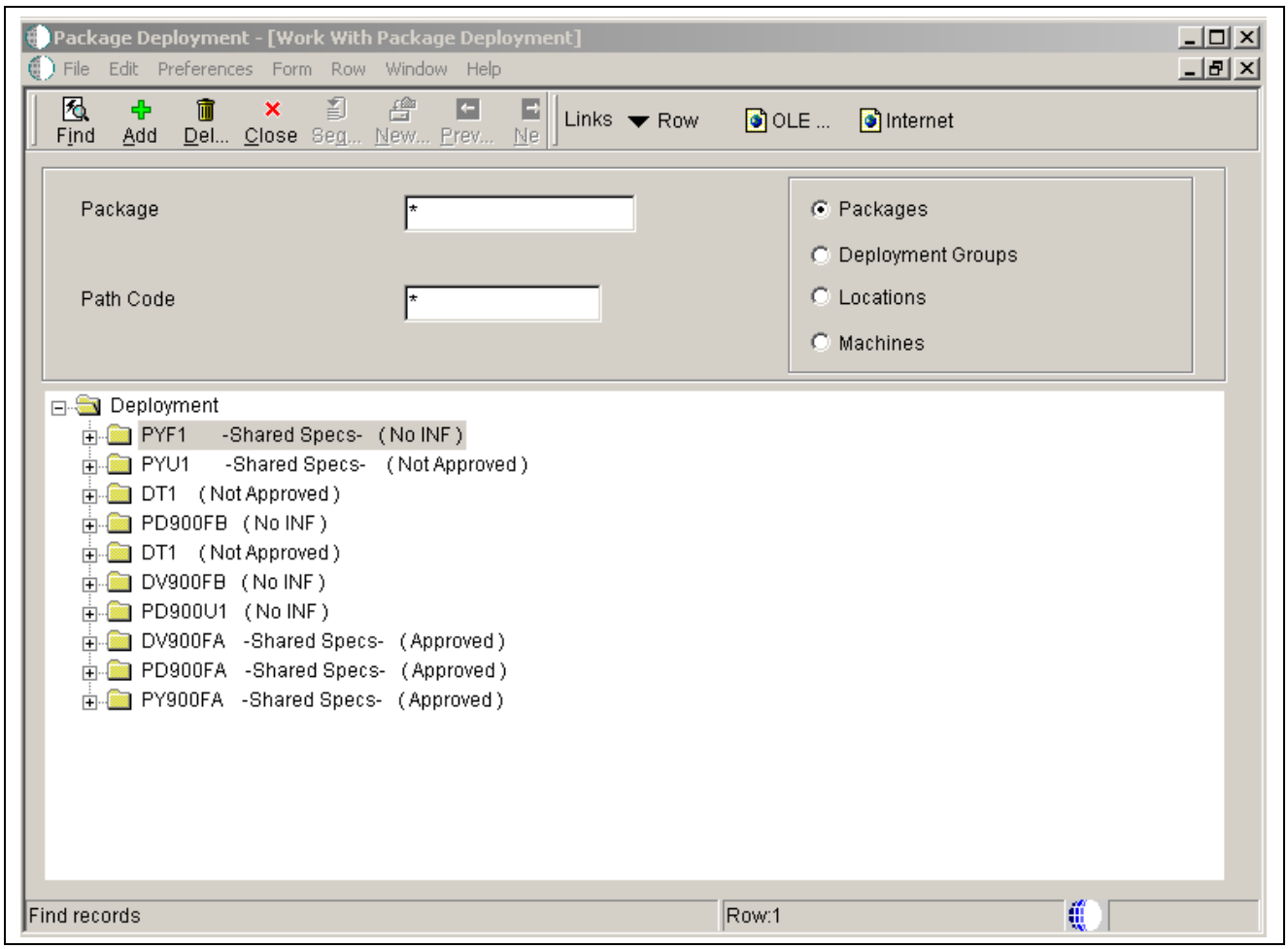
Indicate whether the package is mandatory or optional.

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	Select this option to enable the package to be installed through push installation.
Date	Enter a date to deploy updated objects to the listed machine.

Activating the Scheduled Package

Access the Work with Package Deployment form.



Work with Package Deployment form

1. Click Find.
2. Select from the list the packages that you want to activate or inactivate.

   Alternatively, you can enter the package name in the Package field.
3. Select Active/Inactive from the Row menu.

Installing a Scheduled Package

1. Sign in to JD Edwards EnterpriseOne.

   When you are scheduled to receive a package, Oracle's JD Edwards EnterpriseOne Just-In-Time Installation program launches and the Scheduled Packages form appears.

2. Perform one of these steps:

   - To load the package immediately, bypass to step 3.
   - To decline the package permanently, select Decline from the Row menu.
   - To list all items in the package, select 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 the system until you load the package.

3. To load the package, select one or more packages that you want to install and click Select.

   The JD Edwards EnterpriseOne Workstation Installation program loads the package. If you selected more than one package, the program installs them sequentially. When the installation is complete, the software launches.

Deploying a Server Package

This section provides overviews of server package deployment and deployment to web servers, lists prerequisites, and discusses how to deploy a server package.

Understanding Server Package Deployment

The process for deploying a server package is nearly identical to that for deploying a package to a workstation. In both cases, you need to assemble, define, build, and schedule the package for deployment by using the JD Edwards EnterpriseOne Package Assembly (P9601), Package Build Director (P9621), and Deployment Director (P9631) programs.

After you schedule a server package for deployment, you must complete an additional step to launch the batch program that enables you to deploy to servers. You must perform this task whenever you deploy a package to an enterprise server or deployment server.

Important! 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, the system queues any jobs that are submitted to the enterprise server and runs them as soon as the installation finishes. For this reason, you should schedule enterprise server packages to be deployed after hours to minimize impact on users. (Before you deploy a package to an enterprise server, verify that the services have been started and no UBEs are active.)

To further minimize impact on the network and users, if the development environment is on the same enterprise server as the production environment, consider preventing developers from moving their own objects through server packages. Instead, require that an administrator perform this function.

To deploy a server package, select Deploy from the Row menu on the Work with Package Deployment form. This is the same function that you use to deploy packages to deployment servers during multitier deployment.

The system determines which of the batch programs to call, based on what is currently selected on the Work with Package Deployment form when you select Deploy from the Row menu:

- If a specific deployment server is selected, the system launches the Multi Tier Deployment batch program (R98825C).
- If the deployment server folder is selected, the system launches the Multi Tier Deployment batch program for every deployment server that has a package scheduled.
- If a specific enterprise server is selected, the system launches the Enterprise Server Deployment batch program (R98825D).
- If the Enterprise Server folder is selected, the system launches the Enterprise Server Deployment batch program for every enterprise server that has a package scheduled.
- If a specific package is selected, the system launches the Multi Tier Deployment batch program, and then the Enterprise Server Deployment batch program for the selected package.
- If you sort by packages and the Deployment folder is selected, the system launches both the Multi Tier Deployment batch program and Enterprise Server Deployment batch programs for all packages.
- If a specific workstation or the Workstations folder is selected, the Deploy option is unavailable.

When the system launches a batch program for all servers or all packages, deployment does not occur unless the package has been previously scheduled for a specific server. A full package can be deployed to all servers. However, an update package can be deployed only to servers that already have the parent package deployed. Also, update packages cannot be deployed if the parent package is an inactive or not-deployed full package.

When the system launches the batch program to deploy a package to an enterprise server, the batch process:

1. Verifies that the enterprise server deployment location is the same as the Microsoft Windows 2000 client submitting the package.
2. Changes the enterprise server status to *Pre Deploy*.

   This is done by changing the `MDMCHRCDNM` column to *50* in the F9651 table.
3. Waits for five minutes.
4. Sends lock messages to all the selected enterprise servers.
5. Once the servers are locked, the batch process marks them as unavailable.

   This is done by changing the `MDMCHRCDNM` column to *10* in the F9651 table.
6. Copies the BSFN executables from the package location to the live path code location.
7. Sets the spec.ini file to the new package and spec data source.
8. Sends unlock messages to all the enterprise servers.
9. Marks the servers as available.

   This is done by changing the `MDMCHRCDNM` column to *30* in the F9651 table.
10. Updates the F96511 table with the new package and spec data source information.

    This information is used by the web servers.

Note. If the batch process fails to get a lock on all of the selected enterprise servers, it will not deploy the package.

If a package deployment fails, the servers are marked as unavailable. The system administrator might have to restart the services on the enterprise server to make them available again. He can also reset the status with Oracle's JD Edwards EnterpriseOne Machine Detail application (P9654A).

A deployed package can be deployed multiple times to the same or different servers.

A server package with the specs built in a shared mode can be deployed to a web server. This process of deploying to the web server is automatic and does not require any end user intervention. The web servers pool the package information from the JD Edwards EnterpriseOne logic server. It compares the package manifest from the spec tables to the one in its serialized database and makes the necessary updates.

During server package deployment, the business function (BSFN) dll's, SRVPGMs, .so objects, or .sl objects of the live package are replaced by the objects from the built package. However, if a deployment fails, you may have a mismatched set of BSFNs and specs. With 8.96 JD Edwards EnterpriseOne clients, you can back up the existing BSFN objects. If the deployment fails, you can restore the BSFN objects. The option to back up the live BSFN objects before deployment can be enabled through the Build Settings within Server Manager.

For an AS/400, the BSFN objects in PY900 are copied into the $PY900 library. For Microsoft Windows 2000 and UNIX, the BSFN and spec objects in PY900 are copied to the PY900_BACK folder. Clients can restore BSFN objects by copying the objects from the backup location to the live folder. They can restore the specs by changing the package name to the previous package in the spec.ini file.

Understanding Deployment to Web Servers

Web servers have the ability to determine which package is to be deployed on the default enterprise server and generate serialized objects on demand. The web servers also have the capability to compare the contents of a package with that of a new package and make the necessary adjustments, such as deleting obsolete serialized objects. This is done by using a package manifest.

A package manifest is a spec record that is created by the package build process. The manifest describes the package and its contents. The serialized object generator compares the manifest from the deployed package on the enterprise server to one that is created during the generation process and makes the appropriate changes.

The process for deploying packages to web servers is:

1. The web servers check the F9651 and F96511 tables every five minutes for any change in the package.
2. The five minute interval is changed to five seconds once the enterprise server is in a *Pre Deploy* state.
3. All JD Edwards EnterpriseOne users are prevented from running any applications once the enterprise server is in a *Locked* state.
4. The web server compares the package manifest in the F98770 on the enterprise server with the one in its serialized object database after the package is deployed and the enterprise server is in an *Available* state.
5. The web server synchronizes the serialized object database with the deployed package.

   The contents of the serialized object database are deleted for a new full package deployment. Only the objects in the update package are deleted for an update package deployment.

Note. Deploying server packages to web servers is supported only in shared spec server packages.

You can have only one path code and one package per Java node. Serialized objects should not be shared with nodes running dissimilar packages and path codes.

Prerequisites

Before you complete the tasks in this section:

- Assemble the server package.
- Define the server package.
- Build the server package.

- Schedule the package for deployment to the appropriate server.

Form Used to Deploy Server Packages

Form Name	FormID	Navigation	Usage
Work with Package Deployment	W9631J	Package and Deployment Tools (GH9083), Package Deployment	Select the package to deploy and review your selections.

Deploying a Server Package

Access the Work with Package Deployment form.

1. Locate the server package that you want to deploy.

   Alternatively, select the enterprise server or, if the package is scheduled to deploy to more than one server, the Enterprise Server folder.
2. Select Deploy from the Row menu.
3. On Report Output Destination, select On Screen.
4. Click OK.

Using Push Installation

This section provides an overview of Push Installation and discusses how to:

- Prepare the enterprise server for push installation.
- Prepare workstations for push installation.
- Install the JD Edwards EnterpriseOne Listener.
- Install the JD Edwards EnterpriseOne Listener using silent installation.
- Stop and uninstall the JD Edwards EnterpriseOne Listener.
- Schedule a package for push installation.
- Schedule the JD Edwards EnterpriseOne Push Installation batch application.
- Run the JD Edwards EnterpriseOne Package Installation Results report.

Understanding Push Installation

Push installation is the only deployment method that provides automatic and unattended package deployment. This means that the system administrator can deploy a package (or several packages) to a workstation or group without requiring any action from workstation users.

For example, an administrator might schedule a package to deploy to a particular group after hours. When members of that group report to work the following morning, that package is available for immediate use.

Push installation is particularly useful in situations in which you need to quickly deploy packages with a minimum of intrusion or impact upon your normal production and development routines. By planning and scheduling package deployment judiciously, administrators can also minimize the impact upon network performance that can accompany large numbers of package deployments. Administrators can also use push installation to install the software on a workstation for the first time. This ability can greatly minimize downtime and provide maximum deployment flexibility.

During push installation, package contents are pushed from the deployment server to the workstation. In contrast to push installation, the JD Edwards EnterpriseOne Workstation Installation program pulls package contents from the deployment server to the workstation. Installations that are set up to use JD Edwards EnterpriseOne Deployment Director (P9631) for scheduled packages that are not push enabled also pull packages.

The end result of the deployment is the same, regardless of whether package contents are pushed or pulled. However, the advantage of a push installation is that no action is required from the workstation user other than to leave the workstation turned on during the time when the package is scheduled to deploy.

For an update package that contains program specifications, the term *package contents* refers to specifications. For a full package or an update package that does not contain application specifications, *package contents* refers to objects.

Push Installation Process

This list summarizes the steps in the push installation process:

1. Install a push installation JD Edwards EnterpriseOne Listener program on each workstation that will receive pushed packages.

   Oracle's JD Edwards EnterpriseOne Listener monitors the progress of Oracle's JD Edwards EnterpriseOne Push Package Installation batch program (R98825) that runs on the server and performs functions such as monitoring installation status. The JD Edwards EnterpriseOne Listener can run as either a local service or a network service.

2. Schedule the package using the JD Edwards EnterpriseOne Deployment Director program (P9631).

   The JD Edwards EnterpriseOne Push Package Installation batch program reads the scheduling table and sends a message to the JD Edwards EnterpriseOne Listener on all workstations for which a package has been scheduled.

3. Use the JD Edwards EnterpriseOne Schedule Jobs program (P91300) to launch the batch program on the enterprise server.

   This program enables you to specify the job name, version, start date, start time, and recurrence.

4. At the specified start time, the JD Edwards EnterpriseOne Schedule Jobs program launches the JD Edwards EnterpriseOne Push Package Installation batch program (R98825), which initiates the package installation process from the deployment server.

   The JD Edwards EnterpriseOne Listener and the batch program interact during the process until installation is complete. Codes are passed from the JD Edwards EnterpriseOne Listener to the batch program to indicate the installation status (such as failed, successful, in progress, and so on).

5. When the installation finishes, the system sends an email message to the primary user of the workstation.

   This message indicates whether the installation was successful. Email notification works only if the package recipient is listed in the Machine Master table (F9650) and has an email address in the profile.

6. If the push installation fails for some reason (such as when the package recipient neglects to leave the workstation turned on), the installation status changes to Failed.

If you want to reschedule the installation, you must first delete the row with the failed job, and then schedule the job again.

If the push installation is not successful, when the user signs on, the standard scheduling screen appears. At this point, the user can either accept the mandatory package or quit the program.

Installing the JD Edwards EnterpriseOne Listener

When you install the JD Edwards EnterpriseOne Listener on the workstations in your enterprise, you specify whether to run a local service or a network service. If you run the service locally on the workstation, the user must be signed in to receive a package that has been scheduled for push installation. If you run the service on the network, the JD Edwards EnterpriseOne Listener runs as a network account and the user does not have to be signed in to receive a package through push installation. The network service must have an administrator's user ID.

The disadvantage of running the service on the network is that it can be difficult to administer for all users on the enterprise. For example, because the parameters of the JD Edwards EnterpriseOne Listener apply to every user on the network, push installation users must install to and from the same locations. One user could have the software on drive C and another user have the same release on drive D. Also, every time users change their sign—in passwords, the system administrator must update the JD Edwards EnterpriseOne Listener service with the new passwords for the service to work for those users. For these reasons, you should install the JD Edwards EnterpriseOne Listener locally on each workstation.

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

You can select from one of these ways to install the JD Edwards EnterpriseOne Listener on a workstation:

- Use a third-party software distribution system, such as the Tivoli Management Environment (TME10) Software Distribution System or the Microsoft System Management Server (SMS) software.
- Distribute an executable installation program (a setup.exe file) and the accompanying ancillary files using an intranet website or the World Wide Web.
- Use Windows logon scripts (a .bat file) to call a C program.
- Install from the World Wide Web.

Important! If the JD Edwards EnterpriseOne Listener is not installed and running on the workstation (or if the workstation is turned off), the push installation cannot occur. After you schedule a package, remind package recipients to leave their workstations on and the JD Edwards EnterpriseOne Listener service running, even during off-hours. If you set up the JD Edwards EnterpriseOne Listener to run as a local service, also remind users to remain signed in.

Installing the Listener Using Silent Installation

In some cases, installing the JD Edwards EnterpriseOne Listener on workstations using silent installation might be more convenient than standard installation. Typically, the administrator performs this task. Using this method, the administrator enters configuration settings for all workstations and distributes a batch file that automatically initiates the JD Edwards EnterpriseOne Listener installation the next time that the user signs in to the workstation.

The advantage of using silent installation to install the JD Edwards EnterpriseOne Listener is that the process is transparent to workstation users, and users are not required to enter configuration information or step through the installation process.

Forms Used to Use Push Installation

Form Name	FormID	Navigation	Usage
Scheduling Information	W91300A	Job Scheduler (GH9015), Schedule Jobs Select the time zone that applies to your setup and click Select. Click Add to enter a new job.	Schedule the Push Installation Batch Application.
Push Package Installation Results	NA	Package and Deployment Tools (GH9083), Push Package Installation Results	Verify the status of a pushed package.

Preparing the Enterprise Server for Push Installation

To set up the server for push installation, you must first install and configure the Microsoft Domain Name Service (DNS) that is included with the Microsoft Windows Server. If you have not yet set up a domain name service, you can install Microsoft DNS by clicking the Network button in the Control Panel, then selecting the Services tab, and then adding Microsoft DNS Server.

After you add Microsoft DNS, you must configure the DNS by specifying the domain name and servers.

UNIX and iSeries Considerations

In an environment that is configured for Dynamic Host Configuration Protocol (DHCP), a server must run Windows Server to resolve workstation addresses because the Windows server dynamically assigns them.

To enable name resolution, you need to configure the servers to resolve their IP address lookup through a Windows DNS server, which, in turn, must be configured to review the WINS database when DHCP is enabled in the network domain.

Configuring the servers in this way ensures that this process flow occurs during the push installation process:

1. From the host server on which the JD Edwards EnterpriseOne Push Installation batch program (R98825) runs, a business function attempts to retrieve the machine host address from the DNS server.
2. Because the DNS server does not contain IP addresses, it retrieves the address from the WINS server.
3. The WINS server returns the address to the DNS server.
4. The DNS server returns the address to the host server.
5. The host server finds the JD Edwards EnterpriseOne Listener on the client workstation and sends workstation installation information.
6. The workstation installation process starts.

Preparing Workstations for Push Installation

Before you can push an installation to a workstation, you must install a JD Edwards EnterpriseOne Listener on the workstation, which interacts with a business function that runs on the server. You must install this JD Edwards EnterpriseOne Listener on all workstations that you want to be enabled for push installation, regardless of whether you want to deploy packages to a machine on which JD Edwards EnterpriseOne is already installed or a machine on which you are installing JD Edwards EnterpriseOne for the first time.

The JD Edwards EnterpriseOne Listener runs continuously on the Windows workstation as a service (and on a Windows 95 machine as a pseudo-service), and administrators can monitor it using the Task Manager. The JD Edwards EnterpriseOne Listener communicates with the batch application on the server, receiving and sending messages during the installation process. The JD Edwards EnterpriseOne Listener also monitors the progress of the installation and saves the installation completion code.

Installing the Listener

This task describes how to install the JD Edwards EnterpriseOne Listener by launching an installation program (that is, a setup.exe program). You can distribute this program to users on the enterprise by using email to send them either the program or a shortcut, or by describing where the program is located on the server.

If you have a previous version of the JD Edwards EnterpriseOne Listener already installed, the installation program removes the previous version before copying the new version to the workstation.

Before you begin this task, close any applications that are currently open, and verify that the destination directory where you will be installing the JD Edwards EnterpriseOne Listener has sufficient disk space. You need approximately 2 MB of free disk space to install all of the Listener files and components.

To install the JD Edwards EnterpriseOne Listener on workstations:

1. Launch the installation program by double-clicking setup.exe.
2. On the first Client Listener Setup form, click Next.
3. On the second Client Listener Setup form, enter the release that you want to install through push installation in the Release field.
4. For the release that you selected, enter the full path name on the deployment server from which to initiate the installation in the Path name field.
5. Specify the drive on which you want to install the specified release in the Installation drive field.
6. Select the Uninstall option to uninstall existing versions of the software before installing a new full package.
7. Select the Autostart option to automatically start the JD Edwards EnterpriseOne Listener service whenever the workstation starts up.
8. Click Next to proceed to the next installation form.
9. Select one of these options:
   - Local
   - Network

   Unless the system administrator tells you to install the JD Edwards EnterpriseOne Listener on the network, click Local to install the JD Edwards EnterpriseOne Listener on the local workstation.
10. In the Folder field, specify the destination drive and folder in which the Listener files will reside.
11. Click Finish to complete the installation.

After you have successfully installed the JD Edwards EnterpriseOne Listener on the workstation, a small ear icon appears on the Windows taskbar, indicating that the JD Edwards EnterpriseOne Listener has been loaded. By right-clicking this icon, you can start or stop the JD Edwards EnterpriseOne Listener, or change the default parameter settings.

Installing the Listener Using Silent Installation

To install the JD Edwards EnterpriseOne Listener using silent installation:

1. Edit these settings in the listen_silent_setup.inf file that is included on the software CD:

File Setting	Description
ServiceType	Enter *Local* or *Network*, depending on where you want to run the JD Edwards EnterpriseOne Listener service.
WorkstationDirPath	Enter the location on the workstation where you want to install the JD Edwards EnterpriseOne Listener program and related files. For example, C:\Program Files\JDEdwards EnterpriseOne Client Listener.
Release	Enter the base release. Do not enter a cumulative update release.
InstallPath	Enter the location on the workstation where the software is installed. For example, D:\E900.
LaunchPath	Enter the deployment server name and the location from which the JD Edwards EnterpriseOne Client Workstation Installation program runs. For example, `\\` *server name* `\b9\OneWorld Client Install\setup.exe`.
AutoStart	Enter 1 to automatically start the JD Edwards EnterpriseOne Listener service when the workstation starts up. Enter 0 if you do not want to enable Autostart.
UninstallPackage	Enter 1 if you want to automatically uninstall previous versions before installing a new full package. Enter 0 if you do not want to enable automatic uninstall.

2. Create or modify a batch file to include the silent installation parameter `/s` for the ListenSetup.exe program.

   The batch file must reside in the same location as the ListenSetup.exe program.

   For example, your batch file might contain this line:

   ```
   start \\servername\E900\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 workstations.

4. Instruct users to restart their workstations to run the batch file and load the JD Edwards EnterpriseOne Listener using silent installation.

After workstation users have successfully installed the JD Edwards EnterpriseOne Listener, the Listener icon appears on the Windows taskbar. Users can click this icon to start and stop the JD Edwards EnterpriseOne Listener or change Listener settings.

Important! You cannot change the name of the Listener silent installation file that is shipped with the software. The file name must be listen_silent_setup.inf.

Stopping and Uninstalling the Listener

You can stop the JD Edwards EnterpriseOne Listener if you are certain that you do not want to use push installation to install packages. If you change your mind later, you can restart the JD Edwards EnterpriseOne Listener.

The easiest way to stop the JD Edwards EnterpriseOne Listener is to right-click the Listener icon on the Windows task bar and select Stop Listener.

Alternatively, you can stop the JD Edwards EnterpriseOne Listener using these steps:

1. Open the Control Panel.
2. Select Services.
3. Select JD Edwards EnterpriseOne Client Listener.
4. Click Stop.

To uninstall the JD Edwards EnterpriseOne Listener:

1. Open the Control Panel.
2. Select Add/Remove Programs.
3. Select JD Edwards EnterpriseOne Client Listener.
4. Click Remove All Components.

Scheduling a Package for Push Installation

After you have installed the JD Edwards EnterpriseOne Listener on the workstations, you can schedule a package for deployment.

The process for scheduling a package for push installation is identical to the process for scheduling a package using the JD Edwards EnterpriseOne Deployment Director program (P9631). When you schedule the package using this program, select the Enable Push Installation option on the Package Deployment Attributes form. If you do not select this option, the package will be deployed through normal scheduled deployment.

When you use the JD Edwards EnterpriseOne Schedule Jobs program (P91300), you can set recurrence, which determines how frequently the job runs until it finishes successfully. If you do not set recurrence, the job runs only one time. In the case of push installation, recurrence determines the interval of time between installation attempts. After the package is successfully deployed, the job ceases to run.

As with scheduling any other package for deployment, all machine names (that is, package recipients) must be defined in the Machine Master (F9650) table. This table is populated when users sign in. Alternatively, you can enter machine names manually using the JD Edwards EnterpriseOne Deployment Locations Application program (P9654A).

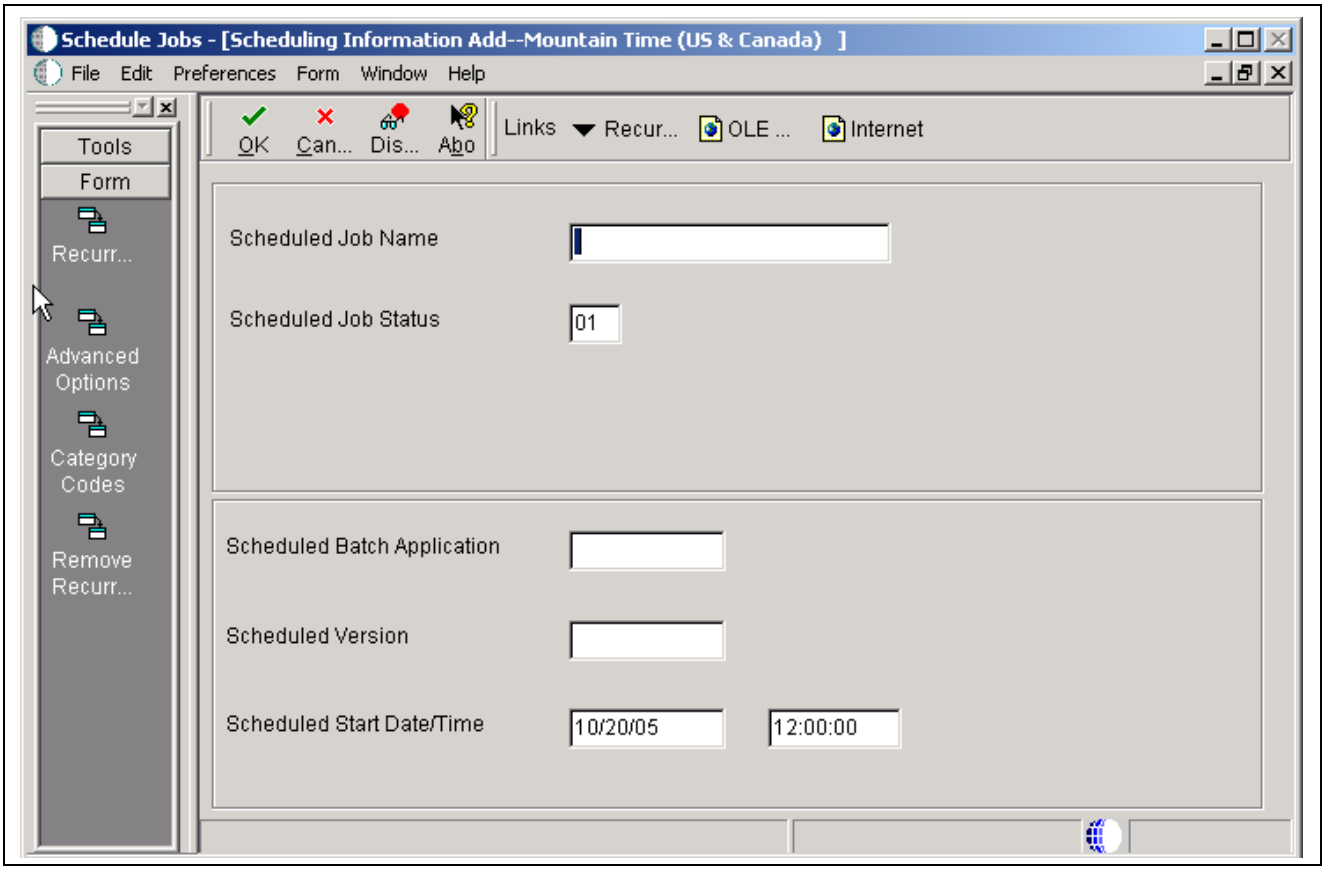
Scheduling the JD Edwards EnterpriseOne Push Installation Batch Application

After you have installed the JD Edwards EnterpriseOne Listener on all affected workstations and have scheduled the package through the JD Edwards EnterpriseOne Deployment Director program (P9631), you must use the JD Edwards EnterpriseOne Schedule Jobs program (P91300) to run the JD Edwards EnterpriseOne Push Package Installation batch program (R98825) on the server.

Before you begin to schedule the JD Edwards EnterpriseOne Push Installation batch application, complete these steps:

- Remind package recipients to leave their workstations turned on, even after hours.
- Remind users who are using a local service that they must be signed in.
- Remind package recipients to verify that the JD Edwards EnterpriseOne Listener is running.

Access the Scheduling Information form:



Scheduling Information form

1. Enter a name that uniquely identifies a scheduled job in the Scheduled Job Name field.
2. Enter the current status of the scheduled job in the Scheduled Job Status field.

   As long as the status is active, the JD Edwards EnterpriseOne Scheduler determines whether the job should be submitted to the server for processing. When the scheduled end date for the job has been reached, the status changes to Not Active. To stop the JD Edwards EnterpriseOne 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 JD Edwards EnterpriseOne Scheduler to include the job again, but you can reactivate a job only if the end date is in the future.

3. Enter the object name of the report that the Schedule submits to the server in the Scheduled Batch Application field.
4. Enter the version of the report to run in the Scheduled Version field.

   This is the version of the report scheduled to run. A version identifies a specific set of data selections and sequencing settings that the batch job uses.
5. In the Scheduled Start Date/Time field, enter the next date on which the JD Edwards EnterpriseOne Scheduler submits the scheduled job to the server for processing.
6. To set the job recurrence (that is, to specify how frequently the job runs) select 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. For the JD Edwards EnterpriseOne Push Installation batch program, you should set recurrence to run every 30 minutes.
7. On the Recurring Scheduling Information Revisions form, click OK.
8. On the Scheduling Information form, to enter any overrides, resubmissions, or expiration options, select Advanced Options from the Form menu.
9. Click the tab that corresponds to the information that you want to enter or revise:
   - Launch Overrides
   - Job Expiration
   - Job Resubmission
   - Batch Application Overrides
10. Revise the information, and click OK.

After scheduling the job, you can use the JD Edwards EnterpriseOne Object Configuration Manager (P986110) to verify that the server on which the JD Edwards EnterpriseOne Push Installation batch program is running points to the same F98825 and F9650 tables that the JD Edwards EnterpriseOne Deployment Director program uses.

See Also

JD Edwards EnterpriseOne Tools 8.98 System Administration Guide, "Using the Scheduler Application," Working with the Job Scheduler

Running the Package Installation Results Report

Access the Push Package Installation Results form.

This report provides the same information that you get when you run the JD Edwards EnterpriseOne Push Package Installation batch program (R98825).

The report includes this 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).

This table lists the status codes and descriptions that the JD Edwards EnterpriseOne Push Package Installation program (R98825) uses. Codes that are marked with an asterisk indicate conditions in which the JD Edwards EnterpriseOne Push Package Installation program continues to attempt the installation the next time that the JD Edwards EnterpriseOne Push Package Installation program runs.

Status Code	Description
200*	Scheduled
210*	In Progress
220	Successful Install
230	Install Failed
240*	Install Running
250*	JD Edwards EnterpriseOne Running
260*	Listener Not Started/Installed
270	General Error
280	Already Installed
290	Invalid Package
300	Install Attempted
310	Machine Down

Installing Workstations from CD

This section provides an overview of how to install workstations from CD, lists a prerequisite, and discusses how to:

- Define the CD Writer location.
- Deploy a package to the CD Writer location.
- Create the installation CD.

Understanding How to Install Workstations from CD

If your system includes a CD writer, you can build and deploy packages to the CD writer location. After copying the package to a CD, you can then use the CD as a portable deployment tier from which to perform workstation installations. That is, you can run from the CD the setup.exe program that launches the JD Edwards EnterpriseOne Workstation Installation program.

You can set up your enterprise so that you can deploy packages to the CD writer and install the software from a CD.

The first step in the process of configuring your system for deployment from CD is to define the CD writer location if it is not already defined. In this step, you essentially create a pseudo deployment server from which you will later copy package data onto the CD by using the software for your CD writer.

When you define the CD writer location in the Machine Identification application, you must also add the correct path codes to the Environments exit.

The process for defining this location is identical to the process for defining any other new deployment server.

Prerequisite

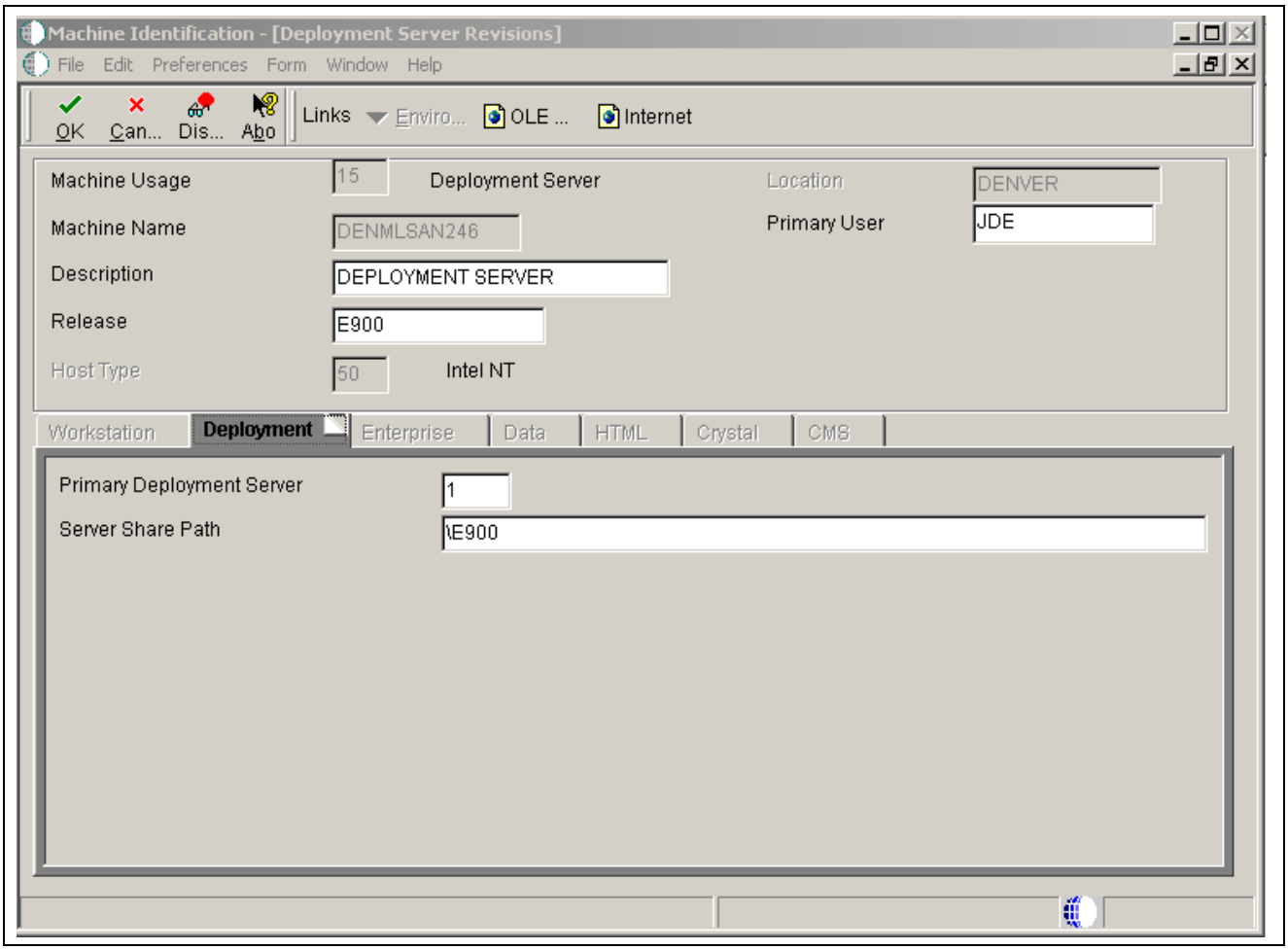
Assemble, define, and build the package that you want to write to the CD.

Forms Used to Install Workstations from CD

Form Name	FormID	Navigation	Usage
Deployment Server Revisions	W9654AC	Package and Deployment Tools (GH9083), Machine Identification Subordinate to the appropriate location, select Deployment server. Click Add to add a new machine.	
Work with Package Deployment	W9631J	Package and Deployment Tools (GH9083), Package Deployment	Select the package to deploy and review your selections.

Defining the CD Writer Location

Access the Deployment Server Revisions form.



Deployment Server Revisions form

1. Enter the name of the machine on the network in the Machine Name field.
2. Enter the release number as defined in the Release Master in the Release field.
3. Enter the primary user for the listed machine in the Primary User field.
4. Enter the shared directory for the path code in the Server Share Path field.
5. If you want to specify a location for data, a foundation, or help files, do so by choosing Data, Foundation, or Helps from the Form menu.

   If you do not specify a location for data, foundation, or helps, the system uses the default locations.
6. Click OK.
7. Click Close to quit the Work with Locations and Machines form.
8. In Windows Explorer, locate the folder named Client Install.
9. Copy this folder by dragging the folder to the CD writer location.

   The location is the server share path that you entered on the Deployment Server Revisions form.

Deploying a Package to the CD Writer Location

After you define the CD writer as a deployment server, you are ready to deploy a package to the CD writer location that you specified. This task involves these two procedures:

- Deploy to the CD writer location the package that you want to write to the CD.
- Modify the Install.inf and Package.inf files in preparation for writing the package contents to the CD.

Access the Work with Package Deployment form.

1. Click Add.
2. Complete the forms in the JD Edwards EnterpriseOne Deployment Director in the same way as you would for any other package.
3. From the Work with Package Deployment form, find and select the package that you just scheduled for deployment, and then select 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 this example:

```
Multitier\package_inf\Appl_B.inf
Multitier\systemcomp\system.cab
Multitier\datacomp\data.cab
Multitier\helpscomp\helps.cab
Multitier\Appl_pgf\Package\Appl_B
Multitier\package_inf\Appl_B.inf
```

In the previous example, Multitier is the name of the server share path. Appl_B is the package name.

Note. The server share path name is not included when you copy folders to the CD. In the previous example, the items that are copied to the CD are \package_inf\Appl_B.inf, \systemcomp\system.cab, and so on.

To modify the Install.inf and Package.inf files:

1. In Windows Explorer, find the CD writer location and open the folder that contains the package that you deployed.

   This folder has the name that you entered in the Server Share Path field on the Deployment Server Revisions form when you defined the CD writer location. In the previous example, the server share path name is Multitier.

2. Open the Client Installation folder, and then open the file Install.inf.

   That is, double-click the icon for the file to launch the Microsoft Notepad application.

3. In the section [FileLocations], modify the line so that two periods and a backslash (\) precede the package_inf entry.

   The line should look like this example after you modify it:

   ```
   [FileLocations]
   PackageInfs=..\package_inf
   ```

4. Similarly, open the Package_inf folder and open the *package name*.inf file.

   In the previous example, this file is named Appl_b.inf.

5. In the section [SrcDirs], modify each of the lines so that two periods and a backslash (\) precede each entry.

   After modification, the [SrcDirs] section should look similar to this example:

   ```
   [SrcDirs]
   SAPPL_PGF=..\APPL_PGF\package\APPL_B
   ```



```
SSYS=..\systemcomp
SAPPL_PGFDATA=..\datacomp
SHELP=..\helpscomp
```

Creating the Installation CD

After you deploy the package to the CD writer location and modify the Install.inf and Package.inf files, you are ready to copy the package contents to the CD. Use the software that came with your CD writer to accomplish this process, which typically involves copying the package contents to the CD. Refer to the documentation that came with your CD writer for more information about this process.

You copy the package to the CD by copying the subdirectories that are subordinate to the server share path directory. The server share path directory is not created on the CD. (In the previous example, the server share path directory is called Multitier, and it is the same name that you entered in the Server Share Path field on the Deployment Server Revisions form.

When you are finished copying the directories to the CD, the CD should contain these directories:

- Appl_pgf (contains package information).
- datacomp (contains the database cabinet file).
- helpscomp (contains the helps cabinet file).
- systemcomp (contains the foundation cabinet file).
- package_inf (contains the package.inf file).
- Client Install (contains the JD Edwards EnterpriseOne Workstation Installation program).

Note. Actual names might not be the same as those listed because each system might be different.

CHAPTER 8

Working with Packages for Business Services

This chapter provides an overview of packages for business services and discusses how to:

- Assemble JD Edwards EnterpriseOne business services.
- Assemble a package that contains published business services.
- Build a package with published business services.
- Deploy the package to the Business Services Server.

Understanding Packages for Business Services

This section discusses an overview of packages for business services.

Once business services have been created, they need to be built for Oracle Application Server (OAS) and WebSphere Application Server (WAS) consumption. The package build process creates the necessary .ear files that are consumable by OAS and WAS. Next, the business services need to be deployed. The client installation process deploys business services to all Microsoft Windows 32 clients. The package deployment process deploys the OAS and WAS .ear files to preconfigured J2EE servers.

See Also

*Business Services Server Reference Guide*for more information on configuration and security of the Business Services Server

Assembling JD Edwards EnterpriseOne Business Services

This section lists prerequisites and discusses how to assemble business services for package build.

Prerequisites

Before you complete the tasks in this section:

- Use Server Manager to create J2EE business service containers for the Business Services Server.
- Use Server Manager to set up Server Manager users.

  In order to deploy the package successfully, the EnterpriseOne user must be a valid Server Manager user. The user cannot deploy the package if the EnterpriseOne user's credentials are not valid for Server Manager.

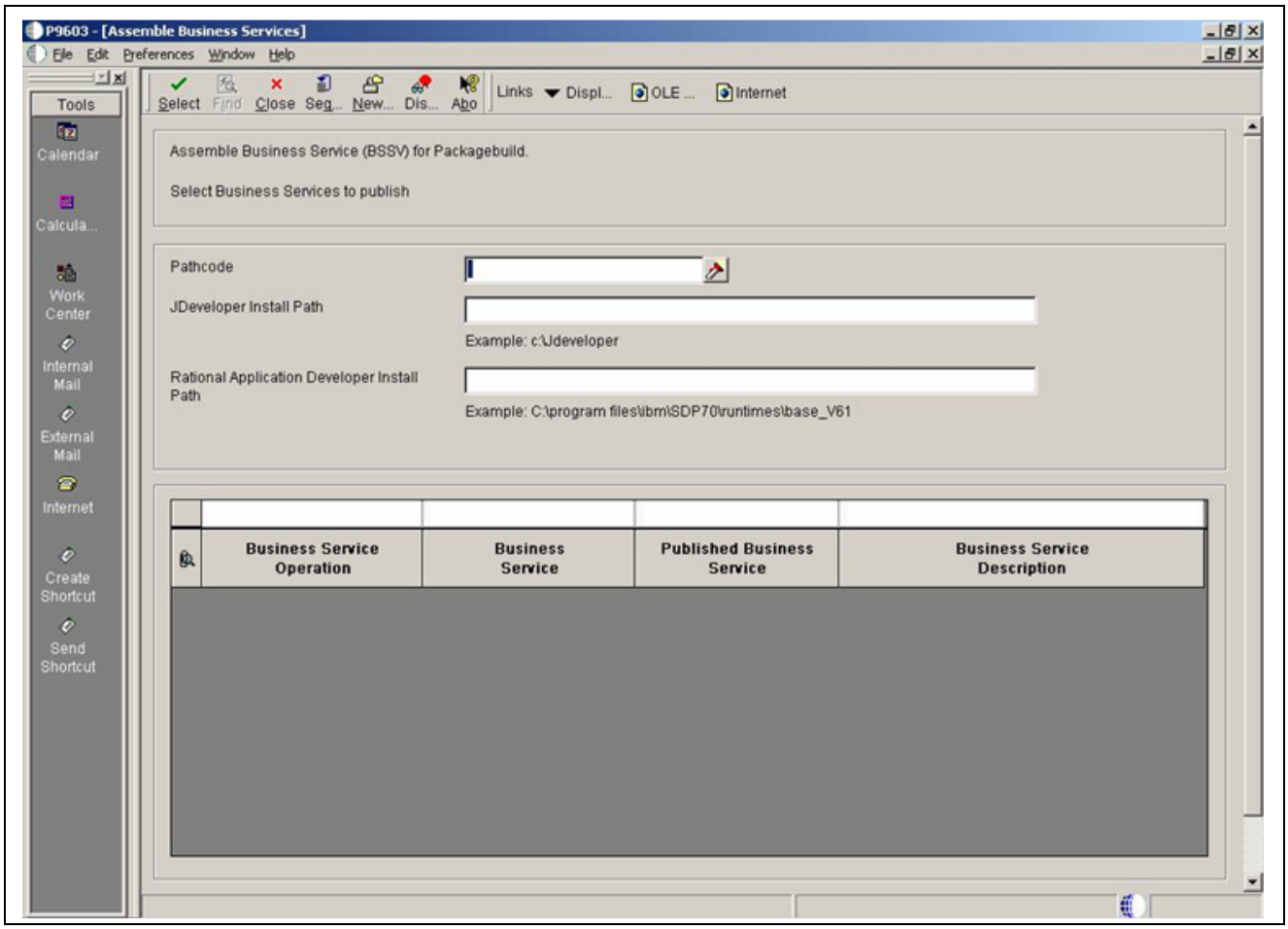
  See the *JD Edwards EnterpriseOne Tools Release 8.98 Server Manager Guide.*

- If the JD Edwards EnterpriseOne client that you are using to build a business services package has been installed with WebSphere Express, you must uninstall WebSphere Express and install IBM Rational Application Developer for WebSphere (RAD). The application for building a package with business services for WAS uses RAD.
- If you have multiple security servers, you must set up EnterpriseOne Trusted Nodes for a successful deployment of business services.

  See *JD Edwards EnterpriseOne Tools 8.98 Security Administration Guide*, "Setting Up JD Edwards EnterpriseOne Single Sign-On," Setting Up a Trusted Node Configuration.

Assembling Business Services for Package Build

Enter P9603 in the Fast Path.



Assemble Business Services form

1. On the Assemble Business Services form, in the Pathcode field, enter the path code of the package that you plan to build and tab to the next field.

   Note. At this point, the application retrieves all the available business services from F98602 and F98603. If you have used this application before, the application also retrieves the values for the JDeveloper Install Path and Rational Application Developer Install Path fields from the JDE.INI file.

2. If this is your first time in the application, you must manually complete these fields:

Field	Description
JDeveloper Install Path	Enter the location where JDeveloper is installed.
Rational Application Developer Install Path	Enter the location where IBM Rational Application Developer for WebSphere (RAD) is installed. Enter this location only if you will be building a package for WAS.

3. When you enter the install path, P9603 verifies the actual location and version.

   If this path is correct, the P9603 adds the information to the jde.ini:

```
[MTR VALIDATION]
JDeveloperInstallPath=<Install path specified by P9603>
JDeveloperVersion=10.1.3
WebSphereInstallPath=<Install path specified by P9603>
WebSphereVersion=6.1
```

   Note. If the path or version is incorrect, an error appears; the Close button is disabled until you enter the correct path.

4. In the grid, select the business services that you want to expose as a web service and click the Select button.

   You can also double-click the row headings of the business services that you want to expose.

   A check mark appears by each business service that is selected.

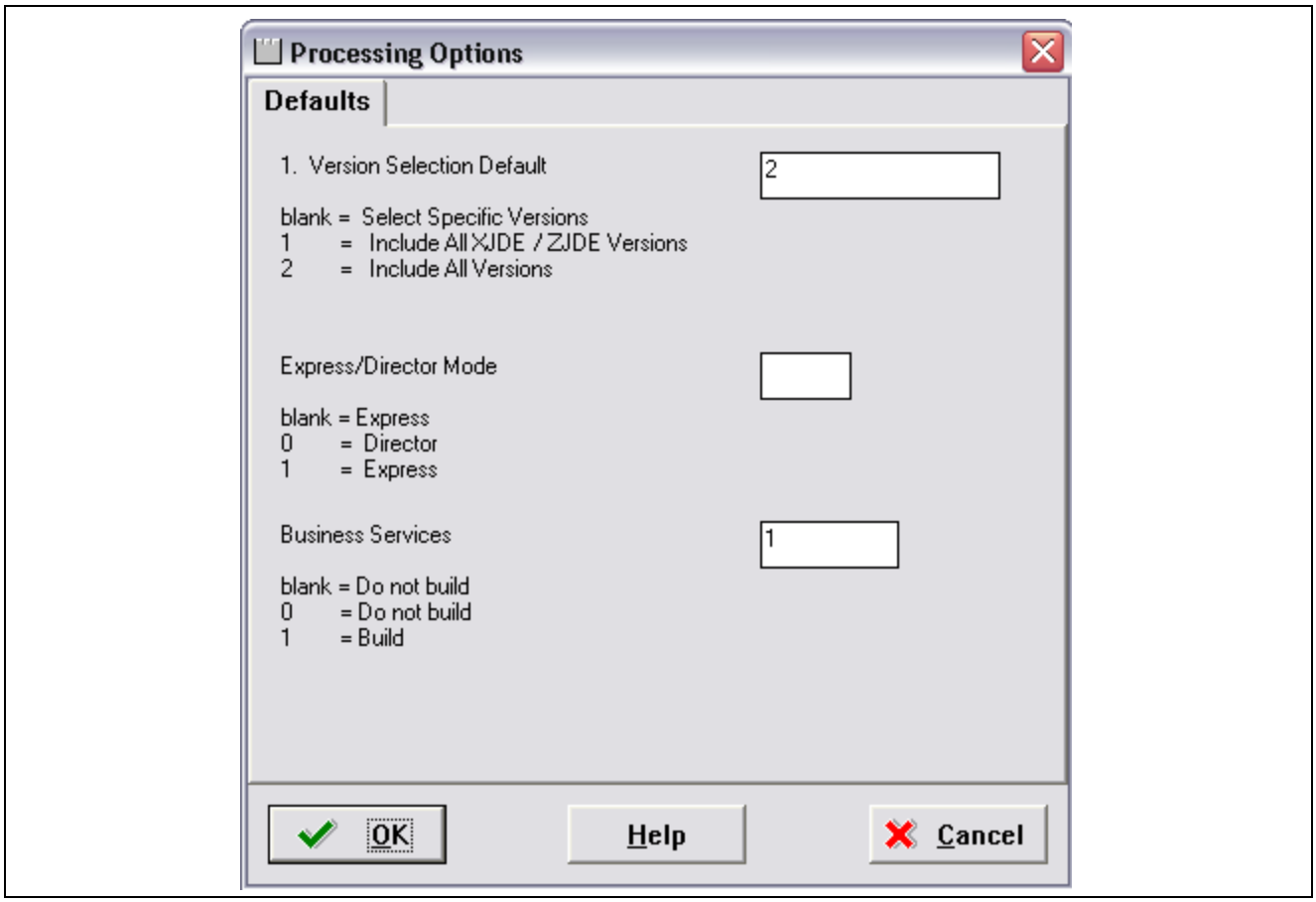
5. Click Select again or double-click the row header to unexpose the web service.
6. Click Close to close the application.

Assembling a Package that Contains Published Business Services

This section discusses how to assemble a business service package.

Assembling a Business Service Package

To set the processing options for Package Assembly, go to the Package and Deployment Tools menu, right-click the Package Assembly application (P9601), and select prompt for values.



Processing Options for Package Assembly, P9601

1. Set the processing option entitled Business Services to *1*. This processing option is blank by default.
2. Select OK.
3. On the Work with Packages form, begin the assembly process.

   See Chapter 4, "Assembling Packages," page 37.

Building a Package with Published Business Services

This section provides an overview of the build process, prerequisites, and discusses how to:

- Define a package build with published business services.
- Resubmit the package build.

Understanding the Build Process

This section provides an overview of how the JD Edwards EnterpriseOne system builds a package that contains business services. The JD Edwards EnterpriseOne system:

1. Creates the \\work\sbf\sbfbuild.ini, which defines the paths to the exposed methods.
2. Creates the Ant scripts logtimestamp.xml and build.xml in the \\work\sbf directory.

3. Runs the build.xml Ant script to extract source.

   When the extract occurs, the business service.log is generated in the \\work\sbf directory.

4. Creates service interface files.

   An interface file is created for each published business service. The selected methods are listed in the created published business service.

5. Creates the Web Service Inspection Language (WSIL) file, which is used for Business Process Execution Language (BPEL).

6. Creates Ant scripts for OAS.

   These scripts are named build.properties and build.xml. They are created within the \\work\sbf\OAS directory.

7. Creates Ant scripts for WAS.

   These scripts are created within the \\work\sbf\WAS directory.

8. Runs the OAS build.xml Ant script.

   The build.xml Ant script:

   a. Creates javadoc.

   b. Compiles java source files.

   c. Assembles all the source into an .ear file that is OAS compatible.

9. Runs the WAS build.xml Ant script.

   The build.xml script creates an .ear file that is WAS compatible.

10. Compresses the java folder for deployment of the client sbf.cab.

Note. Review the buildservices.log to verify that the build was successful. If the build is successful, "Build Successful" appears at the bottom of the log. If the build failed, the business services is flagged as failed; the package still appears to have been built successfully, even though the .ear file is not generated.

Prerequisites

Before building the package, verify that logging is turned off. When the jdeproperties.log file is set for logging in the \system\classes folder and a package build is submitted, the build process is slowed down. It is not recommended to have logging turned on during package builds.

Defining a Package Build with Published Business Services

To define a package build with published business services:

1. Enter P9621 in the Fast Path or go to the Package and Deployment Tools menu and select Package Build.
2. Use the Package Build application to define build properties for the package.
3. On the Package Build Location form, select Client as the Build Location.

Note. Business services are not built for server-only packages.

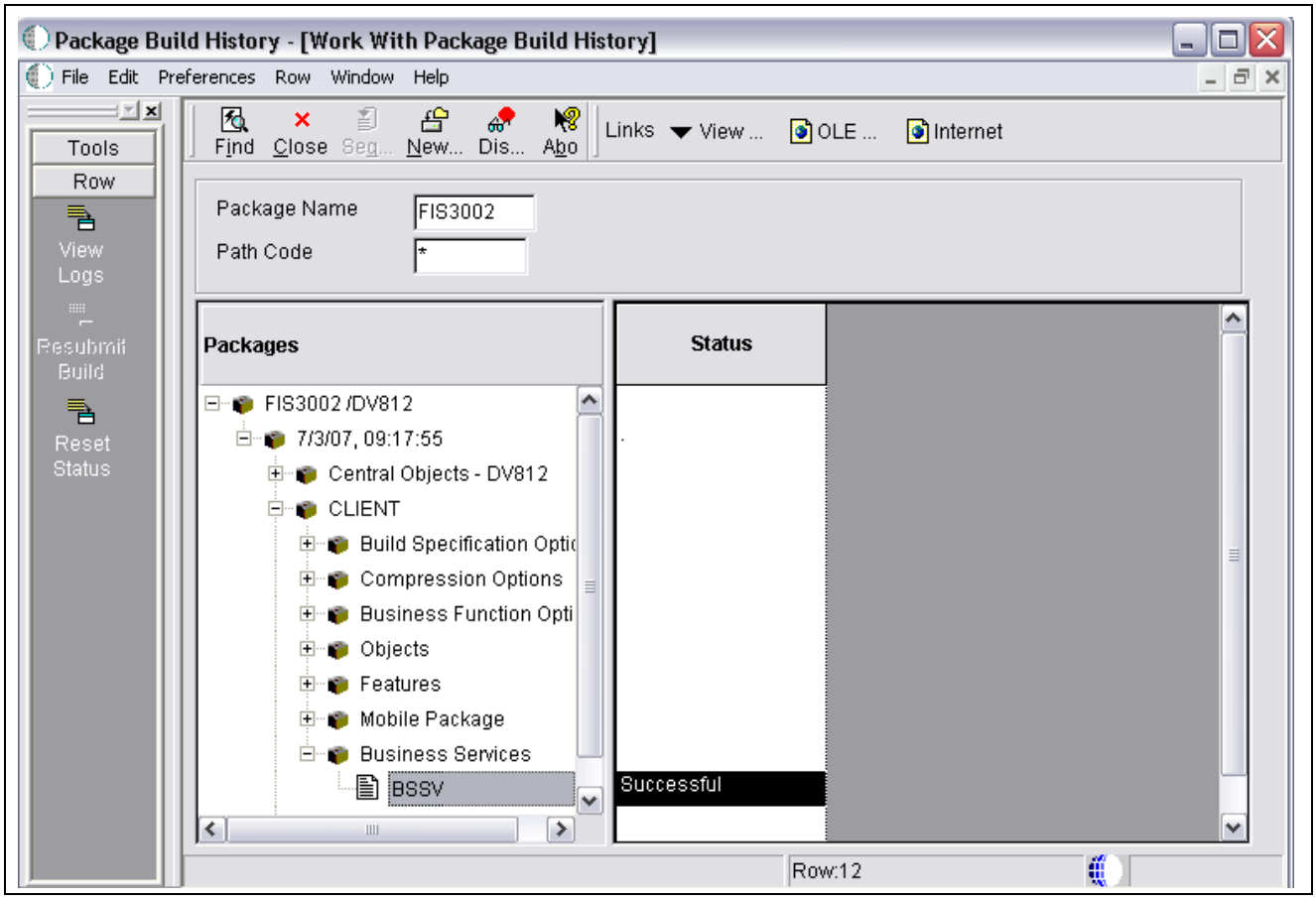
The JVM's virtual memory is set to a maximum of 512 MB. You can change this using the following jde.ini settings:

```
[PACKAGE BUILD]

JavacMaxMemorySize= 512 MB

JavadocMaxMemorySize= 512 MB
```

Resubmitting the Package Build

If errors occur during the package build, you will need to reset the status and resubmit the package build.

Enter P9622 in the Fast Path or go to the Package and Deployment Tools menu and select Package Build History.



Work with Package Build History form

1. On the Work with Package Build History form, enter your package in thePackage Name field and select Find.
2. In the tree structure, expand your package name and CLIENT.
3. Click Business Services and select your business service.
4. Select Reset Status from the Row menu to reset the status of your business services.
5. Select Resubmit Build from the Row menu.

Deploying the Package to the Business Services Server

This section provides an overview of the deployment process and discusses how to deploy the business services.

Understanding the Deployment Process

This is an overview of how business services are deployed to the Business Services Server.

1. When you click Deploy, the R98825F runs.

   Note. If you are deploying the package to both the enterprise server and the Business Services Server, you select the enterprise server and click Deploy. The R98825D then calls R98825F when it is finished.

2. The R98825F creates the scfjar folder.
3. The scf_manifest.xml is created in the scfjar folder.

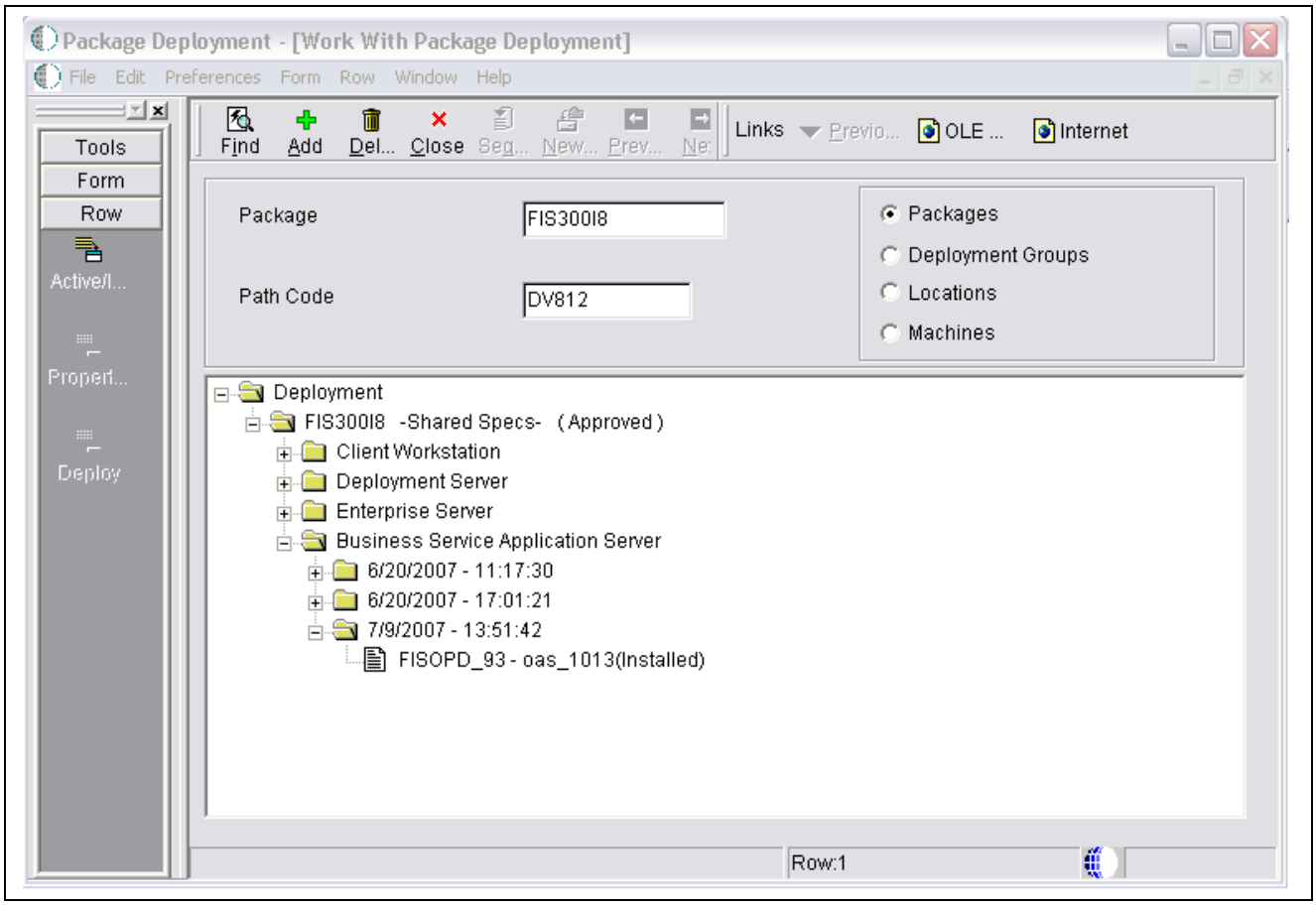
   This xml contains information that the package deployment process uses to communicate with the server manager.
4. The OAS and WAS .ear files are copied to the scfjar folder.
5. The contents of the scfjar folder are combined to form the bssv_timestamp.jar file.

   Note. If errors occur, they are logged to the jas.log or jasdebug.log files.

6. The jar file is uploaded to Server Manager and both the file and the scfjar folder are deleted from the deployment server.

Deploying the Business Services

Enter P9631 in the Fast Path or go to the Package and Deployment Tools menu and select Package Deployment.



Work with Package Deployment form

1. On the Work with Package Deployment form, click the Add button.
2. On the Package Selection form, select the business services package to deploy and click Next.
3. On the Package Deployment Targets form, select Business Services Server as the Deployment Target.

   Note. The check box for Business Services Server is disabled if the selected package does not contain business services.

4. On the Package Deployment Attributes form, enter the Management Server URL and click Next.

   Server Manager returns a list of eligible Business Services Servers to which the user can deploy the business services.

   Note. In order to deploy the package successfully, the EnterpriseOne user must be a valid Server Manager user. The user cannot deploy the package if the EnterpriseOne user's credentials are not valid for Server Manager. The user also cannot deploy the package if there is no valid Business Services Server defined in Server Manager.

   See the *JD Edwards EnterpriseOne Tools Release 8.98 Server Manager Guide* for information about setting up Server Manager users.
5. Select the appropriate servers and click Next.
6. On the Business Services Package Deployment Properties Revisions form, click End.

7. On Work with Package Deployment, open your package name and then Business Service Application Server in the tree structure.
8. Select the date/time stamp and select Deploy from the Row menu.

Note. You cannot deploy to an individual Business Services Server. Business services are deployed to all servers under the selected date/time stamp.

CHAPTER 9

Setting Up Multitier Deployment

This chapter provides an overview of multitier deployment and discusses how to:

- Define deployment servers.
- Distribute software to deployment locations.
- Deploy server packages in a multitier network.

Understanding Multitier Deployment

This section discusses:

- Overview of multitier deployment.
- Multitier deployment terminology.
- Multitier deployment features.
- Multitier implementation.
- Multitier deployment case study.

Overview of Multitier Deployment

JD Edwards EnterpriseOne software is normally distributed to workstations from a centralized location. In many cases, you can minimize the affect on the performance of a single deployment server by using systematic scheduling for software installations. For example, if your site has more than 50 workstations that require a package installation but you release software only four times a year, you can mitigate performance problems by scheduling the installations over a weekend, at night, or during off-peak hours.

While this distribution method is the simplest approach for software deployment, network capacity constrains configurations that have either multiple remote sites or large numbers of users at a single site. For example, software installations to workstations that are connected to the centralized deployment location by a 56 KB circuit can take 4 to 6 hours to run.

Multitier deployment provides sites the flexibility of installing packages on workstations and servers from more than one deployment location and more than one deployment server. These additional deployment locations and servers are called deployment tiers. Specifically, instead of installing multiple workstations across a wide area network (WAN) circuit, multitier deployment enables you to transfer a compressed package from the centralized location to the remote workgroup server, which acts as a second deployment tier. Multitier deployment means deploying from more than one deployment tier.

For example, you might have one deployment server at the main location and a second deployment server for a remote location. Because the server at the remote location is responsible for deploying to workstations and servers at that location, you do not need to deploy packages from the main deployment server across a WAN, as you would in a single-tier deployment configuration.

Workgroup servers can also be used as second-tier deployment locations in a local area network (LAN) environment, where large numbers of workstations need to install software concurrently. It is recommended that you implement multitier deployment if your site has more than 50 workstations receiving multiple installations per day.

The primary function of multitier deployment is to reduce network traffic (and the delays that result from heavy traffic) by enabling enterprises with more than one geographic location to deploy from a secondary deployment server at the remote site. Instead of installing packages across the WAN from the deployment server to workstations at a remote location, you can copy the package and the package.inf file from the deployment server at the primary location to the deployment server at the remote location. The server at the remote location can then deploy the package to the workstations and servers at that location.

Consider implementing a multitier deployment configuration when either of these situations applies:

- Too many workstations are installing packages from the same location, causing server and network performance to suffer.
- Workstations are remotely connected to the deployment server over a WAN, which requires unacceptably long installation times.

Normally, you decide whether to implement multitier deployment during Configurable Network Computing (CNC) implementation. Although you can enable this function at any time, you typically set it up after you have installed the software and are preparing the production sites to go live.

To set up multitier deployment, you must define the machines (and their associated path codes) that are used for deployment. In addition, you can use a scheduler function to define when software should be pushed to the tiered deployment location.

You must also define individual user characteristics for multitier deployment. Normally, you do this by modifying the user profiles to indicate the deployment location from which a user pulls a package.

Multitier Deployment Terminology

This table lists and describes multitier deployment terms:

Term	Description
Primary Deployment Location (tier 1)	The primary or base deployment location is the location in which the package to be deployed to secondary or remote locations is built. The system 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 JD Edwards EnterpriseOne products, and should not be used for any other purposes in your company.

Term	Description
Tier Deployment Location (tier 2)	Tier deployment locations (also known as remote or secondary locations) have one or more deployment servers that enable you to install the software on the workstations at that location. These servers receive their packages from the deployment server at the primary or base location. Machines at the tier deployment locations cannot use Object Librarian functions such as object check-in and check-out. These machines are designed for deployment use only and are not designed to be used for remote development. The number of tiered locations that you can have depends on the network and server capacity.
Tier Workstations	Tier workstations are workstations that connect to a tier deployment location for software installation. The number of workstations per deployment location depends on the network and machine load. All tiered workstations must also have a Microsoft Windows domain connection that enables them to connect to, read, and copy files from the shared drives of their respective deployment locations.

See Also

JD Edwards EnterpriseOne Application Release 9.0 Installation Guide

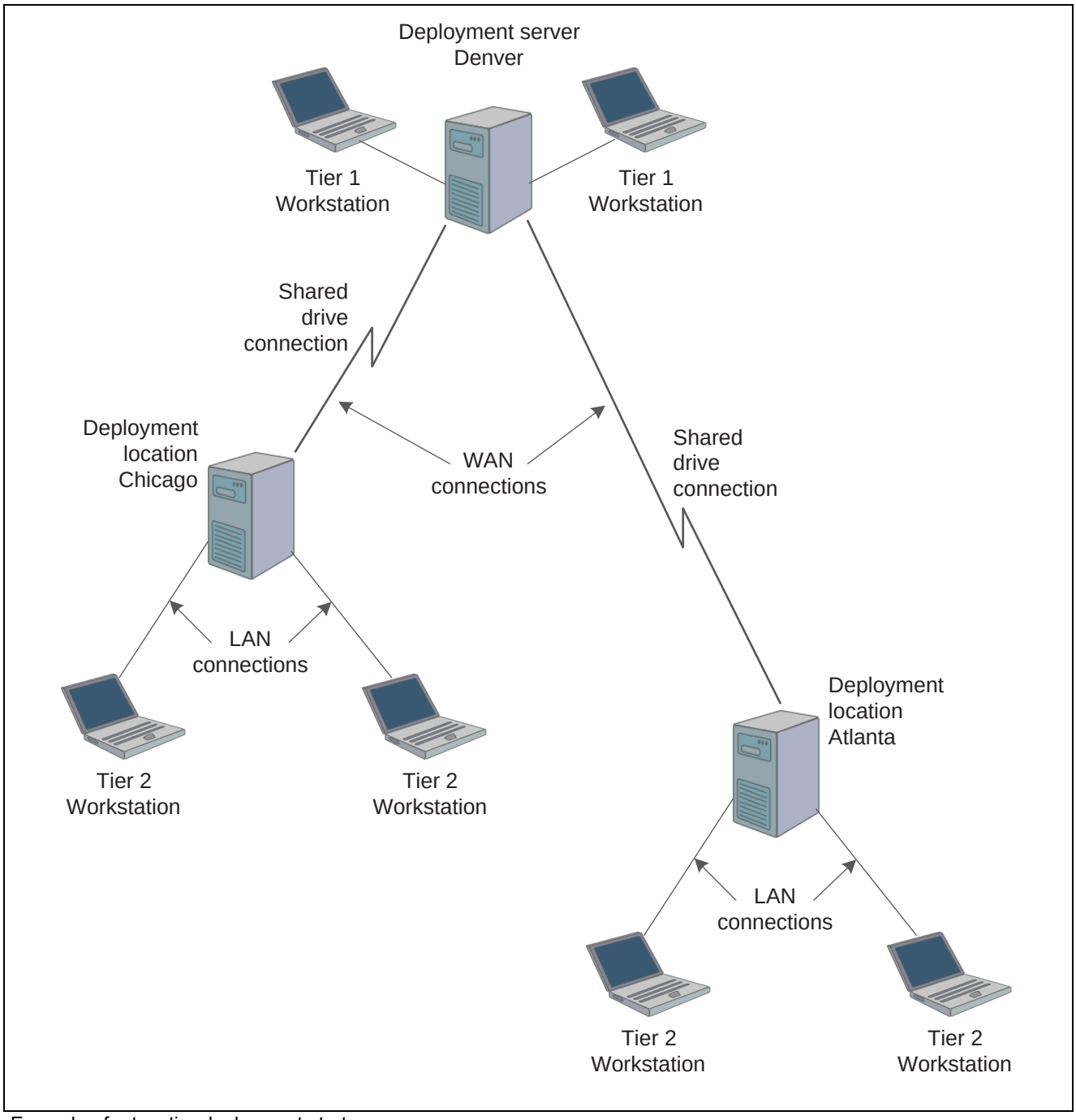
Multitier Deployment Features

Multitier deployment offers these 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 can be a Microsoft Windows workstation.
- Setup and administration are straightforward tasks.
- You maintain central control over files and data that are transferred to remote deployment locations.
- You can easily schedule software for deployment to remote sites.
- Multitier deployment is integrated into Oracle's JD Edwards EnterpriseOne Deployment Director, so the process for deploying is essentially the same as the process for deploying in a single-tiered setup.

Example: Two-Tier Deployment Strategy

This diagram illustrates a typical two-tier deployment strategy:



Example of a two-tier deployment strategy

Multitier Implementation

Packages are always built on the deployment server at the primary location. After you build the package that you want to deploy to remote locations, you must follow these steps to implement multitier deployment:

1. Define deployment locations.

   You must define each physical location to which you want to deploy. For example, if the main office is in Denver and you want to deploy to the branch office in Seattle, you must define the Seattle deployment location.

2. Create deployment server definitions.

   You must define the deployment server at each remote location.

   Note. This step is not necessary if you used Oracle's JD Edwards EnterpriseOne Remote Location Workbench to create deployment server definitions when you installed the software.

3. Schedule the package.

   Use the JD Edwards EnterpriseOne Deployment Director program (P9631) to schedule the package for deployment. The process of scheduling a package for multitier deployment is identical to the process for scheduling any other package.

4. Deploy the package to workstations.

   After you deploy the package to the deployment server at the remote location, that server can deploy to the workstations at that location.

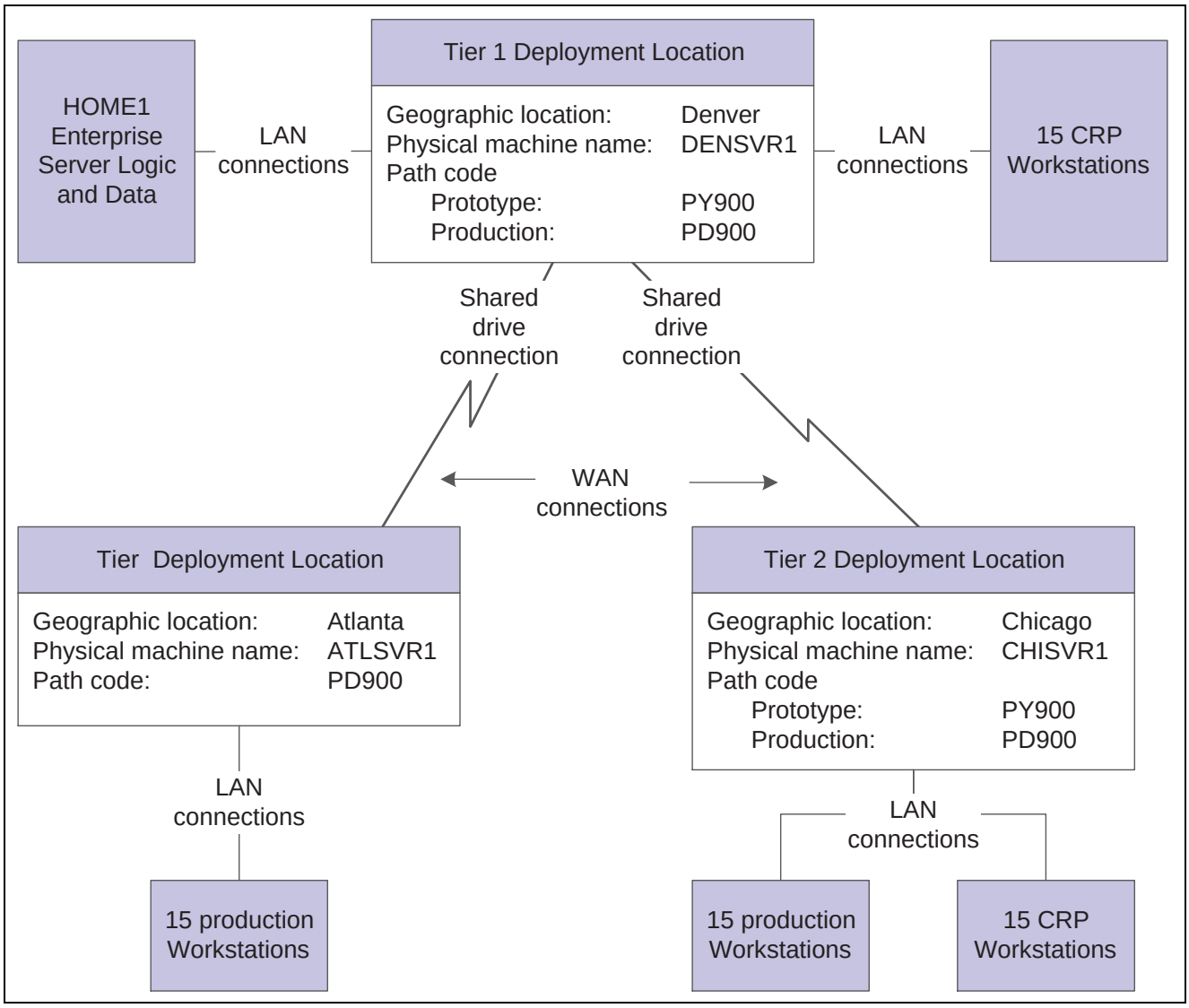
Multitier Deployment Case Study

This case study is for a two-tier deployment environment. As this case study demonstrates, deploying packages to workstations using WAN connections is generally not efficient. Instead, you should deploy from a primary deployment server to tier deployment locations. After that, you can install packages to LAN-attached workstations from each local deployment location.

For 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 enterprise server and database server connectivity.

This diagram illustrates this case study:



Multitier Deployment Strategy

This table describes the assumptions used by the Tier 1 Denver location in this case study:

Characteristic	Setting
Characteristic	Setting
Enterprise server name	HOME1
Deployment server name	Denver: DENSVR1 Atlanta: ATLSVR1 Chicago: CHISVR1
Prototype workstations	Denver: 15 Atlanta: 0 Chicago: 15

Characteristic	Setting
Production workstations	Denver: 0 Atlanta: 15 Chicago: 15
JD Edwards EnterpriseOne release	E900
Deployment tier	Denver: 1 Atlanta: 2 Chicago: 2
Path codes	Denver: PD900, PY900 Atlanta: PD900 Chicago: PD900, PY900

Multitier Deployment Configuration Steps for the Case Study

These steps summarize the steps necessary to configure the system for multitier deployment:

1. Define the deployment locations.

   Define the deployment locations on the deployment server (DENSVR1 in this example). Use Oracle's JD Edwards EnterpriseOne Deployment Locations Application program (P9654A) to define all deployment locations.

   For this case study, complete these fields to define three locations, one for each deployment location in Denver, Atlanta, and Chicago:

Field	Value
Location	Enter the name of the deployment location. In this case study, you assign these names for each physical deployment location: *Denver, Atlanta,* and *Chicago.*
Description	Enter a description (any value up to 30 characters) for each deployment location; for example: Denver: *Denver - Tier 1* Atlanta: *Atlanta - Tier 2* Chicago: *Chicago - Tier 2*
Location Code	Enter the current location for deployment; for example, *DEN.*
Parent Location	Enter the name of the parent location for the location that you are adding; for example, *Corporate.*

2. Create deployment server definitions.

   Use the JD Edwards EnterpriseOne Deployment Locations Application program (P9654A) to create a definition for each deployment server at the deployment locations that you created. For this case study, you need to define a deployment server for Atlanta and Chicago. The deployment server in Denver is already defined because it is the primary (tier 1) server.

   For this case study, complete the fields on the Deployment Server Revisions form as described in this table:

Field	Value
Machine Name	Enter the name of the physical machine. In this case study, enter these 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 number of the release. For example, E900.
Primary User	Enter the primary user for the machine that you entered.
Server Share Path	Enter the name of the shared directory for the path code in which system files and other files reside. For example, \E900.

3. Schedule the package.

   Schedule the package to be deployed from the tier 1 deployment server to the tier 2 deployment servers through the JD Edwards EnterpriseOne Deployment Director program (P9631) or Multi Tier Deployment batch program (R98825C).

See Also

Chapter 7, "Deploying Packages," Defining Deployment Parameters, page 127

Defining Deployment Servers

This section provides an overview of defining deployment servers, lists prerequisites, and discusses how to:

- Define a new deployment server.
- Revise an existing deployment server.

Understanding Defining Deployment Servers

The JD Edwards EnterpriseOne Deployment Locations Application program (P9654A) enables you to either add a new deployment server definition or modify an existing definition. When you add a new deployment server definition, the system creates a record in the F9654 table for each deployment location. Each server at each deployment location must be defined.

Typically, the table contains one record for the primary deployment server and one record per deployment location for each release. If you have multiple releases, you must create multiple records for the servers at each deployment location.

If you used Oracle's JD Edwards EnterpriseOne Remote Location Workbench to create deployment server definitions when you installed the software, you do not need to define deployment servers again.

In many situations, you might need to modify the definition for a deployment server that you already defined. For example, you would need to change the definition if the server share path or release changes, or if you want to designate a different server as the primary deployment server. The process for revising an existing deployment server definition is similar to the process for adding a new definition.

See Also

Working with Installation Workbench in the *JD Edwards EnterpriseOne Application Release 9.0 Installation Guide*

Prerequisites

Before you complete the tasks in this section:

- Ensure that you understand the CNC concepts that enable you to define and implement packages, workstation installations, path codes, central objects, replicated objects, and user profiles.
- Ensure that adequate disk space exists on the disk drive for the machine that you will be using as a tier deployment location.

  Each full package that you deploy adds approximately 1.4 GB. The amount of required disk space varies, depending on the amount of data that you replicate to a Supported Local Database.
- If you are adding a new definition for a deployment server at a remote location, you first need to define that location.

Note. It is recommended that you do not define deployment locations with a DEV path code. The DEV path code is normally associated with developers who are not located at remote locations.

See Chapter 7, "Deploying Packages," Defining Locations, page 132.

Form Used to Define a Deployment Server

Form Name	FormID	Navigation	Usage
Deployment Server Revisions	W9654AC	Package and Deployment Tools (GH9083), Machine Identification Find the location where the deployment server resides and expand the tree, if necessary, to display the different machine types. Select the Deployment Server heading and then click Add, or find the existing deployment server you want to modify and click Select.	Define a new deployment server or revise an existing deployment server for multitier deployment.

Defining a New Deployment Server

Access the Deployment Server Revisions form.

1. Enter the name of the machine on the network in the Machine Name field.

   Because you are entering a definition for a server other than the primary deployment server, the *Primary Deployment Server* field is not available. You can enter this field only if you first delete the definition for the current primary deployment server.
2. Enter the release number as defined in the release master in the Release field.
3. Enter the primary user for the listed machine in the Primary User field.
4. Enter the shared directory for the path code in the Server Share Path field.

   The objects that are stored on a file server will be found in this path.
5. Select Path Code from the Form menu.
6. 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 use the base server share path as the default for that machine.

   Note. 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.
7. If the package includes a user-defined foundation or data, specify those items by selecting Foundation or Data from the Form menu.
8. On the Deployment Server Revisions form, click OK.

Revising an Existing Deployment Server

Access the Deployment Server Revisions form.

1. Modify any of these fields:
   - Description

- Release
- Primary User
- Server Share Path
- Primary Deployment Server

  If you are modifying the primary deployment server, you can also change the primary deployment server. For any other server, you cannot change the value in this field. Enter *1* to indicate that the deployment server is the primary deployment server. You can have only one primary deployment server.

2. Select Path Code from the Form menu.
3. On the Machine Path Code Revisions form, enter the path code for the primary or base location from which the secondary location receives the package.

   When you enter the path code, the system displays the default server share path, which is the base server share path for that machine.

   Note. You must specify the path code for each deployment server at all secondary locations. If you do not indicate the path code, multitier deployment might not work.

4. If the package includes a user-defined foundation or data, specify those items by selecting Foundation or Data from the Form menu.
5. On the Deployment Server Revisions form, click OK.

See Also

Chapter 7, "Deploying Packages," Defining Deployment Parameters, page 127

Distributing Software to Deployment Locations

This section provides an overview of the multitier software distribution process and discusses how to:

- Distribute software through package deployment.
- Schedule packages for multitier deployment.
- Distribute software through the multitier deployment batch process.
- Copy workstation installation programs to deployment locations.

Understanding the Multitier Software Distribution Process

You use the JD Edwards EnterpriseOne Deployment Director program (P9631) or Multi Tier Deployment batch program (R98825C) to distribute software to deployment locations and schedule them for deployment. Use the JD Edwards EnterpriseOne Deployment Director program to define the scheduling parameters or to deploy the package immediately. Otherwise, use a version of Oracle's JD Edwards EnterpriseOne Multi Tier Deployment batch application to distribute the software to deployment locations.

Whether you push or pull the software depends on the machine on which you run the deployment function or batch program for the scheduling program. You push the software if you run the program or report on the primary deployment server or from a workstation. Conversely, you pull the software if you run the program from a workstation at the tier deployment location. In either case, execute the application on the primary deployment server machine for push installation or the destination deployment location for pull installation.

Important! If you push the software, you must have full read and write privileges for both deployment servers (that is, the primary deployment server and the tier deployment location or destination machine), even if you do not run the batch application. If you do not have read and write authority on both servers, the deployment will fail.

After the package software is distributed through the JD Edwards EnterpriseOne Deployment Director program or Multi Tier Deployment batch program, you must manually copy the workstation installation programs from the primary deployment server to the tier deployment location. These programs are located in the client portion of the base installation directory.

Form Used to Distribute Software to Deployment Locations

Form Name	FormID	Navigation	Usage
Work with Package Deployment	W9631J	System Administration Tools, Package and Deployment Tools, Package Deployment	Distribute software that was previously defined and scheduled through the Deployment Director program (P9631).

Distributing Software Through Package Deployment

Use this method when you want to distribute software immediately after you define a deployment schedule. If you select this option, the software is distributed immediately, regardless of the timing parameters that you specify in the scheduling fields.

Access the Work with Package Deployment form.

1. Select the Machines option, and then click Find.
2. Select the deployment server to which you want to deploy.

   If you have scheduled to deploy packages to more than one deployment server, you can select the Deployment Server folder to deploy to all applicable servers rather than deploying to each individual server.
3. Select the deployment server name to deploy all packages that are listed under the server name.

   Alternatively, select only the package that you want to deploy.
4. From the Row menu, select Deploy.

   This option launches Oracle's JD Edwards EnterpriseOne Multi Tier Deployment batch program (R98825C), which enables you to deploy to deployment servers.

   You can also use this option to launch a batch application that deploys enterprise server packages. Therefore, if you select an enterprise server name or the Enterprise Server folder on the Work With Package Deployment form, Oracle's JD Edwards EnterpriseOne Enterprise Server Deployment batch program (R98825D)—not the Multi Tier Deployment batch program—launches. When you select a workstation, the Deploy option is not available.

Scheduling Packages for Multitier Deployment

After you create the server package that you want to deploy using multitier deployment, you can use the JD Edwards EnterpriseOne Deployment Director program (P9631) to schedule that package to a server at the same location or a remote location. Verify that the server package is compressed because, unlike client packages, you cannot deploy the server package using multitier deployment unless it is compressed.

Also, before you can deploy a server package with server multitier deployment, the package status must be either 50 (Build Completed Successfully) or 70 (Build Completed with Errors). If the package does not have a status of 50 or 70, it will be unavailable when you schedule the deployment.

See Chapter 7, "Deploying Packages," Scheduling a Package for Deployment, page 140.

Note. When you create a distribution schedule, remember that you cannot schedule multiple packages to deploy at the same time.

Distributing Software Through the Multitier Deployment Batch Process

Access the Work with Batch Versions - Available Versions form.

1. Enter *R98825C* in the Batch Application field and click Find.
2. Select the Multi Tier Deployment version that you want to use.
3. On Version Prompting, click Submit.
4. On Report Output Destination, select one of these options and click OK:
   - On Screen
   - Printer

Copying Workstation Installation Programs to Deployment Locations

These steps ensure that the system runs the JD Edwards EnterpriseOne Client Workstation Installation program locally at the deployment location. If you do not complete these steps, the system searches the base location for the JD Edwards EnterpriseOne Client Workstation Installation program and its associated files, and copies the files across the WAN to the deployment location.

1. Connect to each deployment location and use Microsoft Windows Explorer to drag this client directory to the tier 2 workstation:

   ```
   \\Tier1DeploymentServerName\E900\OneWorld Client Install
   ```

2. Open the package.inf file and locate the [FileLocations] section.

   Change the PackageInfs line to reflect the machine name of the deployment server and the environment at the deployment location; for example:

   ```
   PackageInfs=\\machine name\environment\ENVIRON\Evapps\appl_pgf\package_inf
   ```

3. Save your changes by selecting Save from the File menu.
4. Select Exit from the File menu.

In the case of multitier deployment, the client installation program resides on the tier deployment location in the client subdirectory that is subordinate to the base installation directory. The workstation that is attached to the deployment location must have read access to this directory on the deployment location to install the workstation package.

See Also

Installing the Workstations for Developers and System Administrators in the *JD Edwards EnterpriseOne Application Release 9.0 Installation Guide*

Deploying Server Packages in a Multitier Network

This section provides an overview of multitier deployment of server packages and discusses how to schedule a server package for multitier deployment.

Understanding Multitier Deployment of Server Packages

Server multitier deployment lets you automatically deploy a server package from one deployment server to another. The target deployment server to which you are deploying the package can be either in the same location as the source deployment server that sends the package or in one or more remote locations.

You typically build packages on the primary deployment server at the base location, but using a different server for installations from the base location might have advantages. In some situations, you might prefer to deploy a server package to a server at a remote location rather than require remote users to access the server package over a WAN.

Server package multitier deployment enables users to select 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 select builds is that users are no longer required to wait to install a new client package.

You use the JD Edwards EnterpriseOne Deployment Director program (P98631) for server multitier deployment. When you deploy a server package, the system copies these components:

- Foundation and data.
- Subdirectories of the package name directory:
  - bin32
  - include
  - lib32
  - obj
  - source
  - spec
- Subdirectories of the server build directories:
  - bin32
  - spec
  - obj

- lib32

- ServerPackage.inf file in the server build directories:
  - The package.inf file.
  - The pack directory that is subordinate to the package name directory.

When server-only builds are deployed without client builds, only the pack subdirectory is copied.

This table describes where major package components are copied from and to during the deployment process; the server share path is derived from the F9651 table:

Package Component	Copied From	Copied To
Package	The path indicated in the SrcDirs section of the package.inf file.	*<server share path>**<path code>**<package name>* on the destination deployment server.
Foundation and data	The path indicated in the SrcDirs section of the package.inf file.	*<server share path>**systemcomp* on the destination deployment server if the default location is selected.
Package.inf file	The path indicated in the F00942 table if the source deployment server is the primary deployment server in the base location. Otherwise, the path to the package.inf file in the F9651 table.	*<server share path>**package_inf* on the destination deployment server.

Smart Deployment

Server multitier deployment incorporates the smart deployment feature, which makes available only the server packages that match the available servers for the package destination. For example, if server packages exist at the base location for the HP9000 and the iSeries but the destination location has only an HP9000, only the HP9000 package is made available for deployment to that location. You cannot deploy a server package unless the package destination supports that server platform.

Even if you have multiple locations, smart deployment ensures that only the server packages that match the destination platform are available.

Automatic Package.inf File Updating

When you deploy a server package to a remote location, these 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.

Prerequisite

Assemble and build the server package.

Form Used to Schedule a Server Package for Multitier Deployment

Form Name	FormID	Navigation	Usage
Package Deployment Targets	W9631B	Package and Deployment Tools (GH9083), Package Deployment Click Add to launch the Deployment Director, and then click Next.	Select a package to deploy.

Scheduling a Server Package for Multitier Deployment

Access the Package Deployment Targets form.

1. Select the Deployment Server option.
2. On the Package Deployment Attributes form, enter the deployment date and time, and select deployment options.
3. On the Deployment Server Selection form, enter the machine name to which you want to deploy the server package.

   Select the machine by double-clicking the row header.
4. On the Build Selection form, select the build (or version) of the package that you want to deploy by double-clicking the row header.

   Because of the Smart Deployment feature, the system enables you to select only the package builds that match the configuration at the destination location. For example, if package builds exist for an iSeries and an HP 9000 but the destination location has only an HP9000, you cannot select the iSeries build.
5. Click Close to exit the Build Selection form.
6. On the Work With Package Deployment form, click End to finish the server package scheduling process.

See Also

Chapter 9, "Setting Up Multitier Deployment," Distributing Software to Deployment Locations, page 183

APPENDIX A

Adding a Security Override for Package Build

This appendix provides an overview of security overrides for package build and discusses how to:

- Add a system user for the central objects data source owner.
- Add a security override to run package build.

Understanding Security Overrides for Package Build

Before you build a package, you must have a security administrator add a security override if your system meets both of these conditions:

- The database that you are using is Oracle, SQL, or DB2 UDB for Microsoft Windows/Unix.
- The security server is enabled.

A security override is required so that the package build process can create the metadata repository tables in central objects. To add a security override, a security administrator must first add a system user for the central objects data source owner, and then add a security override for the JD Edwards EnterpriseOne user who will run the package build.

Adding a System User for the Central Objects Data Source Owner

To add a system user for the central objects data source owner:

In JD Edwards EnterpriseOne Solution Explorer, enter *P98OWSEC* in the Fast Path.

1. On Work with User Security, from the Form menu, select Add System User.
2. On Work with System Users, in the System User field, enter the appropriate data source owner, for example *DV900, PY900,* or *PD900.*
3. Click the Find button.
4. If no values are returned, add the data source owner as a system user:
    a. Click the Add button.
    b. On System User Revisions, complete these fields: System User, Data Source, Password, and Password Verify.
    c. Click OK and then click the Cancel button.
5. Click the Close button to return to the Work with User Security form.

Adding a Security Override to Run Package Build

To add a security override to run package build:

1. On Work with User Security, enter the user who is going to run the package build, and then click the Find button.
2. From the Form menu, select Add Data Source.
3. On Add Data Source, complete these fields:
   - User ID
   - Data Source
   - System User

Note. For iSeries, either sign in as a user who has rights to create tables in the central objects library, or follow the steps to set up a security override for the JD Edwards EnterpriseOne user. When the JD Edwards EnterpriseOne user connects to the data source, the user connects as a system user (iSeries user profile) who has update rights to the library.

APPENDIX B

Using the Microsoft Visual C++ 2005 Compiler

This appendix provides an overview of Microsoft Visual C++ 2005 runtime libraries and discusses how to:

- Create a VS2005 runtime library package feature.
- Create an update package with the VS2005 runtime library feature.
- Build and deploy an update package with the VS2005 runtime library feature.
- Install the VS2005 runtime library on an enterprise server.

Understanding Microsoft Visual C++ 2005 Runtime Libraries

This section discusses:

- Microsoft Visual C++ runtime libraries background
- Redistribution of Microsoft Visual C++ 2005 runtime libraries

Microsoft Visual C++ Runtime Libraries Background

Historically, the Microsoft Visual C++ compiler runtime libraries have been redistributed as part of our JD Edwards OneWorld Xe and EnterpriseOne products. For past Visual C++ compiler releases the redistribution of the compiler specific runtime libraries has been quite simple. Our installer process placed the runtime libraries in a location found within the server's path, (for example, %SYSTEMROOT%\system32), to make them accessible.

Due to Microsoft Visual C++ compiler release 2005 (v.8) changes, the runtime libraries are no longer backward and forward compatible. When JD Edwards EnterpriseOne objects are compiled using Microsoft Visual C++ 2005 and linked into a dynamic link library (DLL), a manifest file is created for each DLL. This DLL-specific manifest identifies the runtime library version used to compile and link the objects that were built. Unlike past compilers, the runtime libraries must not only be release-specific but also version-specific. For instance, when the Microsoft Visual C++ 2005 Compiler and SPn, (where n is some service pack) are installed on a machine, the Windows\WinSxS (side-by-side) folder is updated to include the associated compiler runtime libraries with the release level of '8.0.50727.762'.

With the Visual C++ 2005 compiler, manifests associated with our DLLs must now be created. Each DLL-specific manifest identifies a specific runtime library release and version, for example '8.0.50727.762.' This association requires that all Microsoft Windows machines that are part of a JD Edwards EnterpriseOne solution and that perform business function builds share a Microsoft Visual C++ 2005 compiler with identical service pack releases.

When you build new packages using the Microsoft Visual C++ 2005 SPn compiler, you must ensure that all machines receiving these packages have the corresponding runtime libraries installed. This is important to note. Assuming Microsoft makes a new service pack available for its Visual C++ 2005 compiler and it is installed on your JD Edwards EnterpriseOne Windows build machines, you must do the following:

- Ensure that all JD Edwards EnterpriseOne Windows build machines, both servers and workstations, have the identical compiler service pack release levels installed.
- Distribute the new Visual C++ 2005 SPn runtime libraries to all Microsoft Windows machines that are receiving packages built by Visual C++ 2005 SPn and do not have a compiler installed.

Note. Microsoft does make the Visual C++ 2005 SPn redistributable package available from the Microsoft Download Center. From the Microsoft Download Center search for any of the following: vcredist_x86.exe, Microsoft Visual C++ 2005 SP1 Redistributable Package (x86), etc.

Redistribution of Microsoft Visual C++ 2005 Runtime Libraries

This process applies to customers adopting Microsoft Visual C++ 2005. All JD Edwards EnterpriseOne Windows machines receiving application foundation packages built with Microsoft Visual C++ 2005 require the runtime libraries to be installed. The Microsoft Visual C++ 2005 Redistributable Package (vcredist_x86.exe) installs runtime components of Visual C++ required to run applications developed with Visual C++ on a computer that does not have Visual C++ 2005 compiler installed.

The absence of the Microsoft Visual C++ 2005 runtime libraries from a machine using a JD Edwards EnterpriseOne application foundation package built by the same compiler will result in “Business function Library load failed ...” error messages. Once the Visual C++ 2005 runtime libraries are installed on a Microsoft Windows machine, only new service pack updates to the Microsoft Visual C++ 2005 compiler require redistribution of new runtime libraries.

Note. All references to Microsoft Visual C++ 2005 within this document refer to the JD Edwards EnterpriseOne defined Microsoft Windows platform compiler minimum technical requirements, for example, Microsoft Visual C++ 2005 SPn. Please refer to the JD Edwards EnterpriseOne minimum technical requirements to identify supported versions of the Microsoft Windows platform compiler.

A Microsoft or third-party system management tool such as SMS can be used to distribute the Microsoft Visual C++ 2005 runtime libraries. This is generally the recommended approach for the distribution of Microsoft packaged product. The JD Edwards EnterpriseOne package build “feature” can also be used to push Microsoft’s redistributable runtime library package to all Microsoft Windows client machines. Delivery of Microsoft Visual C++ 2005 runtime libraries for JD Edwards EnterpriseOne enterprise, logic, application, or batch servers is explained in the appendix at the following link.

See Appendix B, "Using the Microsoft Visual C++ 2005 Compiler," Installing the VS2005 Runtime Library on an Enterprise Server, page 196.

Creating a VS2005 Runtime Library Package Feature

The JD Edwards EnterpriseOne package build “feature” makes it possible to distribute third party applications with the deployment of a client package. Create a package feature for the Microsoft Visual C++ 2005 compiler runtime libraries to leverage this facility.

The deployment server should have a copy of the Microsoft Visual C++ 2005 SPn compiler installed. By default the installation path for this compiler release is C:\Program Files\Microsoft Visual Studio 8.

To create a VS2005 runtime library package feature:

1. On the deployment server, open Windows Explorer and navigate to your JD Edwards EnterpriseOne solution's shared folder. For example, E900.
2. Expand the shared node and open OneWorld Client Installs\ThirdParty.
3. Under the folder ThirdParty, create a new folder with the name *VS2005RTL*.
4. Locate and copy the vcredist_x86.exe file from your installed compiler path. For example, C:\Program Files\Microsoft Visual Studio 8\SDK\v2.0\BootStrapper\Packages\vcredist_x86.
5. Copy the vcredist_x86.exe file into the VS2005RTL folder.
6. Log into the deployment server's DEP900 environment.
7. Fast path to menu GH9083, Package and Deployment Tools, and select the Package Assembly application.
8. On the Work with Packages form, select Form and then Features.
9. On the Work with Features form, click Add.
10. Click Next to begin the Feature Based Deployment Director.
11. On the Feature Information form, complete these fields and click Next:

Field	Value
Feature	*VS2005_RTL*
Feature Type	*1*
Description	*Visual C++ 2005 Runtime Libraries*
Feature Installation Options	*Required*
Components	*Additional Install Processes*

12. On the Additional Install Processes form, select the Execute After Install option.
13. Complete these fields and click Next:

Field	Value
Third Party	*VS2005RTL*
Description	*Visual C++ 2005 Runtime Libraries*
Sequence	*1*
Executable Name	*vcredist_x86.exe* This is the name of the executable file found in the ...\ThirdParty\VS2005RTL folder.

Field	Value
Source Path	\\<deploymentservername>\E900\OneWorld Client Install\ThirdParty\VS2005RTL **Note.** Do not use the Select Directory browse function to obtain the aforementioned path. Instead type the UNC path or cut and paste the Universal Naming Convention (UNC) path from Windows Explorer into the Source Path field.
Parameters	*/Q* /Q denotes Quiet Mode and does not require any user intervention.

14. Click Save to preserve the feature settings and then click Next to continue.
15. On the Feature Summary form, click End to complete the feature definition.

Note. If you expand the nodes describing each package feature you may inspect the feature definition. You may notice that the UNC share path has been truncated for your newly created entry. This is NOT an issue as this line entry serves only as a description. The complete UNC share path has been properly preserved in System table F96604.

Creating an Update Package with the VS2005 Runtime Library Feature

Once the VS2005 runtime library package feature has been created, it can be associated with either an update or full package. Creating an update package containing this feature will cause the full parent package assembly information to include this same feature.

To create an update package with the VS2005 runtime library feature:

1. Go to menu GH9083, Package and Deployment Tools, and select the Package Assembly application.
2. On the Work with Packages form, click Add.
3. On the Package Assembly Director, click Next to begin the package assembly process.
4. On the Package Information form, select the Express option, complete these fields, and click Next:

Field	Example Value
Package Name	*DV2005RTL*
Description	*Visual C++ 2005 RTL for DV*
Path Code	*DV900*

5. On the Package Component Revisions form, select the Update option and type or select the parent package. For example, *DV900FA*.
6. Click the Features button.
7. On the Features Components form, click the Browse button.
8. On the Feature Component Selection form, click Find.

9. Highlight the entry associated with VS2005_RTL and click Select to mark the entry with a check mark.
10. Click Close and Close again to return to the Package Component Revisions form.
11. Verify that the form shows “Individual Features Selected” and click End to complete the package assembly process.
12. On the Work with Packages form, select Row and Active/Inactive to activate the package.

Building and Deploying an Update Package with the VS2005 Runtime Library Feature

After creating the update package, you will need to build and deploy the package.

To build and deploy an update package with the VS2005 runtime library feature:

1. Highlight your package and select Row and Define Build.
2. On the Package Build Definition Director, click Next to begin the build definition process.
3. On the Package Build Revisions form, ensure that the Build Location Client check box is checked and click Next.
4. On the Build Features tab, select the Build Feature INFs option.
5. On the Package Build Revisions form, click End to complete the build definition process.
6. On the Work with Package Build Definition form, select Row and Active/Inactive to activate the package build definition.
7. Select Row and Submit Build to build the package.
8. On the Report Output Destination form, select On Screen and click OK.
9. Once the package build has completed, review the R9621 PDF report file to verify that the build completed successfully.

   The successful package build with the included package feature results in the creation of a feature-specific INF file.
10. After building the package, the appropriate person must approve it for client deployment.

    Afterwards, both the update and associated parent package will automatically include the Microsoft Visual C++ 2005 runtime libraries as part of the client installation process.

    Since this feature was configured to install in Quiet Mode (/Q), it does not require any user intervention. If the Microsoft Visual C++ 2005 runtime libraries are already installed on the machine, the feature-specific installer will exit.

Installing the VS2005 Runtime Library on an Enterprise Server

All JD Edwards EnterpriseOne Microsoft Windows machines receiving application foundation packages built by Microsoft Visual C++ 2005 require the runtime libraries to be installed. Customers using a JD Edwards EnterpriseOne enterprise, logic, application, or batch server that does not have a Microsoft Visual C++ 2005 compiler installed, must install the associated Microsoft redistributable runtime library package (vcredist_x86.exe). Failure to install the runtime libraries while using a Microsoft Visual C++ 2005 built application foundation package on a machine without the supported compiler will result in "Business function Library load failed ..." error messages.

A Microsoft or third-party system management tool such as SMS could be used to distribute the Microsoft Visual C++ 2005 runtime libraries to these machines. This is generally the recommended approach for the distribution of Microsoft packaged products. In the absence of a system management tool, simply install the Microsoft redistributable runtime library package (vcredist_x86.exe).

To install the Microsoft redistributable runtime library package:

1. 1. Locate the vcredist_x86.exe file on a machine with the Microsoft Visual C++ 2005 compiler installed. The default installed compiler path is C:\Program Files\Microsoft Visual Studio 8\SDK\v2.0\BootStrapper\Packages\vcredist_x86.
2. 2. Copy the vcredist_x86.exe file to a folder on your local machine.
3. Double-click the vcredist_x86.exe file executable to install the Microsoft Visual C++ 2005 runtime libraries.

Glossary of JD Edwards EnterpriseOne Terms

Accessor Methods/Assessors Java methods to "get" and "set" the elements of a value object or other source file.

activity rule The criteria by which an object progresses from one given point to the next in a flow.

add mode A condition of a form that enables users to input data.

Advanced Planning Agent (APAg) A JD Edwards EnterpriseOne tool that can be used to extract, transform, and load enterprise data. APAg supports access to data sources in the form of rational databases, flat file format, and other data or message encoding, such as XML.

alternate currency A currency that is different from the domestic currency (when dealing with a domestic-only transaction) or the domestic and foreign currency of a transaction.

In JD Edwards EnterpriseOne Financial Management, alternate currency processing enables you to enter receipts and payments in a currency other than the one in which they were issued.

Application Server Software that provides the business logic for an application program in a distributed environment. The servers can be Oracle Application Server (OAS) or WebSphere Application Server (WAS).

as if processing A process that enables you to view currency amounts as if they were entered in a currency different from the domestic and foreign currency of the transaction.

as of processing A process that is run as of a specific point in time to summarize transactions up to that date. For example, you can run various JD Edwards EnterpriseOne reports as of a specific date to determine balances and amounts of accounts, units, and so on as of that date.

Auto Commit Transaction A database connection through which all database operations are immediately written to the database.

back-to-back process A process in JD Edwards EnterpriseOne Supply Management that contains the same keys that are used in another process.

batch processing A process of transferring records from a third-party system to JD Edwards EnterpriseOne.

In JD Edwards EnterpriseOne Financial Management, batch processing enables you to transfer invoices and vouchers that are entered in a system other than JD Edwards EnterpriseOne to JD Edwards EnterpriseOne Accounts Receivable and JD Edwards EnterpriseOne Accounts Payable, respectively. In addition, you can transfer address book information, including customer and supplier records, to JD Edwards EnterpriseOne.

batch server A server that is designated for running batch processing requests. A batch server typically does not contain a database nor does it run interactive applications.

batch-of-one immediate A transaction method that enables a client application to perform work on a client workstation, then submit the work all at once to a server application for further processing. As a batch process is running on the server, the client application can continue performing other tasks.

See also direct connect and store-and-forward.

best practices Non-mandatory guidelines that help the developer make better design decisions.

BPEL Abbreviation for *Business Process Execution Language*, a standard web services orchestration language, which enables you to assemble discrete services into an end-to-end process flow.

BPEL PM Abbreviation for *Business Process Execution Language Process Manager*, a comprehensive infrastructure for creating, deploying, and managing BPEL business processes.

Build Configuration File Configurable settings in a text file that are used by a build program to generate ANT scripts. ANT is a software tool used for automating build processes. These scripts build published business services.

build engineer An actor that is responsible for building, mastering, and packaging artifacts. Some build engineers are responsible for building application artifacts, and some are responsible for building foundation artifacts.

Build Program A WIN32 executable that reads build configuration files and generates an ANT script for building published business services.

business analyst An actor that determines if and why an EnterpriseOne business service needs to be developed.

business function A named set of user-created, reusable business rules and logs that can be called through event rules. Business functions can run a transaction or a subset of a transaction (check inventory, issue work orders, and so on). Business functions also contain the application programming interfaces (APIs) that enable them to be called from a form, a database trigger, or a non-JD Edwards EnterpriseOne application. Business functions can be combined with other business functions, forms, event rules, and other components to make up an application. Business functions can be created through event rules or third-generation languages, such as C. Examples of business functions include Credit Check and Item Availability.

business function event rule See named event rule (NER).

business service EnterpriseOne business logic written in Java. A business service is a collection of one or more artifacts. Unless specified otherwise, a business service implies both a published business service and business service.

business service artifacts Source files, descriptors, and so on that are managed for business service development and are needed for the business service build process.

business service class method A method that accesses resources provided by the business service framework.

business service configuration files Configuration files include, but are not limited to, interop.ini, JDBj.ini, and jdelog.properties.

business service cross reference A key and value data pair used during orchestration. Collectively refers to both the code and the key cross reference in the WSG/XPI based system.

business service cross-reference utilities Utility services installed in a BPEL/ESB environment that are used to access JD Edwards EnterpriseOne orchestration cross-reference data.

business service development environment A framework needed by an integration developer to develop and manage business services.

business services development tool Otherwise known as JDeveloper.

business service EnterpriseOne object A collection of artifacts managed by EnterpriseOne LCM tools. Named and represented within EnterpriseOne LCM similarly to other EnterpriseOne objects like tables, views, forms, and so on.

business service framework Parts of the business service foundation that are specifically for supporting business service development.

business service payload An object that is passed between an enterprise server and a business services server. The business service payload contains the input to the business service when passed to the business services server. The business service payload contains the results from the business service when passed to the Enterprise Server. In the case of notifications, the return business service payload contains the acknowledgement.

business service property Key value data pairs used to control the behavior or functionality of business services.

Business Service Property Admin Tool An EnterpriseOne application for developers and administrators to manage business service property records.

business service property business service group A classification for business service property at the business service level. This is generally a business service name. A business service level contains one or more business service property groups. Each business service property group may contain zero or more business service property records.

business service property categorization A way to categorize business service properties. These properties are categorized by business service.

business service property key A unique name that identifies the business service property globally in the system.

business service property utilities A utility API used in business service development to access EnterpriseOne business service property data.

business service property value A value for a business service property.

business service repository A source management system, for example ClearCase, where business service artifacts and build files are stored. Or, a physical directory in network.

business services server The physical machine where the business services are located. Business services are run on an application server instance.

business services source file or business service class One type of business service artifact. A text file with the .java file type written to be compiled by a Java compiler.

business service value object template The structural representation of a business service value object used in a C-business function.

Business Service Value Object Template Utility A utility used to create a business service value object template from a business service value object.

business services server artifact The object to be deployed to the business services server.

business view A means for selecting specific columns from one or more JD Edwards EnterpriseOne application tables whose data is used in an application or report. A business view does not select specific rows, nor does it contain any actual data. It is strictly a view through which you can manipulate data.

central objects merge A process that blends a customer's modifications to the objects in a current release with objects in a new release.

central server A server that has been designated to contain the originally installed version of the software (central objects) for deployment to client computers. In a typical JD Edwards EnterpriseOne installation, the software is loaded on to one machine—the central server. Then, copies of the software are pushed out or downloaded to various workstations attached to it. That way, if the software is altered or corrupted through its use on workstations, an original set of objects (central objects) is always available on the central server.

charts Tables of information in JD Edwards EnterpriseOne that appear on forms in the software.

check-in repository A repository for developers to check in and check out business service artifacts. There are multiple check-in repositories. Each can be used for a different purpose (for example, development, production, testing, and so on).

connector Component-based interoperability model that enables third-party applications and JD Edwards EnterpriseOne to share logic and data. The JD Edwards EnterpriseOne connector architecture includes Java and COM connectors.

contra/clearing account A general ledger account in JD Edwards EnterpriseOne Financial Management that is used by the system to offset (balance) journal entries. For example, you can use a contra/clearing account to balance the entries created by allocations in JD Edwards EnterpriseOne Financial Management.

Control Table Workbench An application that, during the Installation Workbench processing, runs the batch applications for the planned merges that update the data dictionary, user-defined codes, menus, and user override tables.

control tables merge A process that blends a customer's modifications to the control tables with the data that accompanies a new release.

correlation data The data used to tie HTTP responses with requests that consist of business service name and method.

cost assignment The process in JD Edwards EnterpriseOne Advanced Cost Accounting of tracing or allocating resources to activities or cost objects.

cost component In JD Edwards EnterpriseOne Manufacturing, an element of an item's cost (for example, material, labor, or overhead).

credentials A valid set of JD Edwards EnterpriseOne username/password/environment/role, EnterpriseOne session, or EnterpriseOne token.

cross-reference utility services Utility services installed in a BPEL/ESB environment that access EnterpriseOne cross-reference data.

cross segment edit A logic statement that establishes the relationship between configured item segments. Cross segment edits are used to prevent ordering of configurations that cannot be produced.

currency restatement The process of converting amounts from one currency into another currency, generally for reporting purposes. You can use the currency restatement process, for example, when many currencies must be restated into a single currency for consolidated reporting.

cXML A protocol used to facilitate communication between business documents and procurement applications, and between e-commerce hubs and suppliers.

database credentials A valid database username/password.

database server A server in a local area network that maintains a database and performs searches for client computers.

Data Source Workbench An application that, during the Installation Workbench process, copies all data sources that are defined in the installation plan from the Data Source Master and Table and Data Source Sizing tables in the Planner data source to the system-release number data source. It also updates the Data Source Plan detail record to reflect completion.

date pattern A calendar that represents the beginning date for the fiscal year and the ending date for each period in that year in standard and 52-period accounting.

denominated-in currency The company currency in which financial reports are based.

deployment artifacts Artifacts that are needed for the deployment process, such as servers, ports, and such.

deployment server A server that is used to install, maintain, and distribute software to one or more enterprise servers and client workstations.

detail information Information that relates to individual lines in JD Edwards EnterpriseOne transactions (for example, voucher pay items and sales order detail lines).

direct connect A transaction method in which a client application communicates interactively and directly with a server application.

See also batch-of-one immediate and store-and-forward.

Do Not Translate (DNT) A type of data source that must exist on the iSeries because of BLOB restrictions.

dual pricing The process of providing prices for goods and services in two currencies.

duplicate published business services authorization records Two published business services authorization records with the same user identification information and published business services identification information.

embedded application server instance An OC4J instance started by and running wholly within JDeveloper.

edit code A code that indicates how a specific value for a report or a form should appear or be formatted. The default edit codes that pertain to reporting require particular attention because they account for a substantial amount of information.

edit mode A condition of a form that enables users to change data.

edit rule A method used for formatting and validating user entries against a predefined rule or set of rules.

Electronic Data Interchange (EDI) An interoperability model that enables paperless computer-to-computer exchange of business transactions between JD Edwards EnterpriseOne and third-party systems. Companies that use EDI must have translator software to convert data from the EDI standard format to the formats of their computer systems.

embedded event rule An event rule that is specific to a particular table or application. Examples include form-to-form calls, hiding a field based on a processing option value, and calling a business function. Contrast with the business function event rule.

Employee Work Center A central location for sending and receiving all JD Edwards EnterpriseOne messages (system and user generated), regardless of the originating application or user. Each user has a mailbox that contains workflow and other messages, including Active Messages.

enterprise server A server that contains the database and the logic for JD Edwards EnterpriseOne.

Enterprise Service Bus (ESB) Middleware infrastructure products or technologies based on web services standards that enable a service-oriented architecture using an event-driven and XML-based messaging framework (the bus).

EnterpriseOne administrator An actor responsible for the EnterpriseOne administration system.

EnterpriseOne credentials A user ID, password, environment, and role used to validate a user of EnterpriseOne.

EnterpriseOne object A reusable piece of code that is used to build applications. Object types include tables, forms, business functions, data dictionary items, batch processes, business views, event rules, versions, data structures, and media objects.

EnterpriseOne development client Historically called "fat client," a collection of installed EnterpriseOne components required to develop EnterpriseOne artifacts, including the Microsoft Windows client and design tools.

EnterpriseOne extension A JDeveloper component (plug-in) specific to EnterpriseOne. A JDeveloper wizard is a specific example of an extension.

EnterpriseOne process A software process that enables JD Edwards EnterpriseOne clients and servers to handle processing requests and run transactions. A client runs one process, and servers can have multiple instances of a process. JD Edwards EnterpriseOne processes can also be dedicated to specific tasks (for example, workflow messages and data replication) to ensure that critical processes don't have to wait if the server is particularly busy.

EnterpriseOne resource Any EnterpriseOne table, metadata, business function, dictionary information, or other information restricted to authorized users.

Environment Workbench An application that, during the Installation Workbench process, copies the environment information and Object Configuration Manager tables for each environment from the Planner data source to the system-release number data source. It also updates the Environment Plan detail record to reflect completion.

escalation monitor A batch process that monitors pending requests or activities and restarts or forwards them to the next step or user after they have been inactive for a specified amount of time.

event rule A logic statement that instructs the system to perform one or more operations based on an activity that can occur in a specific application, such as entering a form or exiting a field.

explicit transaction Transaction used by a business service developer to explicitly control the type (auto or manual) and the scope of transaction boundaries within a business service.

exposed method or value object Published business service source files or parts of published business service source files that are part of the published interface. These are part of the contract with the customer.

facility An entity within a business for which you want to track costs. For example, a facility might be a warehouse location, job, project, work center, or branch/plant. A facility is sometimes referred to as a "business unit."

fast path A command prompt that enables the user to move quickly among menus and applications by using specific commands.

file server A server that stores files to be accessed by other computers on the network. Unlike a disk server, which appears to the user as a remote disk drive, a file server is a sophisticated device that not only stores files, but also manages them and maintains order as network users request files and make changes to these files.

final mode The report processing mode of a processing mode of a program that updates or creates data records.

foundation A framework that must be accessible for execution of business services at runtime. This includes, but is not limited to, the Java Connector and JDBj.

FTP server A server that responds to requests for files via file transfer protocol.

header information Information at the beginning of a table or form. Header information is used to identify or provide control information for the group of records that follows.

HTTP Adapter A generic set of services that are used to do the basic HTTP operations, such as GET, POST, PUT, DELETE, TRACE, HEAD, and OPTIONS with the provided URL.

instantiate A Java term meaning "to create." When a class is instantiated, a new instance is created.

integration developer The user of the system who develops, runs, and debugs the EnterpriseOne business services. The integration developer uses the EnterpriseOne business services to develop these components.

integration point (IP) The business logic in previous implementations of EnterpriseOne that exposes a document level interface. This type of logic used to be called XBPs. In EnterpriseOne 8.11, IPs are implemented in Web Services Gateway powered by webMethods.

integration server A server that facilitates interaction between diverse operating systems and applications across internal and external networked computer systems.

integrity test A process used to supplement a company's internal balancing procedures by locating and reporting balancing problems and data inconsistencies.

interface table See Z table.

internal method or value object Business service source files or parts of business service source files that are not part of the published interface. These could be private or protected methods. These could be value objects not used in published methods.

interoperability model A method for third-party systems to connect to or access JD Edwards EnterpriseOne.

in-your-face-error In JD Edwards EnterpriseOne, a form-level property which, when enabled, causes the text of application errors to appear on the form.

IServer service This internet server service resides on the web server and is used to speed up delivery of the Java class files from the database to the client.

jargon An alternative data dictionary item description that JD Edwards EnterpriseOne appears based on the product code of the current object.

Java application server A component-based server that resides in the middle-tier of a server-centric architecture. This server provides middleware services for security and state maintenance, along with data access and persistence.

JDBNET A database driver that enables heterogeneous servers to access each other's data.

JDEBASE Database Middleware A JD Edwards EnterpriseOne proprietary database middleware package that provides platform-independent APIs, along with client-to-server access.

JDECallObject An API used by business functions to invoke other business functions.

jde.ini A JD Edwards EnterpriseOne file (or member for iSeries) that provides the runtime settings required for JD Edwards EnterpriseOne initialization. Specific versions of the file or member must reside on every machine running JD Edwards EnterpriseOne. This includes workstations and servers.

JDEIPC Communications programming tools used by server code to regulate access to the same data in multiprocess environments, communicate and coordinate between processes, and create new processes.

jde.log The main diagnostic log file of JD Edwards EnterpriseOne. This file is always located in the root directory on the primary drive and contains status and error messages from the startup and operation of JD Edwards EnterpriseOne.

JDENET A JD Edwards EnterpriseOne proprietary communications middleware package. This package is a peer-to-peer, message-based, socket-based, multiprocess communications middleware solution. It handles client-to-server and server-to-server communications for all JD Edwards EnterpriseOne supported platforms.

JDeveloper Project An artifact that JDeveloper uses to categorize and compile source files.

JDeveloper Workspace An artifact that JDeveloper uses to organize project files. It contains one or more project files.

JMS Queue A Java Messaging service queue used for point-to-point messaging.

listener service A listener that listens for XML messages over HTTP.

local repository A developer's local development environment that is used to store business service artifacts.

local standalone BPEL/ESB server A standalone BPEL/ESB server that is not installed within an application server.

Location Workbench An application that, during the Installation Workbench process, copies all locations that are defined in the installation plan from the Location Master table in the Planner data source to the system data source.

logic server A server in a distributed network that provides the business logic for an application program. In a typical configuration, pristine objects are replicated on to the logic server from the central server. The logic server, in conjunction with workstations, actually performs the processing required when JD Edwards EnterpriseOne software runs.

MailMerge Workbench An application that merges Microsoft Word 6.0 (or higher) word-processing documents with JD Edwards EnterpriseOne records to automatically print business documents. You can use MailMerge Workbench to print documents, such as form letters about verification of employment.

Manual Commit transaction A database connection where all database operations delay writing to the database until a call to commit is made.

master business function (MBF) An interactive master file that serves as a central location for adding, changing, and updating information in a database. Master business functions pass information between data entry forms and the appropriate tables. These master functions provide a common set of functions that contain all of the necessary default and editing rules for related programs. MBFs contain logic that ensures the integrity of adding, updating, and deleting information from databases.

master table See published table.

matching document A document associated with an original document to complete or change a transaction. For example, in JD Edwards EnterpriseOne Financial Management, a receipt is the matching document of an invoice, and a payment is the matching document of a voucher.

media storage object Files that use one of the following naming conventions that are not organized into table format: Gxxx, xxxGT, or GTxxx.

message center A central location for sending and receiving all JD Edwards EnterpriseOne messages (system and user generated), regardless of the originating application or user.

messaging adapter An interoperability model that enables third-party systems to connect to JD Edwards EnterpriseOne to exchange information through the use of messaging queues.

messaging server A server that handles messages that are sent for use by other programs using a messaging API. Messaging servers typically employ a middleware program to perform their functions.

Middle-Tier BPEL/ESB Server A BPEL/ESB server that is installed within an application server.

Monitoring Application An EnterpriseOne tool provided for an administrator to get statistical information for various EntepriseOne servers, reset statistics, and set notifications.

named event rule (NER) Encapsulated, reusable business logic created using event rules, rather that C programming. NERs are also called business function event rules. NERs can be reused in multiple places by multiple programs. This modularity lends itself to streamlining, reusability of code, and less work.

nota fiscal In Brazil, a legal document that must accompany all commercial transactions for tax purposes and that must contain information required by tax regulations.

nota fiscal factura In Brazil, a nota fiscal with invoice information.

See also *nota fiscal.*

Object Configuration Manager (OCM) In JD Edwards EnterpriseOne, the object request broker and control center for the runtime environment. OCM keeps track of the runtime locations for business functions, data, and batch applications. When one of these objects is called, OCM directs access to it using defaults and overrides for a given environment and user.

Object Librarian A repository of all versions, applications, and business functions reusable in building applications. Object Librarian provides check-out and check-in capabilities for developers, and it controls the creation, modification, and use of JD Edwards EnterpriseOne objects. Object Librarian supports multiple environments (such as production and development) and enables objects to be easily moved from one environment to another.

Object Librarian merge A process that blends any modifications to the Object Librarian in a previous release into the Object Librarian in a new release.

Open Data Access (ODA) An interoperability model that enables you to use SQL statements to extract JD Edwards EnterpriseOne data for summarization and report generation.

Output Stream Access (OSA) An interoperability model that enables you to set up an interface for JD Edwards EnterpriseOne to pass data to another software package, such as Microsoft Excel, for processing.

package JD Edwards EnterpriseOne objects are installed to workstations in packages from the deployment server. A package can be compared to a bill of material or kit that indicates the necessary objects for that workstation and where on the deployment server the installation program can find them. It is point-in-time snapshot of the central objects on the deployment server.

package build A software application that facilitates the deployment of software changes and new applications to existing users. Additionally, in JD Edwards EnterpriseOne, a package build can be a compiled version of the software. When you upgrade your version of the ERP software, for example, you are said to take a package build.

Consider the following context: “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.” The process of creating a package build is often referred to, as it is in this example, simply as “a package build.”

package location The directory structure location for the package and its set of replicated objects. This is usually \\deployment server\release\path_code\package\package name. The subdirectories under this path are where the replicated objects for the package are placed. This is also referred to as where the package is built or stored.

Package Workbench An application that, during the Installation Workbench process, transfers the package information tables from the Planner data source to the system-release number data source. It also updates the Package Plan detail record to reflect completion.

Pathcode Directory The specific portion of the file system on the EnterpriseOne development client where EnterpriseOne development artifacts are stored.

patterns General repeatable solutions to a commonly occurring problem in software design. For business service development, the focus is on the object relationships and interactions. For orchestrations, the focus is on the integration patterns (for example, synchronous and asynchronous request/response, publish, notify, and receive/reply).

planning family A means of grouping end items whose similarity of design and manufacture facilitates being planned in aggregate.

preference profile The ability to define default values for specified fields for a user-defined hierarchy of items, item groups, customers, and customer groups.

print server The interface between a printer and a network that enables network clients to connect to the printer and send their print jobs to it. A print server can be a computer, separate hardware device, or even hardware that resides inside of the printer itself.

pristine environment A JD Edwards EnterpriseOne environment used to test unaltered objects with JD Edwards EnterpriseOne demonstration data or for training classes. You must have this environment so that you can compare pristine objects that you modify.

processing option A data structure that enables users to supply parameters that regulate the running of a batch program or report. For example, you can use processing options to specify default values for certain fields, to determine how information appears or is printed, to specify date ranges, to supply runtime values that regulate program execution, and so on.

production environment A JD Edwards EnterpriseOne environment in which users operate EnterpriseOne software.

production-grade file server A file server that has been quality assurance tested and commercialized and that is usually provided in conjunction with user support services.

Production Published Business Services Web Service Published business services web service deployed to a production application server.

program temporary fix (PTF) A representation of changes to JD Edwards EnterpriseOne software that your organization receives on magnetic tapes or disks.

project In JD Edwards EnterpriseOne, a virtual container for objects being developed in Object Management Workbench.

promotion path The designated path for advancing objects or projects in a workflow. The following is the normal promotion cycle (path):

11>21>26>28>38>01

In this path, *11* equals new project pending review, *21* equals programming, *26* equals QA test/review, *28* equals QA test/review complete, *38* equals in production, *01* equals complete. During the normal project promotion cycle, developers check objects out of and into the development path code and then promote them to the prototype path code. The objects are then moved to the productions path code before declaring them complete.

proxy server A server that acts as a barrier between a workstation and the internet so that the enterprise can ensure security, administrative control, and caching service.

published business service EnterpriseOne service level logic and interface. A classification of a published business service indicating the intention to be exposed to external (non-EnterpriseOne) systems.

published business service identification information Information about a published business service used to determine relevant authorization records. Published business services + method name, published business services, or *ALL.

published business service web service — Published business services components packaged as J2EE Web Service (namely, a J2EE EAR file that contains business service classes, business service foundation, configuration files, and web service artifacts).

published table — Also called a master table, this is the central copy to be replicated to other machines. Residing on the publisher machine, the F98DRPUB table identifies all of the published tables and their associated publishers in the enterprise.

publisher — The server that is responsible for the published table. The F98DRPUB table identifies all of the published tables and their associated publishers in the enterprise.

pull replication — One of the JD Edwards EnterpriseOne methods for replicating data to individual workstations. Such machines are set up as pull subscribers using JD Edwards EnterpriseOne data replication tools. The only time that pull subscribers are notified of changes, updates, and deletions is when they request such information. The request is in the form of a message that is sent, usually at startup, from the pull subscriber to the server machine that stores the F98DRPCN table.

QBE — An abbreviation for *query by example*. In JD Edwards EnterpriseOne, the QBE line is the top line on a detail area that is used for filtering data.

real-time event — A message triggered from EnterpriseOne application logic that is intended for external systems to consume.

refresh — A function used to modify JD Edwards EnterpriseOne software, or subset of it, such as a table or business data, so that it functions at a new release or cumulative update level, such as B73.2 or B73.2.1.

replication server — A server that is responsible for replicating central objects to client machines.

Rt-Addressing — Unique data identifying a browser session that initiates the business services call request host/port user session.

rules — Mandatory guidelines that are not enforced by tooling, but must be followed in order to accomplish the desired results and to meet specified standards.

quote order — In JD Edwards Procurement and Subcontract Management, a request from a supplier for item and price information from which you can create a purchase order.

In JD Edwards Sales Order Management, item and price information for a customer who has not yet committed to a sales order.

secure by default — A security model that assumes that a user does not have permission to execute an object unless there is a specific record indicating such permissions.

Secure Socket Layer (SSL) — A security protocol that provides communication privacy. SSL enables client and server applications to communicate in a way that is designed to prevent eavesdropping, tampering, and message forgery.

SEI implementation — A Java class that implements the methods that declare in a Service Endpoint Interface (SEI).

selection — Found on JD Edwards EnterpriseOne menus, a selection represents functions that you can access from a menu. To make a selection, type the associated number in the Selection field and press Enter.

serialize — The process of converting an object or data into a format for storage or transmission across a network connection link with the ability to reconstruct the original data or objects when needed.

Server Workbench — An application that, during the Installation Workbench process, copies the server configuration files from the Planner data source to the system-release number

	data source. The application also updates the Server Plan detail record to reflect completion.
Service Endpoint Interface (SEI)	A Java interface that declares the methods that a client can invoke on the service.
SOA	Abbreviation for *Service Oriented Architecture*.
softcoding	A coding technique that enables an administrator to manipulate site-specific variables that affect the execution of a given process.
source repository	A repository for HTTP adapter and listener service development environment artifacts.
spot rate	An exchange rate entered at the transaction level. This rate overrides the exchange rate that is set up between two currencies.
Specification merge	A merge that comprises three merges: Object Librarian merge, Versions List merge, and Central Objects merge. The merges blend customer modifications with data that accompanies a new release.
specification	A complete description of a JD Edwards EnterpriseOne object. Each object has its own specification, or name, which is used to build applications.
Specification Table Merge Workbench	An application that, during the Installation Workbench process, runs the batch applications that update the specification tables.
SSL Certificate	A special message signed by a certificate authority that contains the name of a user and that user's public key in such a way that anyone can "verify" that the message was signed by no one other than the certification authority and thereby develop trust in the user's public key.
store-and-forward	The mode of processing that enables users who are disconnected from a server to enter transactions and then later connect to the server to upload those transactions.
subscriber table	Table F98DRSUB, which is stored on the publisher server with the F98DRPUB table and identifies all of the subscriber machines for each published table.
superclass	An inheritance concept of the Java language where a class is an instance of something, but is also more specific. "Tree" might be the superclass of "Oak" and "Elm," for example.
supplemental data	Any type of information that is not maintained in a master file. Supplemental data is usually additional information about employees, applicants, requisitions, and jobs (such as an employee's job skills, degrees, or foreign languages spoken). You can track virtually any type of information that your organization needs. For example, in addition to the data in the standard master tables (the Address Book Master, Customer Master, and Supplier Master tables), you can maintain other kinds of data in separate, generic databases. These generic databases enable a standard approach to entering and maintaining supplemental data across JD Edwards EnterpriseOne systems.
table access management (TAM)	The JD Edwards EnterpriseOne component that handles the storage and retrieval of use-defined data. TAM stores information, such as data dictionary definitions; application and report specifications; event rules; table definitions; business function input parameters and library information; and data structure definitions for running applications, reports, and business functions.
Table Conversion Workbench	An interoperability model that enables the exchange of information between JD Edwards EnterpriseOne and third-party systems using non-JD Edwards EnterpriseOne tables.

table conversion An interoperability model that enables the exchange of information between JD Edwards EnterpriseOne and third-party systems using non-JD Edwards EnterpriseOne tables.

table event rules Logic that is attached to database triggers that runs whenever the action specified by the trigger occurs against the table. Although JD Edwards EnterpriseOne enables event rules to be attached to application events, this functionality is application specific. Table event rules provide embedded logic at the table level.

terminal server A server that enables terminals, microcomputers, and other devices to connect to a network or host computer or to devices attached to that particular computer.

three-tier processing The task of entering, reviewing and approving, and posting batches of transactions in JD Edwards EnterpriseOne.

three-way voucher match In JD Edwards Procurement and Subcontract Management, the process of comparing receipt information to supplier's invoices to create vouchers. In a three-way match, you use the receipt records to create vouchers.

transaction processing (TP) monitor A monitor that controls data transfer between local and remote terminals and the applications that originated them. TP monitors also protect data integrity in the distributed environment and may include programs that validate data and format terminal screens.

transaction processing method A method related to the management of a manual commit transaction boundary (for example, start, commit, rollback, and cancel).

transaction set An electronic business transaction (electronic data interchange standard document) made up of segments.

trigger One of several events specific to data dictionary items. You can attach logic to a data dictionary item that the system processes automatically when the event occurs.

triggering event A specific workflow event that requires special action or has defined consequences or resulting actions.

two-way authentication An authentication mechanism in which both client and server authenticate themselves by providing the SSL certificates to each other.

two-way voucher match In JD Edwards Procurement and Subcontract Management, the process of comparing purchase order detail lines to the suppliers' invoices to create vouchers. You do not record receipt information.

user identification information User ID, role, or *public.

User Overrides merge Adds new user override records into a customer's user override table.

value object A specific type of source file that holds input or output data, much like a data structure passes data. Value objects can be exposed (used in a published business service) or internal, and input or output. They are comprised of simple and complex elements and accessories to those elements.

variance In JD Edwards Capital Asset Management, the difference between revenue generated by a piece of equipment and costs incurred by the equipment.

In JD Edwards EnterpriseOne Project Costing and JD Edwards EnterpriseOne Manufacturing, the difference between two methods of costing the same item (for example, the difference between the frozen standard cost and the current cost is an engineering variance). Frozen standard costs come from the Cost Components table, and the current costs are calculated using the current bill of material, routing, and overhead rates.

versioning a published business service Adding additional functionality/interfaces to the published business services without modifying the existing functionality/interfaces.

Version List merge The Versions List merge preserves any non-XJDE and non-ZJDE version specifications for objects that are valid in the new release, as well as their processing options data.

visual assist Forms that can be invoked from a control via a trigger to assist the user in determining what data belongs in the control.

vocabulary override An alternate description for a data dictionary item that appears on a specific JD Edwards EnterpriseOne form or report.

wchar_t An internal type of a wide character. It is used for writing portable programs for international markets.

web application server A web server that enables web applications to exchange data with the back-end systems and databases used in eBusiness transactions.

web server A server that sends information as requested by a browser, using the TCP/IP set of protocols. A web server can do more than just coordination of requests from browsers; it can do anything a normal server can do, such as house applications or data. Any computer can be turned into a web server by installing server software and connecting the machine to the internet.

Web Service Description Language (WSDL) An XML format for describing network services.

Web Service Inspection Language (WSIL) An XML format for assisting in the inspection of a site for available services and a set of rules for how inspection-related information should be made.

web service proxy foundation Foundation classes for web service proxy that must be included in a business service server artifact for web service consumption on WAS.

web service softcoding record An XML document that contains values that are used to configure a web service proxy. This document identifies the endpoint and conditionally includes security information.

web service softcoding template An XML document that provides the structure for a soft coded record.

Where clause The portion of a database operation that specifies which records the database operation will affect.

Windows terminal server A multiuser server that enables terminals and minimally configured computers to display Windows applications even if they are not capable of running Windows software themselves. All client processing is performed centrally at the Windows terminal server and only display, keystroke, and mouse commands are transmitted over the network to the client terminal device.

wizard A type of JDeveloper extension used to walk the user through a series of steps.

workbench A program that enables users to access a group of related programs from a single entry point. Typically, the programs that you access from a workbench are used to complete a large business process. For example, you use the JD Edwards EnterpriseOne Payroll Cycle Workbench (P07210) to access all of the programs that the system uses to process payroll, print payments, create payroll reports, create journal entries, and update payroll history. Examples of JD Edwards EnterpriseOne workbenches include Service Management Workbench (P90CD020), Line Scheduling Workbench (P3153), Planning Workbench (P13700), Auditor's Workbench (P09E115), and Payroll Cycle Workbench.

work day calendar In JD Edwards EnterpriseOne Manufacturing, a calendar that is used in planning functions that consecutively lists only working days so that component and work order scheduling can be done based on the actual number of work days available. A work

	day calendar is sometimes referred to as planning calendar, manufacturing calendar, or shop floor calendar.
workflow	The automation of a business process, in whole or in part, during which documents, information, or tasks are passed from one participant to another for action, according to a set of procedural rules.
workgroup server	A server that usually contains subsets of data replicated from a master network server. A workgroup server does not perform application or batch processing.
XAPI events	A service that uses system calls to capture JD Edwards EnterpriseOne transactions as they occur and then calls third-party software, end users, and other JD Edwards EnterpriseOne systems that have requested notification when the specified transactions occur to return a response.
XML CallObject	An interoperability capability that enables you to call business functions.
XML Dispatch	An interoperability capability that provides a single point of entry for all XML documents coming into JD Edwards EnterpriseOne for responses.
XML List	An interoperability capability that enables you to request and receive JD Edwards EnterpriseOne database information in chunks.
XML Service	An interoperability capability that enables you to request events from one JD Edwards EnterpriseOne system and receive a response from another JD Edwards EnterpriseOne system.
XML Transaction	An interoperability capability that enables you to use a predefined transaction type to send information to or request information from JD Edwards EnterpriseOne. XML transaction uses interface table functionality.
XML Transaction Service (XTS)	Transforms an XML document that is not in the JD Edwards EnterpriseOne format into an XML document that can be processed by JD Edwards EnterpriseOne. XTS then transforms the response back to the request originator XML format.
Z event	A service that uses interface table functionality to capture JD Edwards EnterpriseOne transactions and provide notification to third-party software, end users, and other JD Edwards EnterpriseOne systems that have requested to be notified when certain transactions occur.
Z table	A working table where non-JD Edwards EnterpriseOne information can be stored and then processed into JD Edwards EnterpriseOne. Z tables also can be used to retrieve JD Edwards EnterpriseOne data. Z tables are also known as interface tables.
Z transaction	Third-party data that is properly formatted in interface tables for updating to the JD Edwards EnterpriseOne database.

Index

W