

JD Edwards EnterpriseOne

System Extensibility Guidelines

5.1

Copyright © 2015, 2025, Oracle and/or its affiliates.

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.

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.

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

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software" or "commercial computer software documentation" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware 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 that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about 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 unless otherwise set forth in an applicable agreement between you and Oracle. 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, except as set forth in an applicable agreement between you and Oracle.

Contents

Preface	i
<hr/>	
1 Extensibility	1
Understanding Extensibility	1
Terminology	1
2 Understanding Extensibility	3
Approaches to Offering Extended Solutions	3
Feasibility of the Proposed Extended Solution	6
Application Extensibility Design Approach	7
Cloning	13
Common Extensibility Patterns and Practices	14
Advanced Extensibility Patterns and Practices	17
3 Enabling Extensibility	19
Obtaining a System Code	19
Creating a DLL for Business Functions	19
Understanding Component Approvals and Registration	19
Understanding Object Naming Standards	20
Understanding Data Dictionary Naming Standards	26
Understanding Code Compliance	28
Understanding Public Business Functions and Tools APIs	29
4 Programming Restrictions	31
Programming Restrictions	31
5 Foundational Design Patterns	33
Understanding Foundational Design Patterns	33
Implementing Currency	33
Retrieving Address Book Numbers	34

Using Unknown Item Format	36
Transaction Processing	38
Implementing Record Reservation	51
Using Next Numbers	52
Adding a Date and Time Field	53
Searching Tree Controls	54
6 Application Extensibility Design Patterns	57
Determining an Active System Code	57
Using a Tag File to Add Fields to the Database	58
Adding Processing Options to Existing Applications	58
Adding Hooks to Existing Applications	60
Adding New Versions	62
Adding a Task List and Task	62
Changing Displayed EnterpriseOne Carousel Icons	63
Using Hover Form Extensibility	66
Adding New Auto Suggest Functionality	73
Understanding Data Model	74
7 Understanding Advanced Application Extensibility Design Patterns	77
Adding Fields to a Cache	77
Adding Logic to an Existing Application	77
Delegating Authorization to another Employee	77
8 Understanding Advanced Functional Extensibility Design Patterns	79
Understanding Prebuilt Business Function Extension Points for JD Edwards EnterpriseOne	79
Understanding Prebuilt Business Function Extension Points by System Code	80
9 Localization Extensibility	85
Understanding Localization Extensibility	85
Adding a New Localization Extensibility Point	86
Adding a New Country to an Existing Extensibility Point	88
10 Integration Extensibility	91
Understanding Integration Extensibility	91

Understanding Business Service Extensibility	91
Understanding Real-Time Event Extensibility	93
Understanding Application Integration Architecture Connector Extensibility	94
11 Appendix A - Agile Product Lifecycle Management	99
Background	99
Customer Scenario	99
Business Flows	99
Architecture	100
Environment and Setup	101
Post Install Configuration of Agile PLM-JD Edwards EnterpriseOne PIP	104
Code Changes	107
Naming Conventions	124
Reference Links	126
Known Issues and Workarounds	127
Troubleshooting	128
12 Appendix B - Prebuilt Business Function Extension Points for Transportation Management System	131
Prebuilt Business Function Extension Point for Shipment Tracking	131
Prebuilt Business Function Extension Point for Document Number Generation	133
13 Appendix C - Oracle Data Modeler Setup	137
Oracle Data Modeler Setup	137
14 Appendix D - Creating an EnterpriseOne Page Example	149
Creating an EnterpriseOne Page Example	149

Preface

Welcome to the JD Edwards EnterpriseOne documentation.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

Related Information

For additional information about JD Edwards EnterpriseOne applications, features, content, and training, visit the JD Edwards EnterpriseOne pages on the JD Edwards Resource Library located at:

<http://learnjde.com>

Conventions

The following text conventions are used in this document:

Convention	Meaning
Bold	Boldface type indicates graphical user interface elements associated with an action or terms defined in text or the glossary.
<i>Italics</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
Monospace	Monospace type indicates commands within a paragraph, URLs, code examples, text that appears on a screen, or text that you enter.
> Oracle by Example	Indicates a link to an Oracle by Example (OBE). OBEs provide hands-on, step-by-step instructions, including screen captures that guide you through a process using your own environment. Access to OBEs requires a valid Oracle account.

1 Extensibility

Understanding Extensibility

Extensibility is the process of extending the capabilities of the host system by adding new features. The purpose of this document is to help customers and business partners create new functionality within existing JD Edwards EnterpriseOne applications.

The goals of this document are to provide customers and business partners the tools for:

- Reducing JD Edwards EnterpriseOne system total cost of ownership.
- Easier implementation of customized or extended solutions.
- Eliminating object contention for extended solutions applied from multiple providers.
- Minimizing code overrides when applying ESUs or upgrading.
- Understanding Oracle development standards.
- Providing links to code compliance documentation.
- Gaining knowledge of the tools that assist in managing and merging code changes.

Terminology

These terms are commonly used when discussing extensibility:

Add-in (Add-On, Extension, Plug-In, Snap-In, Plug and Play)—A customization; something that adds functionality to a host application. Usually this is custom code written by the customer or a third-party vendor.

Add-In Model—The inclusion of design features (extensibility patterns) that support the extension of the system or application in the future. These extensibility patterns are part of the standard system architecture.

Composite Applications—An application built by combining multiple functions or services.

Custom Solution—A solution that is developed to meet the needs of a single customer. The solution is developed over the base solution from a point in time. The solution is maintained for that point forward as a separate instance of the solution.

Extended Solution—An extension of the base solution; software that is built on top of the base solution so that it looks like and acts like it is part of the base solution. Both custom solutions and productized solutions can be considered to be an extended solution.

Extensibility—The process of extending the capabilities of the host application by adding new features.

Extensibility Pattern—A design pattern that provides the framework for straightforward addition of functionality to a system at a later date.

Foundational Design Patterns—Design patterns that are utilized by the host solution to provide a consistent architecture for solving the foundational business or technical problems.

Host Application (Base)—An application that is extended. Preferably this application supports extensibility in some way.

Productized Solution—A solution that is developed to meet the needs of multiple customers. Changes to the solution are managed with a change control process that utilizes service packs and versioning to deliver product fixes and upgrades. All customers have access to the same changes and upgrades, which are cumulative.

Software Development Kits (SDK, devkit)—A set of public interfaces (APIs) and associated documentation. These APIs allow a software engineer to more easily create applications that interface and extend the software solution.

Prerequisites

This document assumes an understanding of the JD Edwards EnterpriseOne design tools, Object Management Workbench (OMW), application standards, business function coding standards, and usability standards.

2 Understanding Extensibility

Approaches to Offering Extended Solutions

Several ways exist for extending system capabilities. This section discusses these extensibility approaches:

- Disparate System Integration
- Comparable Solutions Integration
- System Extensibility
- Application Extensibility

Disparate System Integration

In terms of extensibility, this is the integration of two disparate systems that were designed completely independent from one another. This is the most common method for extending solutions. Disparate systems are typically integrated through an integration technology middleware using a standard protocol. The integrations between the systems are difficult to analyze and build. Because the systems remain independent and maintain independent release cycles, the integration requires maintenance in order to upgrade either system. This type of integration makes sense when two best-of-breed solutions that were developed independently bring value to the organization.

Disadvantages:

- Duplication of data.
- The integrations are difficult to analyze, build, and maintain.
- Integration technology or middleware is required to integrate.
- IT staff required to learn multiple technologies.
- User interfaces vary.

Advantages:

- Enables the integration between two best-of-breed systems.
- Standard integrations enable additional integrations.
- Existing technology does not have to be re-implemented.

Examples:

- Integration of the JD Edwards EnterpriseOne system to a quality management system.
- Integration of the JD Edwards EnterpriseOne system to a supply chain planning solution.
- Integration of the JD Edwards EnterpriseOne system to a package delivery tracking solution.
- Integration of the JD Edwards EnterpriseOne system to a navigation solution.

JD Edwards EnterpriseOne Solution:

- JD Edwards EnterpriseOne offers a suite of integration services that enable this type of integration.

See the *Business Services Reference Guide* in the JD Edwards EnterpriseOne Applications library on Oracle Technology Network. <http://www.oracle.com/technetwork/documentation/jdedent-098169.html>

Comparable Solutions Integration

This is a method of extending a solution where the solutions share a common data model and interact with each other through integration or standard business services. The extended solution may reside on a different platform and may be built using a separate technology. The system dependency is in the data model and the integration and business services. The extended system can release independently but may have a dependency on the host system for some enhancements. This type of integration makes sense when the solution has technical requirements that can best be enabled through a different programming language or technology.

Disadvantages:

- Some duplication of data may be required.
- IT staff required to learn multiple technologies.
- User interfaces may vary.

Advantages:

- Productized solution that is designed to be an extension of the JD Edwards EnterpriseOne system.
- Less complex integration that is easier to maintain.
- Solutions reside on technology platform that supports its requirements.

Examples:

- Data collection solution developed to work with the JD Edwards EnterpriseOne system.
- Manufacturing execution system developed to work with the JD Edwards EnterpriseOne system.
- Integration to Vertex.

JD Edwards EnterpriseOne Solution:

- JD Edwards EnterpriseOne offers a suite of integration services that enable this type of integration.

See the *Business Services Reference Guide* in the JD Edwards EnterpriseOne Applications library on Oracle Technology Network. <http://www.oracle.com/technetwork/documentation/jdedent-098169.html>

System Extensibility

This is a method of extending a solution by implementing the solution in the same technology. The solutions reside on the same platform, utilize the same technology, and share a common model. The extensions are implemented in a nonintrusive design method. This type of extensibility makes sense when the technology used by the JD Edwards EnterpriseOne system fits the needs and the cost of developing the system is justified or a solution that fits the business needs does not exist. System extensibility also may provide a means of automating or innovating an existing business processes to obtain a competitive advantage.

Disadvantages:

- Some duplication of data may be required.
- User interface cannot be combined.

Advantages:

- One technology stack to support both the JD Edwards EnterpriseOne system and the extended solution.

- Productized solution that is designed to be an extension of the JD Edwards EnterpriseOne system.
- User Interfaces (UI) have the same look and feel.
- Less complex integration that is easier to maintain.

Examples:

- Tool Management System
- Apprenticeship Management System
- Material Shortage Workbench
- Sourcing Dashboard

JD Edwards EnterpriseOne Solution:

- The JD Edwards EnterpriseOne toolset enables the development of applications that use the same technology.

This document provides guidance for extending the JD Edwards EnterpriseOne solution.

Application Extensibility

This is the most intrusive level of extensibility. Existing applications must be modified to provide a common user interface that enables the user access to the extendible features within the existing interface. This type of extensibility usually is required to customize the host system to fit the need of a particular industry or to support localizations. Similar to system extensibility, application extensibility may provide a means for automating or innovating an existing business process to obtain a competitive advantage.

Disadvantages:

- Most intrusive method.
- Combined UI causes manual maintenance.
- Some duplication of data may be required.

Advantages:

- One technology stack to support both the JD Edwards EnterpriseOne and extended solution.
- Productized solution that is designed to be an extension of the JD Edwards EnterpriseOne system.
- UI has the same look and feel.

Examples:

- China localization.
- Order promise date calculated from an advanced Supply Chain Planning system.
- Vendor-managed inventory.

JD Edwards EnterpriseOne Solution:

- The JD Edwards EnterpriseOne toolset enables the development of applications utilizing the same technology.

This document provides guidance for extending the JD Edwards EnterpriseOne solution.

Feasibility of the Proposed Extended Solution

Your first step should be an evaluation of the feasibility of offering the additional functionality as an extended solution. Some extensions make sense and others should be implemented into the base code of the JD Edwards EnterpriseOne system. The JD Edwards EnterpriseOne product line contains numerous integrated systems. Adding a concept into the existing footprint takes you down an intertwining path that inevitably makes it very difficult, if not impossible, to upgrade to the next release. This applies to both user customization of the software and extensions offered by business partners.

This list includes some basic questions to ask about the feasibility of extending your system:

- Will this solution prevent a future upgrade of the base JD Edwards EnterpriseOne system?
- Will this solution impact the ability to take an ESU?
- What is the impact on testing after taking an ESU or upgrade?
- How does the solution impact my change management strategy?
- For business partner provided solutions, can this solution be easily productized?
- Does the solution already exist in the JD Edwards EnterpriseOne software?
- What manual maintenance burden will be created?
- How can the manual maintenance burden be minimized in the design?
- Does the toolset provide the technology to meet the requirements of the solution?
- Does the benefit of extending the solution outweigh the cost? Consider the additional cost that will be incurred for a subsequent upgrade of the JD Edwards EnterpriseOne system.

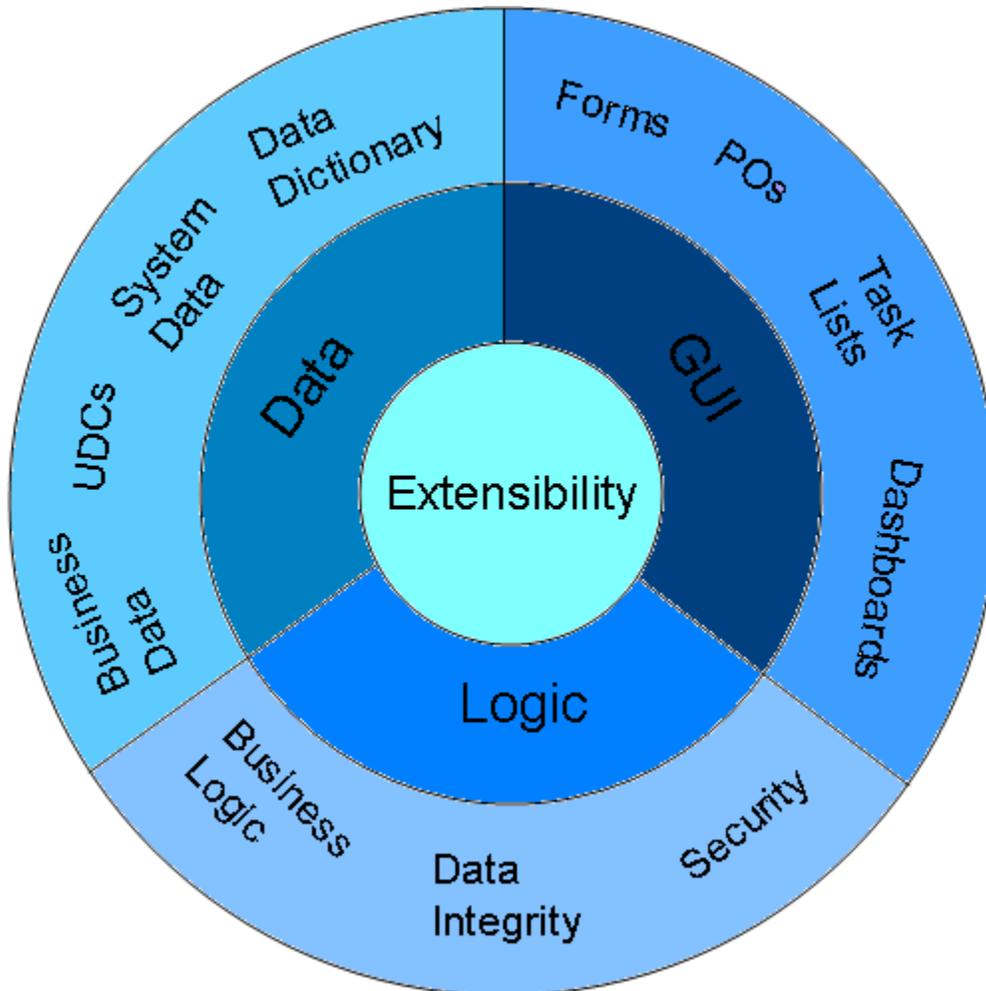
This list identifies the types of enhancements that are best suited for delivery as customized or extended solutions:

- New features delivered as new completely independent applications called from the task list (for example, New Material Shortage Inquiry).
- New systems that integrate into the base JD Edwards EnterpriseOne system and require minimal additional setup data and are run either directly from the task list or are integrated through hooks to the base JD Edwards EnterpriseOne system.
- New reports.
- New dashboards.
- New rapid start profiles.
- New BI Publisher reporting solutions.
- New portals or new portal components.
- New localizations.
- New workflows.
- New integrations.

Application Extensibility Design Approach

You should analyze the proposed solution in terms of the elements that are being used for the extension. Look at the extensions that are required for the data, logic, and GUI. Try to incorporate a design approach that requires only the addition of data or creation of new objects.

This graphic provides an overview of the different elements eligible for extension. For example, to extend data, one might choose to use a new UDC associated with a new data dictionary item.



The following scenarios compare design approaches for extending functionality within the existing footprint. The level of intrusiveness is dependent on the design approach that is selected; however, tradeoffs to usability and performance should also be considered.

Scenario A

The extended functionality requires that additional attributes be added to the Item Master table and requires the user to set the value of the new attributes prior to using the new functionality. How will the values of the new attributes be populated?

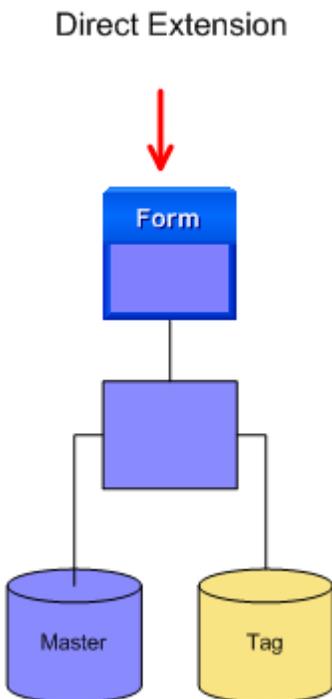
Assumptions:

- A tag file is added to the Item Master table to store the new attributes.
- All business logic required to support validation or calculation of the additional attributes is implemented as unobtrusively as possible through data or new business functions.

Alternative Solutions:

- Direct Extension

Extend the existing application by adding the new item attributes onto the existing form that updates the Item Master table. Add logic to the tag file to maintain the new attributes when the Item Master record is maintained. This diagram provides a high-level overview of this solution:

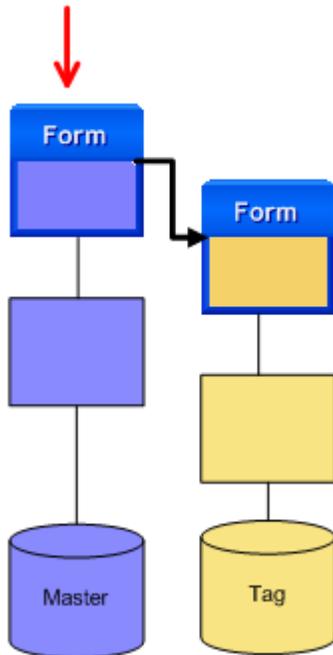


- Intrusive—logic to extend must be manually reapplied when the base object is modified.
- If multiple providers extend the same base object, additional analysis is required to merge the extensions.
- Connected Form

The item attributes are placed on a new form. A hook is created from the existing application to prompt users to maintain the new attributes when they maintain the Item Master table. This assumes that the new form is

created as a separate application. If you add a new form to the existing application, you defeat the purpose of separating the logic. This diagram provides a high-level overview of this solution:

Connected Form

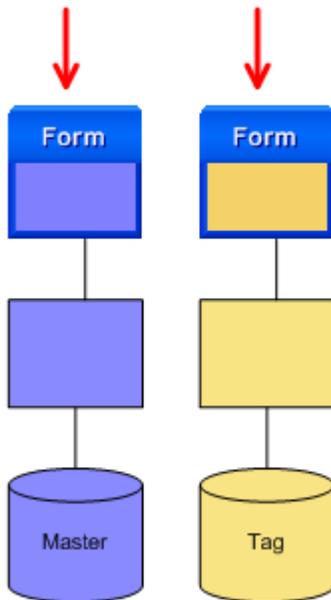


- Less intrusive—logic to connect the new form must be reapplied when the base object is modified.
- Hooks provided by multiple providers could collide. Collisions must be negotiated and reworked by one of the providers.
- User interface is less intuitive, but requires the user to press Save on both forms to save data to the master file and press Save on the extended form to save the data in the tag file.
- Data integrity must be managed; for example, deleting master records or changing the key.

- Separate User Interface

The item attributes are placed in a separate application that is called separately from the task list or menu. This diagram provides a high-level overview of this solution:

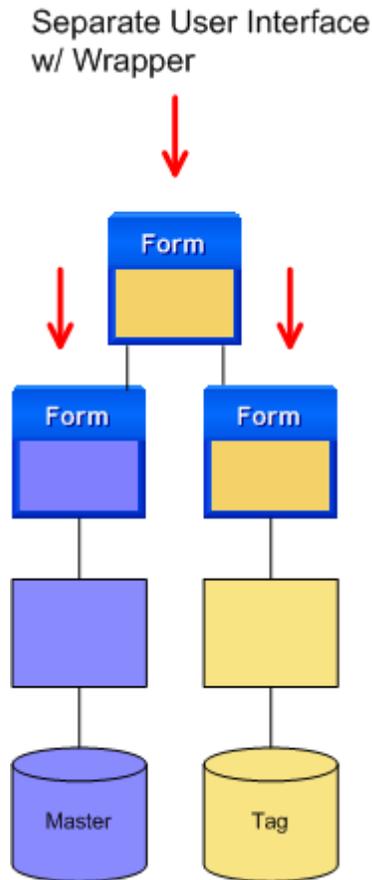
Separate User Interface



- Users must access two separate user interfaces to maintain the data.
 - Data integrity must be managed; for example, deleting master records or changing the key.
 - Multiple providers can use this method to extend without collision.
- Separate User Interface with Wrapper

The item attributes are placed in a separate application that is called separately from the task list or menu. You create a workflow or wizard application that calls the existing Item Master application and then calls a new

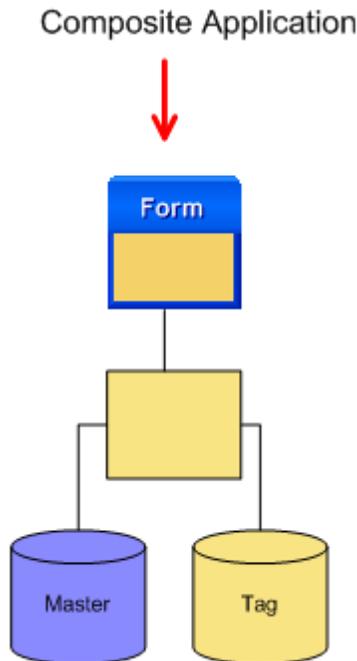
form that prompts users to add the additional attributes. This diagram provides a high-level overview of this solution:



- Data integrity must be managed.
- Extensions provided by more than one provider cannot be delivered with a common wrapper.
- User interface is less intuitive—the form updating the master file already contains a Save button. An additional save button must be added to either the wrapper or the form that updates the tag file.

- Composite Application

Create a new application that combines the existing functionality with the new functionality. This diagram provides a high-level overview of this solution:



- Modifications made to the base code potentially would have to be applied to the composite application. Less risk of this if the business logic is separated from the GUI in the base JD Edwards EnterpriseOne application.
- Extensions provided by more than one provider cannot be delivered in a combined composite application. Provider A is not aware of the extensions made by Provider B.
- Intuitive user interface—one save button on the new composite application to update both the master and the tag file.

- Alternative Solution

An alternative solution that is available for customers only, is to use fields that are already in the Item Master table. The Item Master table contains category codes and user reserved fields that you can define. If these fields suffice for holding the required information, they can be used instead of adding a new table. You can use jargon to override the text that is displayed on the screen to give the fields a more meaningful title.

Scenario B

The extended functionality provides an alternate calculation of the order promise (OP) date at the time of sales order entry.

Assumptions

- The current calculation is directly in the main code line and not provided as a separate business function in the base code.
- The new calculation is created as a new business function to minimize the customized code that is put into base.

Alternative Solutions

- Direct Modification to the Code

The event rules (ER) or business function code is modified to remove the current calculation of the OP date and call the new business function to calculate the order promise date.

- Intrusive—logic to extend must be manually reapplied when the base object is modified.
- If multiple providers extend the same base object, additional analysis is required to merge the extensions.

- Hook with Recalculation

The existing code path to calculate the order promise date and create the sales order is not modified. However, a hook is added to the Save and Close Post Button Clicked event of the Sales Order Entry application to recalculate and update the order promise date.

- Less intrusive—logic to extend must be manually reapplied when the base object is modified, but there will be less logic to modify.
- If multiple providers extend the same base object, additional analysis is required to merge the extensions.
- Negatively impacts performance. Additional I/O is required to update the records with the recalculated value.

- Batch or Subsystem Job

The existing code path to calculate the order promise date and create the sales order is not modified. A regularly scheduled batch job or a subsystem job is run to repromise and update the sales order.

- Nonintrusive—the logic to repromise the order is in a new object.
- Repromising the order in a batch job may not fit the needs of the business process.

Scenario C

The requirements of the extended solution require the Address Book Number field size to be increased.

This type of enhancement is highly intrusive and is best implemented into the base JD Edwards EnterpriseOne system.

Cloning

In software development, cloning is when an object is copied to create a duplicate copy with a different name or identifier. Applications are cloned as either a starting point for creating a new application that has a similar architectural structure, or applications are cloned to extend the functionality of the existing application while preserving a version of the original code.

Cloning an application such as a report for the purpose of creating a completely new and different report is technically very safe. The reason for cloning in this case is to speed up the efficiency of creating a new application by copying an existing one. From this point forward, the applications are managed independently from one another.

The recommended methodology for extending a solution is to create hooks in the host application in a nonintrusive manner. Consider the following before cloning an object for the purpose of extending the solution:

- How will the cloned object be managed for maintenance or upgrade?

- Changes delivered by JD Edwards EnterpriseOne for maintenance or upgrade are defined for the original object, not the cloned object. Merging these changes to a cloned object is difficult to plan because the level of complexity is unknown until the merge is in progress. In addition, specification changes are not identified by the merge tool.
- If your extensions are implemented in a nonintrusive manner, these changes can be documented and reapplied as part of the maintenance and upgrade process. The best practices advocated in this guide are to use hooks to extend the functionality of the host application. This method provides a definable and repeatable process that you can plan.
- If two or more extended solutions are created in separate cloned applications, how will the two separate extensions be integrated?

This creates a process that does not scale and quickly becomes unmanageable. You end up with the burden of integrating the various extended solutions. This problem does not show up until multiple extended solutions are created by multiple providers. Here is the scenario:

- Solution Provider A clones Sales Order Entry (P4210) creating PQ404210 and adds an extended solution to the cloned application.
- Solution Provider B clones Sales Order Entry (P4210) creating PQ884210 and adds an extended solution to the cloned application.
- You want both extended solutions.

You now have three Sales Order Entry applications (P4210) where JD Edwards EnterpriseOne changes will be deployed—P4210, PQ404210, and PQ884210. Before cloning the application, consider how these two extensions will be merged, and how the host application logic will be supported.

- Cloning for a customized solution that is implemented directly onto the original host application as a customized solution is less problematic. This solution is more manageable because there is only one cloned application that becomes the new host application. Assuming that other extended solutions provided by a third party followed the recommendations in this document, the hooks provided for the second extended solution can be applied to the new host application.

Common Extensibility Patterns and Practices

This section identifies some of the typical approaches to extending the JD Edwards EnterpriseOne system without requiring intrusive modifications to core applications or augmentation of the data model.

User Reserved Data Items

JD Edwards EnterpriseOne provides a standard set of user-defined fields to allow the user to easily extend the solution within the limits of the provided fields. You can easily add additional informational attributes to the table using the user-reserved fields. These fields are reserved for customer use and should not be used for business partner extended solutions unless the business partner is implementing a custom solution on the behalf of the customer.

Note: Some tables contain future use fields. These fields are reserved for future use by JD Edwards EnterpriseOne.

Category Codes

Category codes exist in a number of tables. The category code and the allowed values are user definable. You can use these category codes to extend the information stored in the table. Category codes can be made into a required field by removing blank as an allowed value from the connected UDC table.

Extension Tables

Extension tables are new tables that are created to complement, break down, summarize, or link information within existing JD Edwards EnterpriseOne tables.

- Tag table

You create a tag table to append data to another table. The key to the tag table is that it is a duplicate of the master table. The tag table may or may not have a duplicate record for every record in the master table depending on the logic that you implement. Ideally, a duplicate record is created only as needed. When possible, tag tables should be removed or combined to remove the reliance on joins over more than two tables.

- Detail table

If the data required for the extended solution must be kept at a more detail level than the host table, consider adding a detail table.

- Header (Summary) table

If the data required for the extended solution is at a summary level to the information stored in the host table, consider adding a header or summary table. This saves on the database I/O required to update the information if it is modified.

- Cross-Reference (Link) table

Creating a table to cross-reference one business entity to another can change a one-to-one relationship into a one-to-many or many-to-many relationship. Consider that the Sales Order application has a field to identify the work order (WO) and WO contains information to identify the sales order line it is fulfilling. You use a cross-reference table to connect the WO to two or more sales order lines.

Utilizing Existing Public Business Functions

Using existing business functions can greatly decrease the amount of time it takes to create extended solutions. In addition, the use of existing business functions ensures that functionality in the extended solutions behaves consistently with the base system. The business functions provided by JD Edwards EnterpriseOne for maintaining master tables and performing standard business transactions should be used to ensure consistency, data integrity, and future growth.

Utilizing Existing Search and Select Applications

The base JD Edwards EnterpriseOne system contains numerous search and select applications that provide lookup capability over the majority of the setup and master data tables. Existing search and select applications are rarely modified. Search and select applications can be incorporated into the extended solution to improve the usability of the applications by providing standard search and select capabilities.

Customizing a Search and Select Application

Creating a search and select application that is specific to the business needs can improve the time and accuracy of completing the task. An existing search and select application that is similar can be copied and modified to create a new search and select application. You also can create a User Defined Code (UDC) table to create a custom search and select lookup and validation for data entry. You can attach the custom search and select application through a change to the data dictionary (DD).

Calculation Plug-Ins

Some applications within JD Edwards EnterpriseOne are enabled for users to plug-in their own custom calculation. A template business function with the default calculation is provided. The custom calculation function must use the same data structure as the template function for passing the values between the calling application and the function. The custom calculation can then be enabled through different means, depending on its application.

For example, within the JD Edwards Configurator module, a user can model a custom calculation function using the sample business function, B32EXT, and enable it through Assembly Inclusion Rules.

Advanced Pricing (B4500210), Transportation Management (B4900470, B4900920, D4901080B, D4900830, D4901280A, D4901290A), and Fulfillment Management (B4277750, B4277788) provide similar extensibility options.

User Defined Code Tables

UDC tables allow for the addition of new lookup and validation tables that are enabled through the extension of the data without having to extend the data model. You can create a new UDC table under the assigned system code or customer user system code. You add values to the table through the standard UDC maintenance application. You can then set up new DD items to enable lookup and validation from the UDC table when the DD item is used on an interactive application.

In addition to adding a new UDC table, second description and special handling code can be used under certain circumstances. Typically, you use these fields to group or flag UDC values to dictate how a business process behaves based on the setting. These fields can be used on UDC tables that you own. You can use these fields on JD Edwards EnterpriseOne owned UDC tables with some risk. The risk is overlaying the fields with a fix or enhancement. Business partners should not use the fields in JD Edwards EnterpriseOne owned UDC tables for extended solutions.

Note: Hard-coded user-defined codes should not be removed or modified. The UDC value, second description, and special handling code may be used to hard code Boolean statements that control the logic flow.

Using a Standard Data Dictionary Item

Standard base DD items should be used in the creation of new tables when the data item that is added to the table is the same data element. The base DD item is already enabled for DD validation and lookup functionality.

Note: The data dictionary item should not be modified.

Using Jargon and Vocabulary Overrides

Renaming the fields to fit the implementation is a simple way to customize the software. Jargon and vocabulary overrides are standard features in the toolset that enable this type of customization.

Jargon enables you to replace the DD title that appears on forms and reports with terminology that is specific to the needs of your business. For example, replacing *Branch* with *Facility* or *Location* gives specific meaning to how *Branch* is defined and used for a particular implementation of the software. Jargon is defined for a DD item at the system-code level.

Vocabulary overrides also allow for customization of the text that is presented to the user. Vocabulary overrides are defined at the form or report level.

Advanced Extensibility Patterns and Practices

The following approaches are less obvious, more invasive alternatives to extending the JD Edwards EnterpriseOne system.

Plug and Play Components

Several applications are enabled for extensibility through plug and play components that are embedded in the existing applications to enable extensibility for localizations. If a host application has the plug and play component already embedded in the needed location within the host application, the software can be modified through the plug and play or country server components. This method requires a modification to the host components but is less intrusive than modifying the event rules (ER) within the application.

Adding a Value to a Processing Option

Instead of adding a new processing option, determine whether a new value can be utilized for an existing processing option without changing the code. This works well for processing options that are validated by UDC tables.

Note: This method is not recommended unless both the processing option template and UDC table that are used to validate belong to your system. This method is used frequently within the software to deliver extended functionality post GA in an ESU. The need typically arises when a customer needs a variation on the functionality that is delivered in the base code. Do not think that it is safe to add a value to an Oracle owned processing option just because it was not shipped in base.

Supplemental Data

Supplemental data is a table that is designed to allow for extended data to be appended to a table without creating a tag file. Information on supplemental data can be found in the JD Edwards EnterpriseOne Applications Implementation guides that support supplemental data. There are some considerations for collisions in the definition of the data that need to be considered. Naming conventions to prevent collisions in the supplemental data table have not been defined. In general, JD Edwards EnterpriseOne does not plan to add additional uses for this table. Considerations for collisions is for customizations and business partner extensions.

3 Enabling Extensibility

Obtaining a System Code

Having your own system code ensures that your objects, versions, and data dictionary items will not be overwritten by JD Edwards EnterpriseOne updates. Naming conventions for customer and business partner objects are based on a unique system code. System codes can be two to four characters long; however, JD Edwards EnterpriseOne assigned system codes use only three characters to allow three to four characters to be available for unique identification of objects in Object Management Workbench (OMW).

System codes 55-59 are reserved for custom solutions implemented by customers. System codes 55-59 (and all derivatives such as 55S, 57RF, and so on) should not be used by partners to create solutions. (Version 5 Update)

System codes assigned to business partners for the purpose of offering extended solutions consist of 3 characters. These system codes must be a letter followed by two numbers; for example, Z49. This system code naming convention avoids duplicate objects or data. An example of a potential data contention issue is if system codes 60 and 60P are assigned, data dictionary items 60PQTY and 60PQTY could be created.

Note: Business partners can request a unique system code for an extended solution that is intended to be productized by clicking the following link and following the process described on the site. You must sign on to Oracle Partner Network. (Version 5 Update) <http://www.oracle.com/partners/en/products/applications/jde-enterprise-one/develop-solutions/index.html>

System codes 60-69 and all derivatives (such as 60S, 67RF, and so on) are reserved for Oracle Consulting and should not be used by customers or partners. (Version 5 Update)

Creating a DLL for Business Functions

Create a dynamic link library (.dll) for your extended solution business functions. In order to accomplish this, assign the business function to a .dll name that is the same as your solution's system code.

Understanding Component Approvals and Registration

Some components require approval to avoid contention between new JD Edwards EnterpriseOne solutions being developed and solutions developed for commercial purposes. Approval and registration are not required for custom solutions implemented for customers.

These components require approval by JD Edwards EnterpriseOne:

- Adding indexes on existing tables.
JD Edwards EnterpriseOne permanently adds an index to the JD Edwards EnterpriseOne system or asks you to add one of the naming standards discussed later in this chapter.

- Adding a new business entity that drives record reservation.
- Adding a new country code for localizations.

Other components require registration with Oracle. Oracle tracks the usage of open source and third-party solutions that are used within the Oracle systems.

These components require registration with Oracle:

- Using the term, open source.
- Embedding third-party solutions into a business partner provided solution.

Understanding Object Naming Standards

JD Edwards EnterpriseOne object naming standards are provided in these JD Edwards EnterpriseOne guides:

- Development Guidelines for Application Design Guide in the JD Edwards EnterpriseOne Tools library on Oracle Technology Network. <http://www.oracle.com/technetwork/documentation/jdedent-098169.html>
- Development Standards for Business Function Programming Guide in the JD Edwards EnterpriseOne Tools library on Oracle Technology Network. <http://www.oracle.com/technetwork/documentation/jdedent-098169.html>

Some additional guidelines and modifications are necessary for customer and business partner development. This section discusses these modifications.

Note: Naming conventions are extremely important in offering extended solutions. If the naming conventions in the following tables are not followed, object contention will occur. Please follow these naming conventions strictly. Pay specific attention to business functions naming standards that require naming standards to be followed for both the Object Librarian name and the business function name.

Object Naming Conventions

Object Librarian limits object names to 8 characters, and object names must be unique. To ensure that your objects are unique, you use your system code in the object name. This table defines naming conventions for your Object Librarian type:

Object Librarian Type	Object Naming Standard	Description
Table	FXXXZZZ Example: FQ55100	F = defines the object as a file or table. XXX = your system code. ZZZ = a sequential number for your file.
Business View	VXXXZZZZ Example: VQ551000	V = defines the object as a UBE. XXX = your system code. ZZZZ = the table or joined tables name or sequential number for your business view.

Object Librarian Type	Object Naming Standard	Description
		The business view name should be the same as the file or table except for the first character. For example, table FQ5501 has business views VQ5501A, VQ5501B, VQ5501C, and so on.
Table Conversions	R89XXXZZZZ Example: R89Q551000	R89 = defines the object as a table conversion. XXX = your system code. ZZZZ = a sequential number for your table conversion.
Processing Option Template	TXXXZZZZ Example: TQ551000	T = defines the object as a processing option template. XXX = your system code. ZZZZ = the name of your new application or UBE. The processing option name should be the same as the application or UBE except for the first character. For example, application PQ551000 has processing option template TQ551000.
Interactive Application or Portal Component	PXXXZZZZ Example: PQ551000	P = defines the object as an application or portal component. XXX = your system code. ZZZZ = a sequential number for your application or portal component.
Universal Batch Engine (UBE)	RXXXZZZZ Example: RQ551000	R = defines the object as a UBE. XXX = your system code. ZZZZ = a sequential number for your UBE.
Media Objects	GTXXXZZZ Example: GTQ55100	GT = defines the object as a media object. XXX = your system code. ZZZ = a sequential number for your media object. The sequential number that is defined for the media object is typically the file number of the business entity that the media object is extending. For example, the Work Orders application (F4801) has GT4801 for the media object name.
Workflow	KXXXZZZ Example: KQ55100	K = defines the object as a workflow. XXX = your system code. ZZZ = a sequential number for your workflow.

Object Librarian Type	Object Naming Standard	Description
Workflow Data Structure	<p>WFXXXXZZ</p> <p>Example: WFQ55100</p>	<p>WF = defines the object as a workflow data structure.</p> <p>XXX = your system code.</p> <p>ZZZ = an optional, unique number that identifies the workflow data structure.</p> <p>The workflow data structure should have the same name as the workflow except for the identifying characters at the start of the name. For example, workflow KQ55100 has workflow data structure WFQ55100.</p>
C Business Function	<p>BXXXXZZZ</p> <p>Example: BQ551000</p>	<p>B = defines the object as a C business function (BSFN).</p> <p>XXX = your system code.</p> <p>ZZZZ = a sequential number for your business function.</p> <p>BSFNs can be created in either C programming language or named event rule (NER) business logic. BSFNs are numbered by tens. The numbers are unique between C functions and NER functions.</p>
Named Event Rule (NER) Business Function	<p>NXXXXZZZ</p> <p>Example: NQ551000</p>	<p>N = defines the object as a NER business function.</p> <p>XXX = your system code.</p> <p>ZZZZ = a sequential number for your business function.</p> <p>BSFNs can be created in either C programming language or NER business logic. BSFNs are numbered by tens. The numbers are unique between C functions and NER functions.</p>
BSFN Data Structure	<p>DXXXXZZZ</p> <p>Example: DQ551000</p>	<p>D = defines the object as a data structure.</p> <p>XXX = your system code.</p> <p>ZZZZ = a sequential number for your business function.</p> <p>The data structure name should be the same as the business function name except for the first character. Business function BQ550010 has data structure DQ550010.</p>
Task Lists	<p>GXXX</p> <p>or</p> <p>GXXXXZZZ</p> <p>Example: GQ55</p> <p>or</p> <p>GQ551000</p>	<p>G = defines the object as a task list.</p> <p>XXX = your system code.</p> <p>ZZZZ = optional, unique number that identifies the various task lists.</p>

Object Librarian Type	Object Naming Standard	Description
Report Definition	RDXXXXZZZ Example: RDQ55100	RD = defines the object as a report definition. XXX = your system code. ZZZ = a sequential number for your report definition. The report definition name should be the same as the UBE producing the XML except for the first two letters. Report RQ55300 has report definition RDQ55300.
EnterpriseOne BI Publisher Templates	TPXXXXXXXXTYY Example: tp03b311tra1	TP = defines the object as an EnterpriseOne BI Publisher template. XXX = your system code. ZZZZ = three or four characters to identify the associated UBE. TT = template type (UDC H95/XP). YY = different outputs associated with the UBE.

Other Naming Conventions

Other types of objects are defined in programs other than Object Librarian. When you develop extended solutions, follow the naming conventions in this table to ensure that your objects are not overwritten during upgrades and ESUs.

Type	Naming Standard	Description	Notes
System Code	XXX	XXX = your system code	System Codes 55x–59x are reserved for customers. System codes 60x–69x are reserved for Oracle Consulting. (Version 5 Update) The system code assigned to business partners cannot begin with a two-character system code that is already in place. Business partner system codes should be assigned as a three character system code starting with a letter. For example, using system codes 60 and 60P could result in duplicate data dictionary items such as 60PQTY and 60PQTY. (Version 5 Update)

Type	Naming Standard	Description	Notes
			More details on system codes can be found under <i>Obtaining a System Code</i> . (Version 5 Update)
.dll	<p>External:</p> <p>XXX</p> <p>Example: Q55, Q72</p> <p>Internal: JD Edwards:</p> <p>CYYY</p> <p>Example: CMFG, CINV</p>	<p>External:</p> <p>XXX = your system code or approved acronym.</p> <p>Internal JD Edwards:</p> <p>C = defines the .dll as an application development .dll.</p> <p>YYY = Unique identifier.</p>	<p>An OMW setting on business functions.</p> <p>Customers and business partners must use their system code.</p>
UDC Table	<p>SY: XXX</p> <p>RT: YY</p>	<p>XXX = your system code.</p> <p>YY = a unique UDC table.</p>	N/A
Processing Option Versions	<p>External:</p> <p>YXXXXZZZ</p> <p>Example: YQ550001</p> <p>Internal JD Edwards:</p> <p>ZJDEZZZZ</p> <p>XJDEZZZZ</p> <p>RISZZZZ</p> <p>Example: ZJDE0001</p>	<p>External:</p> <p>Y = defines the object as an external processing option version.</p> <p>XXX = your system code.</p> <p>ZZZZ = a sequential number.</p> <p>Internal JD Edwards:</p> <p>ZJDE, XJDE, RIS = defines the object as an internal processing option version.</p> <p>ZZZZ = a sequential number.</p>	<p>Versions beginning with Z, X, and RIS are reserved for Oracle JD Edwards EnterpriseOne. Versions beginning with these letters are handled with specific logic. To avoid conflicts, create versions using Y followed by your system code.</p>
Combo Box	<p>External:</p> <p>FC XXX YYYYYYYYYY</p> <p>Example:</p> <p>FC Q55 Form Control</p> <p>Internal JD Edwards:</p> <p>FC YYYYYYYYYY</p> <p>Example:</p> <p>FC Form Control</p>	<p>External:</p> <p>FC = Nomenclature for form control within the ER for Form Design Aid (FDA).</p> <p>XXX = your system code.</p> <p>YYYYYYYYYY = Unique name.</p> <p>Internal JD Edwards:</p> <p>FC = Nomenclature for form control within the ER for FDA.</p>	N/A

Type	Naming Standard	Description	Notes
		YYYYYYYYY = Unique name.	
Index Name	<p>External: XXX Field 1, Field 2, ... Example: Q55 DOCO, LINE</p> <p>Internal JD Edwards: Field 1, Field 2, ... Example: DOCO, LINE</p>	<p>External: XXX = your system code. Field 1, Field 2 = data items that make up the index.</p> <p>Internal JD Edwards: Field 1, Field 2 = data items that make up the index.</p>	Cannot exceed 19 characters. If the name exceeds 19 characters, the table will not build and business functions will not compile. Do not use special characters.
Function Name (NERs and C Business Functions)	<p>External: XXXAlphaName Example: Q55CalculateQuantity</p> <p>Internal JD Edwards: ActionVerbAlphaName Example: CalculateQuantity</p>	<p>External: XXX = your system code. AlphaName = Description of the function.</p> <p>Internal JD Edwards: ActionVerb = Standard list of action verbs. AlphaName = Description of the function.</p>	This name for C business functions and NERs must be unique. In this example, if the system code of Q55 was not used in front of the function name, it is very likely that Oracle would use the same function name, which would cause a problem when you upgrade the product.
Country Server Function Name (NER)	<p>External: XXX-YYYYYYY - CS - Process Localization Requirements</p> <p>Internal JD Edwards: YYYYYYY - CS - Process Localization Requirements</p>	<p>External: XXX = your system code. YYYYYYY = the host program name. - CS - = defines the object as a country server.</p> <p>Internal JD Edwards: YYYYYYY = the host program name. - CS - = defines the object as a country server.</p>	A country server is a standard component that is used for localizations.
Plug and Play Function Name (NER)	<p>External: XXX-YYYYYYY - Plug and Play</p> <p>Internal JD Edwards:</p>	<p>External: XXX = your system code. YYYYYYY = the host program name.</p>	A plug and play is a standard component that is used for localizations.

Type	Naming Standard	Description	Notes
	YYYYYYY - Plug and Play	<p>- Plug and Play = defines the object as a plug and play business function.</p> <p>Internal JD Edwards:</p> <p>YYYYYYY = the host program name.</p> <p>- Plug and Play = defines the object as a plug and play business function</p>	
Plug and Play Data Structure Name	<p>External:</p> <p>XXX-YYYYYYY - Plug and Play</p> <p>Internal JD Edwards:</p> <p>YYYYYYY - Plug and Play</p>	<p>External:</p> <p>XXX = your system code.</p> <p>YYYYYYY = the host program name.</p> <p>- Plug and Play = defines the object as a plug and play data structure.</p> <p>Internal JD Edwards:</p> <p>YYYYYYY = the host program name.</p> <p>- Plug and Play = defines the object as a Plug and Play data structure.</p>	A plug and play is a standard component that is used for localizations.

Understanding Data Dictionary Naming Standards

Data dictionary items not only define and describe data, but they also can trigger the runtime engine to react or process in certain ways. Online help, error messages, term substitutions for different industries, and translations are all tied to data dictionary items.

Adding a New Data Dictionary Item

Do not modify existing data dictionary items. The better practice is to add a new data dictionary item. All data dictionary items must have a unique alias and data item name.

Before you add a new data dictionary item, consider the following:

- Decimal Places.
- Display Decimals.
- Amount fields that are used for calculations, such as costing fields, should be 29 decimals with 9 display decimals. This minimizes rounding issues.

DD Type	Naming Standard	Description	Notes
	or SXXXXYYYY00 Internal JD Edwards: SYYYYYYYY00 Example: S480101	YYYYYYYY = the program name. You must include your system code if you are adding to a host application. 00 = a sequential number for your processing option template. Internal JD Edwards: S = defines the object as processing option help text. YYYYYYYY = the program name. 00 = a sequential number for your processing option template.	If you add a processing option to an application that was added as part of the extended solution, the program name should already include your system code.

Item Classes

Item classes are used to group data dictionary items together. For example, the item class QTYINV is used to group quantity fields. This allows for all of the fields within the item class to have the same decimal precision. A user sets the precision for all quantity fields in that item class. Any data dictionary item may be added to an existing item class. Proper analysis should be completed when adding data dictionary items to see if they need to be added to an existing item class.

Adding an item class is recommended when adding a new system. Amounts or quantity fields are the usual data dictionary items that are added to the item class. The name of the item class should follow the recommended naming convention of adding your system code to the beginning of the name.

Understanding Code Compliance

Information about coding standards can be found in usability standards, application standards, and business function standards guides and documentation.

Usability Standards

Usability standards ensure that the extended solutions have the same look and feel as the JD Edwards EnterpriseOne developed solution. Extended solutions should be based on JD Edwards EnterpriseOne base software designs to achieve the same look and feel as the JD Edwards EnterpriseOne base product.

See *User Interface Guidelines for Creating EnterpriseOne Applications Forms*.

Application Standards

Application standards exist to create a consistency in how the applications are constructed. In addition, certain features are enabled through the proper construction of the application (that is, transaction processing, 21CFR11, and media objects).

See the *Development Guidelines for Application Design Guide* in the JD Edwards EnterpriseOne Tools library on Oracle Technology Network. <http://www.oracle.com/technetwork/documentation/jdedent-098169.html>

Business Function Standards

Business function standards exist to ensure that the code is thread safe, supports Unicode, and will compile on the various platforms.

See the *Development Guidelines for Application Design Guide* in the JD Edwards EnterpriseOne Tools library on Oracle Technology Network. <http://www.oracle.com/technetwork/documentation/jdedent-098169.html>

Performance

You should evaluate your extensions for performance and memory leaks. Performance Workbench and Kernel Resource Manager can help with this task.

See "Troubleshooting the Enterprise Server Processing" in the *Server and Workstation Administrator Guide*, in the JD Edwards EnterpriseOne Tools library on Oracle Technology Network. <http://www.oracle.com/technetwork/documentation/jdedent-098169.html>

See JD Edwards EnterpriseOne Performance Library (Document ID 978813.1) on My Oracle Support. <https://support.us.oracle.com/oip/faces/secure/km/DocumentDisplay.jspx?id=978813.1> This document provides links to various performance material published by JD Edwards EnterpriseOne.

Understanding Public Business Functions and Tools APIs

When available, master business functions, existing application APIs, and exiting tools APIs should be used to perform standard logic. For purposes of solution stability and performance, the use of the Oracle JD Edwards EnterpriseOne business functions must be consistent with their use in the base software. Public functions include, but are not limited to utility functions, application APIs, master business functions, and file servers.

See

APIs and Business Functions Guide in the JD Edwards EnterpriseOne Tools library on Oracle Technology Network. <http://www.oracle.com/technetwork/documentation/jdedent-098169.html>

JD Edwards EnterpriseOne Tools API Reference (Document ID 705446.1) on My Oracle Support. <https://support.us.oracle.com/oip/faces/secure/km/DocumentDisplay.jspx?id=705446.1>

4 Programming Restrictions

Programming Restrictions

This chapter discusses programming restrictions for extensibility.

This list identifies programming features that you should follow when you extend the JD Edwards EnterpriseOne base code:

- If you are not a JD Edwards EnterpriseOne customer, do not use system codes 55-59 and all derivatives (such as 55S, 57RF, and so on). System codes 55x-59x are reserved for customers, and only JD Edwards EnterpriseOne customers should use these system codes. (Version 5 Update)
- Do not modify JD Edwards EnterpriseOne owned tables or their indexes.
- Do not modify JD Edwards EnterpriseOne owned business views.
- Do not modify JD Edwards EnterpriseOne owned data dictionary items (except the glossary).
- Do not delete data items from JD Edwards EnterpriseOne owned data structures (including but not limited to processing option templates, form data structures, report data structures, and business function data structures).
- Do not add a new value to an existing processing option in JD Edwards EnterpriseOne owned processing option templates.
- If you are not a JD Edwards EnterpriseOne customer, do not use user-reserved fields. User reserved fields are reserved for customers, and only JD Edwards EnterpriseOne customers should use these fields;
- Do not use future use fields. These are reserved for use by JD Edwards EnterpriseOne.
- Do not use existing fields for another purpose.
- Do not override data dictionary item settings in the application.
- Do not add a toolbar to an application if the application does not have an existing toolbar.
- Do not add hot keys to existing JD Edwards EnterpriseOne applications because there are a limited number of available letters.
- Do not use open source solutions within your JD Edwards EnterpriseOne solution.
- Dependency on third-party products needs approval from JD Edwards EnterpriseOne.
- If you copy Oracle owned objects, maintenance of the original JD Edwards EnterpriseOne code is owned by JD Edwards EnterpriseOne.

5 Foundational Design Patterns

Understanding Foundational Design Patterns

Foundational design patterns provide instructions for consistent implementation of standard features in the software. Consistency in the architecture ensures that you can test one occurrence and predictably know that other occurrences will work the same or similarly.

Implementing Currency

Within the JD Edwards EnterpriseOne system, you can set up multiple-currency from the Multi-Currency Setup menu on G1141. Within the General Accounting Constants application (P0000), you can set up multicurrency conversion with these three possible values:

- N—Do not use multicurrency
- Y—Activate multicurrency accounting and multipliers to convert currency.
- Z—Activate multicurrency accounting and divisors to convert currency.

You use the Designate Currency Codes application (P0013) to set up different currencies, and then use the Designate Company Currency application (P0010) to associate those currencies with different company codes. Next, you use the Designate Monetary Amount application (P0901) and the Currency Exchange Rate application (P0015A) to set up monetary amounts and exchange rates.

Currency Triggers

Table triggers are an easy way to implement currency conversion for a table. On the table trigger event, Currency Conversion is On, a call to the DecimalsTriggerGetbyCOCRCD (B1100007) business function can be made to automatically handle currency conversions for a table. The company key must be passed into the szCompany field and the From Currency Code must be passed into the szTransactionCurrencyCode field. The DecimalsTriggerGetbyCOCRCD (B1100007) business function takes a company and a transaction currency code and ensures that the currency is right for the transaction (potentially foreign) and base (domestic) currencies. This table trigger ensures that the values, whether read or write, that come out of the table have the right currency and decimal values.

Currency Conversion in the Business Logic

When copying currency fields to other currency fields, it is important to maintain the appropriate decimal position. When you copy currency fields in C business function code, you should use the API, jdeMathCopyCurrencyInfo, to maintain the correct number of decimals. When you copy currency fields in Event Rules (ER), whether an NER, application, or UBE, you should use the system function, Copy Currency Information to maintain the correct number of decimals. You can use the business function GetInvExchRate (B03B0174) to retrieve the system currency details (rates and methods). This business function calls GetExchangeRate (B0000033) and F00151GetConversionMethod (B0000172). Also CurrencyConvForAndDom (B0000027) converts amounts to other currencies.

Retrieving Address Book Numbers

In the JD Edwards EnterpriseOne Address Book System, address numbers are created with an address number, long address number, and tax ID. The address number is the primary key that is used throughout the system. However, the long address number and tax ID also can be used as keys throughout the system. In Address Book Constants (P01012), users set up the address book number to be used throughout the system as the primary number when accessing address book numbers. Address Book Constants provides separate fields for the address number, long address number, and tax ID. The field that is left blank is the primary address number. The other two fields are populated with a symbol.

For example, in Address Book Constants, this setup indicates that the Long Address Number is the primary address number in the system:

Symbol to Identify A/B Short Number *
Symbol to Identify Alternate Key
Symbol to Identify A/B Tax ID /

In this example, the Tax ID is designated as the primary address number:

Symbol to Identify A/B Short Number *
Symbol to Identify Alternate Key /
Symbol to Identify A/B Tax ID

Regardless of the setup in Address Book Constants, the short address book number (AN8) is always stored in the tables.

Address book numbers are used throughout the system to track all users, suppliers, and customers. There are a few common ways to retrieve address book numbers. The common data dictionary element to use is AddressNumber, which is data dictionary item AN8. This data dictionary item has an edit rule of IsColumnInAddressBook (B0100039) and a visual assist for the Address Book Master Search. When using this data dictionary item on forms, this edit rule and visual assist automatically apply. The edit rule automatically validates and makes sure that any value that is entered exists in the Address Book Master file. No additional code is needed for a validation. In addition, the visual assist is automatically available for users to search and select an address book number.

The IsColumnInAddressBook (B0100039) business function can be used to retrieve the address book description. You pass the Address Book field into the szBehaviorEditString field and retrieve the description from the szDescription001 field.

Throughout the system, the concept of Unknown Address Number (ALKY) is used when accessing and displaying address numbers. This concept allows applications and UBEs to display and access address numbers regardless of the setup in Address Book Constants.

By placing the ALKY field on forms, users can type in one of the three address numbers that they have set up as the main number in Address Book Constants. Then by calling the business function, ScrubAddressNumberNER (N0100061), the system determines which of the three address numbers to use. You can use this business function to move back and forth between the address book number, tax ID, and long address number. If the address number passed is not primary, then the address number must be prefixed with the symbol identifier specified in Address Book Constants. For example, in Address Book Constants, the following setup indicates that the long address number is the primary address number in the system:

Symbol to Identify A/B Short Number *
Symbol to Identify Alternate Key
Symbol to Identify A/B Tax ID /

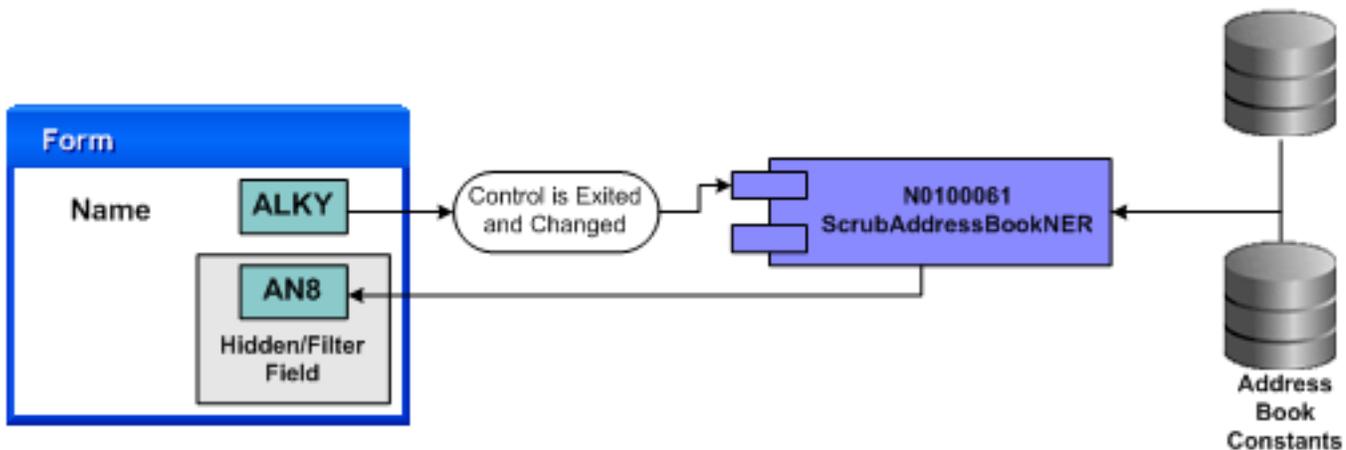
If you want to pass the short number, then the short address number must be prefixed with the symbol identifier *. Similarity, the tax ID must be prefixed with /.

The ScrubAddressNumberNER (N0100061) business function can be used in the following ways:

- Use an unknown address number.
- Use a short address number and return the primary format.
- Use a long address number and return the primary format.
- Use a tax ID and return the primary format.

The entered values and returns are discussed in the following topics.

The following diagram shows the flow for using the ScrubAddressNumberNER (N0100061) business function:



Using an Unknown Address Number

Pass an ALKY form control into the szAlternateAddressKey field and 1 in the szInputSelection field. The short address number can be returned in the mnAddressNumber field, the tax ID in the szTaxid field, description in the szNameAlpha field, and long address number in the szLongAddressKey field.

Using a Short Address Number and Return the Primary Format

Pass an AN8 value into the mnAddressNumber field and 2 in the szInputSelection field. The unknown address number can be returned in the szAlternateAddressKey field.

Using a Long Address Number and Return the Primary Format

Pass a long address number into the szLongAddressKey field and 3 in the szInputSelection field. The unknown address number can be returned in the szAlternateAddressKey field.

Using a Tax ID and Return the Primary Format

Pass a long address number into the szTaxid field and 4 in the szInputSelection field. The unknown address number can be returned in the szAlternateAddressKey field.

Using Unknown Item Format

Throughout the system, the concept of unknown item format (UITM) is used for accessing inventory item numbers. This concept enables the system to be configured to access item numbers using one of three item numbers that are defined for identification of the item. This configuration is defined by branch/plant when setting up the Branch/Plant constants. Because every branch/plant could potentially be set up to use a different item number as its primary item number, the concept of UITM enables applications to display and access items regardless of the branch/plant setup.

In the JD Edwards EnterpriseOne Inventory System, items are created with a short item number, second item number, and third item number. The short item number is the primary key and is automatically assigned by the system. The second and third item numbers are text and are setup by users. In the Branch/Plant Constants application (P41001), users set up which item number to use as the primary number when accessing item numbers in that branch/plant throughout the system. Branch/Plant Constants has three separate fields for each of the item numbers (short, second, and third). The field that is left blank is the primary item number. The other two fields are populated with a symbol.

For example, in Branch/Plant Constants, the following setup indicates that the second item number is the primary item number for that particular branch/plant:

```
Short Item Number /
Second Item Number
Third Item Number *
```

In this example, the third item number is the primary item number:

```
Short Item Number /
Second Item Number *
Third Item Number
```

Regardless of the setup in Branch/Plant Constants, the short item number (ITM) is always the primary key within JD Edwards EnterpriseOne tables. When the item number is displayed on a form, you should be able to enter the item number in either short, second, or third item format to validate and retrieve an existing item master entry. If the item number passed is not primary, then the item number must be prefixed with the symbol identifier specified in Branch/Plant Constants. For example, in Branch/Plant Constants, the following setup indicates that the second item number is the primary item number for that particular branch/plant:

```
Short Item Number /
Second Item Number
Third Item Number *
```

If you want to pass the short item number, then the short item number must be prefixed with the symbol identifier /. Similarly, the third item number must be prefixed with an asterisk (*).

The unknown item format can be used in these ways:

- Use UITM for form data selection.
- Use UITM to display the primary item

Using UITM for Form Data Selection

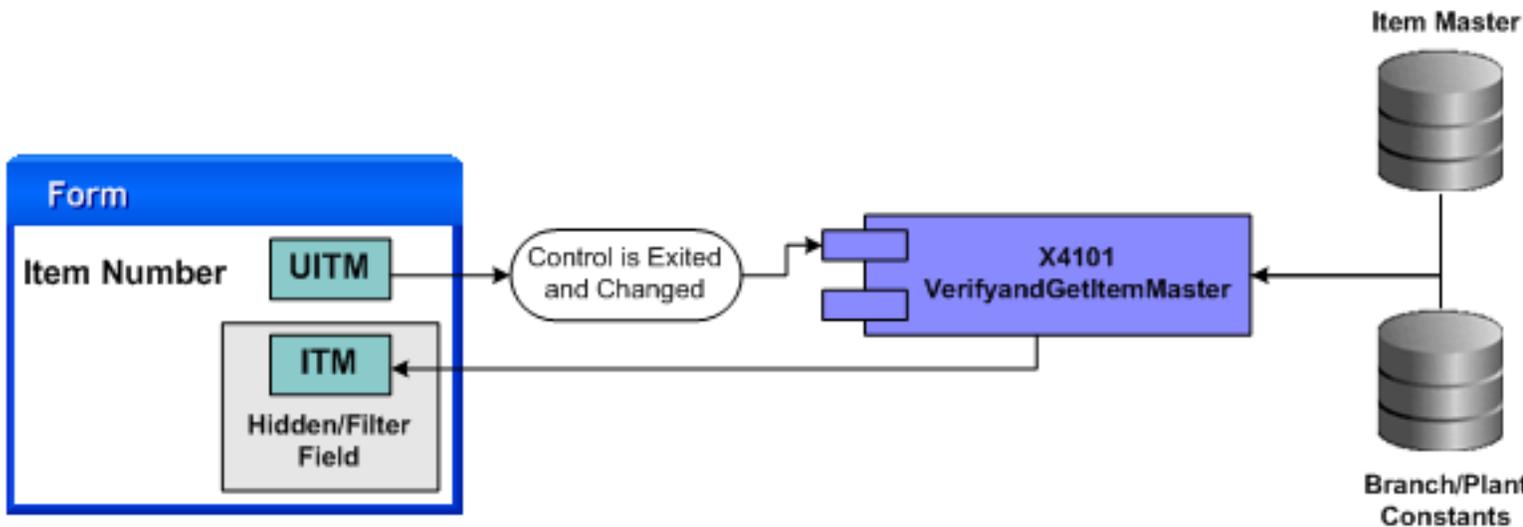
This scenario is typically used for the Bill of Material Revisions (P3002) application. In a user entered field, enter a primary item number. The system determines what item number you are using before validating the item number.

Use these steps to implement the UITM for item numbers for this scenario:

- Enter the UITM data dictionary field on the form.
- Within the Control is Exited and Changed event rule, call the VerifyAndGetItemMaster (X4101) function.
- Pass in FC as the value for the primary item number (value contained in the UITM field that a user entered), branch/plant, and a <Blank> value in the symbol identifier field.
- Retrieve the short, second, and third item numbers as variables or other form fields.

When you use the item number for data selection, return the short item number to a form field that is defined as a filter field.

The following diagram illustrates this scenario:



Using UITM to Display the Primary Item

This scenario is typically used for the Bill of Material Revisions (P3002) application. In a display field, a table might have stored the short item number, but you want to see only the item number that is relevant for that branch/plant.

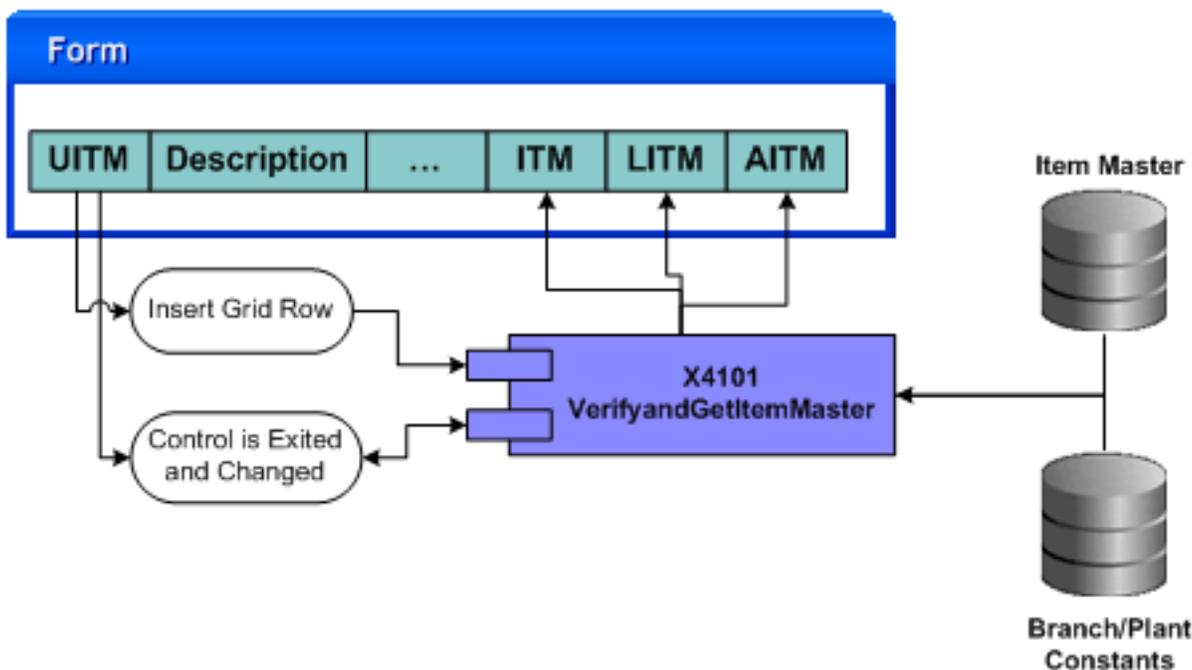
Use these steps to implement the UITM for item numbers for this scenario:

- Enter the UITM data dictionary field on the form/grid.

- When the form/grid retrieves the existing item number, if the form does not have all three item numbers, do one of the following:
 - If only the short item number was stored in the table used by the application, add a call to the GetItemMasterByShortItem (X4101) business function. Pass in the short item number and retrieve the second and third item numbers.
 - If the form has only the second item number, but not the short and third item numbers, call the GetItemMasterBy2ndItem(X4101) business function. Pass in the second item number and retrieve the short and third item number.
 - If the form has only the third item number, but not the short and second item numbers, call the GetItemMasterBy3rdItem(X4101) business function. Pass in the third item number and retrieve the short and second item number.

After all three item numbers are retrieved, add a call to the ReturnPrimaryItemNumber (X4101) business function. Pass in the short item number, second item number, third item number, and branch/plant. Retrieve the primary item number and display the value in the UITM field.

This diagram illustrates this scenario:



Transaction Processing

Transaction Processing ensures the highest data integrity and quality within the system. It provides a way to rollback related changes to the database when an error occurs. Related changes are processed within a transaction boundary. Changes (also referred to as updates) include inserts, updates, and deletes to records within the database.

A transaction is a logical unit of work (comprised of one or more SQL statements) performed on the database to achieve a common task and maintain data consistency. Transaction statements are closely related and perform interdependent

actions. Each statement performs part of the task but all are required for the completion of the task. Everything within the defined transaction boundary is rolled back (or undone) if an error updating the database occurs.

A transaction must be performed to completion or not performed at all. The updates for a transaction should not be visible to other transactions running concurrently until it commits. A commit signifies the successful end of a transaction and updates made within the transaction boundary are written permanently to the database to bring the database to a consistent state. A rollback signifies the unsuccessful end of a transaction, and database operations performed within the transaction boundary are undone or rolled back so as to return the database to its previous consistent state before the transaction began.

Orphan records occur when a transaction is processed without using transaction processing, and the processing of the transaction terminates somewhere in the middle of the related updates. Data updated by dependent operations can be committed in an all or nothing way so that orphan records are not left behind by a failure. Transaction processing guards against system failure and maintains data integrity and data consistency.

JD Edwards EnterpriseOne supports two types of transaction processing:

- Auto Commit-database updates are written permanently to the database (committed) as they run.
- Manual Commit-database updates are written permanently to the database only when either a commit or rollback is performed.

Within JD Edwards EnterpriseOne, transaction scope for interactive applications is defined in Form Design Aid (FDA) and is controlled by the runtime engine. The transaction scope for UBEs is defined in Report Design Aid (RDA) and is controlled by the batch engine.

The transaction boundary can include:

- Forms (application specific).
- Form interconnects (application specific).
- Subforms (application specific).
- Table I/O.
- Calls to business functions (both C and NER functions).
- Boundaries contained within business functions (C function only).

Business functions can be included in the scope of a transaction from either FDA or RDA, or they can define their own scope.

Extensive research should be done prior to implementing transaction processing. This list identifies some special considerations for transaction processing:

- If logic is extended in an application that has transaction processing enabled, you must determine whether the extended logic is within or outside of the transaction processing boundary.

For example, if logic is added to the OK button, you must determine where to add the logic. If the table I/O within the logic needs to be included within the transaction processing boundary, the logic should be extended on the OK Button-Post Button Clicked event. If the logic uses the data that is updated within the transaction processing boundary, that logic should be placed on the OK Button-Post Commit event.

- When mixed, manual commits and auto commits form different and distinct transaction boundaries, not one boundary.

Therefore, any rollback with a mixture of manual and auto commits in place can cause unexpected results.

- Records processed for database that update more than once within a transaction boundary can result in database deadlocks, forced record lock releases (dirty reads), and kernel failures.
Using cache to store information about a record until the final update of the record is performed within a boundary is a common method of resolving this issue. Other solutions include managing the table outside of the transaction processing boundary.
- Not all tables should be included in a transaction boundary.
Tables that are used by various users at any time may need to be outside of the transaction boundary. If certain common tables are included inside the transaction boundary, many other processes are locked because another process locks a table.
For example, various financial, inventory, and next number tables should not be included in the transaction boundary. If the Next Number table (F0002) is included inside the transaction boundary then only one user at a time is able to retrieve a next number. This is not practical and can cause system-wide issues.
- Next numbers should be managed outside the transaction processing boundary.
Another example is the Inventory Item Location file (F41021). This table records all transactions against an item and is accessed by numerous processes. Other applications are likely to update the same record frequently. Locking the table could cause a huge impact system wide.
Implementing transaction processing for F41021 records requires additional logic for managing F41021 transactions outside of the transaction processing boundary.

You can enable transaction processing in these objects:

- Applications
- Business functions
- UBEs

Enabling Transaction Processing in an Application

Transaction processing can be performed on JD Edwards EnterpriseOne application forms.

Form Transaction Scope

The default transaction processing boundary includes all adds, updates, and deletes and are available on Header Detail, Headerless Detail, Power Edit, and Fix Inspect forms.

This list identifies events that can be called within the transaction processing boundary:

Start of transaction processing boundary

- Add Record to DB-Before
- Add Record to DB-After
- Update Record to DB-Before
- Update Record to DB-After
- Add Grid Record to DB-Before
- Add Grid Record to DB-After
- All Grid Records Added to DB
- Update Grid Record to DB-Before

- Update Grid Record to DB-After
- All Grid Records Updated to DB
- Delete Grid Record from DB-Before
- Delete Grid Record from DB-After
- All Grid Records Deleted from DB
- OK Post Button Clicked
- OK Post Button Clicked-Asynchronous

End of transaction processing boundary

OK Post Button Clicked-Asynchronous

The use of the OK Post Button Clicked-Asynchronous event has the following limitations:

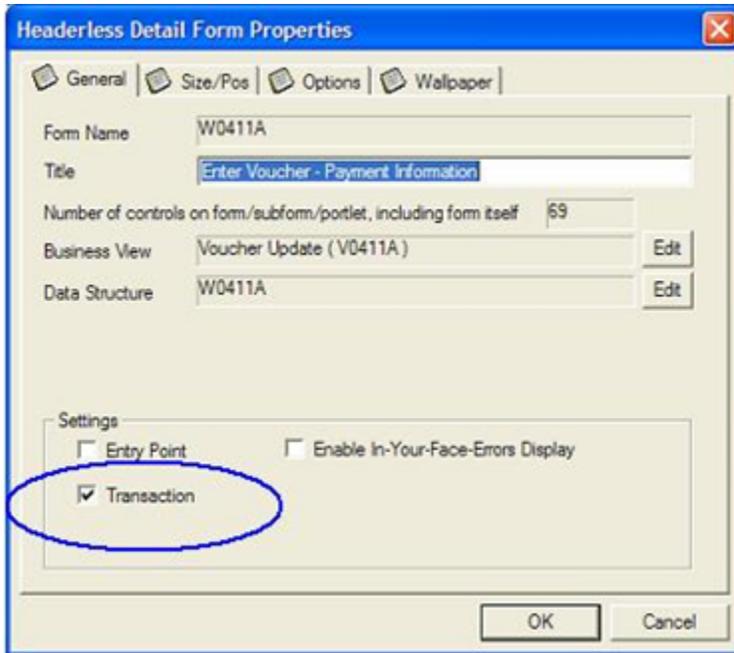
- No form interconnect.
- No grid access (GC, GB).
- No updating values to form and grid controls, form and grid variables, and form interconnect.
- Any system functions that perform on form control, grid controls, or on the form itself (Media Object-Display) are not allowed.
- Updating values for the event variables are allowed.
- System variable: TP_FORM_STATUS.

Post Commit

The use of the Post Commit event has the following limitations:

- No form interconnect.
- No grid access (GC, GB).
- No updating values to form and grid controls, form and grid variables, and form interconnect.
- Any system functions that perform on form control, grid controls, or on the form itself (Media Object-Display) are not allowed.
- Updating value for the event variables are allowed.
- System Variable: TP_COMMIT_STATUS.

In FDA, the form transaction scope is defined on the Headerless Detail Form Properties, as illustrated in this example:



Subform Transaction Scope

Transaction processing can be performed on JD Edwards EnterpriseOne subforms. The default transaction processing boundary includes all adds, updates, and deletes and are available on subforms only.

This list identifies events that can be called within the transaction processing boundary:

- OK Button Clicked
- OK Post Button Clicked
- Add Record to DB-Before
- Add Record to DB-After
- Update Record to DB-Before
- Update Record to DB-After
- Add Grid Record to DB-Before
- Add Grid Record to DB-After
- All Grid Records Added to DB
- Update Grid Record to DB-Before
- Update Grid Record to DB-After
- All Grid Records Updated to DB
- Delete Grid Record from DB-Before
- Delete Grid Record from DB-After
- All Grid Records Deleted from DB

In FDA, the subform transaction scope is defined on the Edit Subform Properties form, as illustrated in this example:

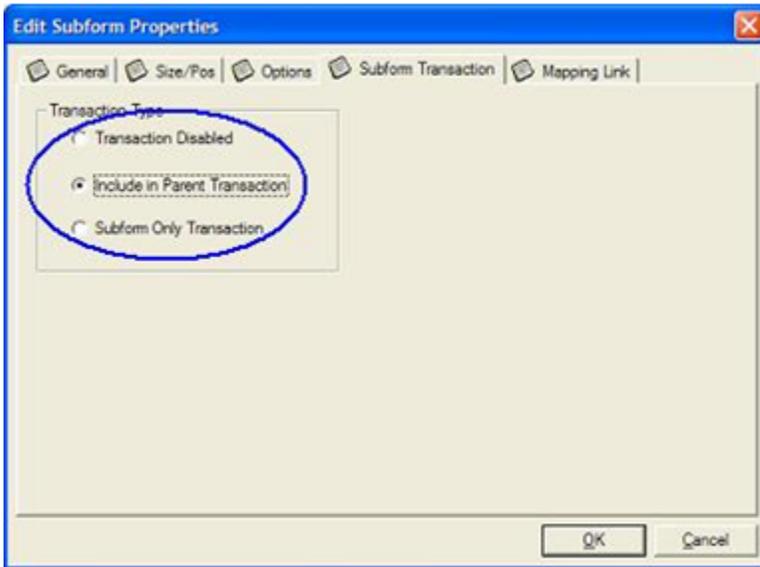
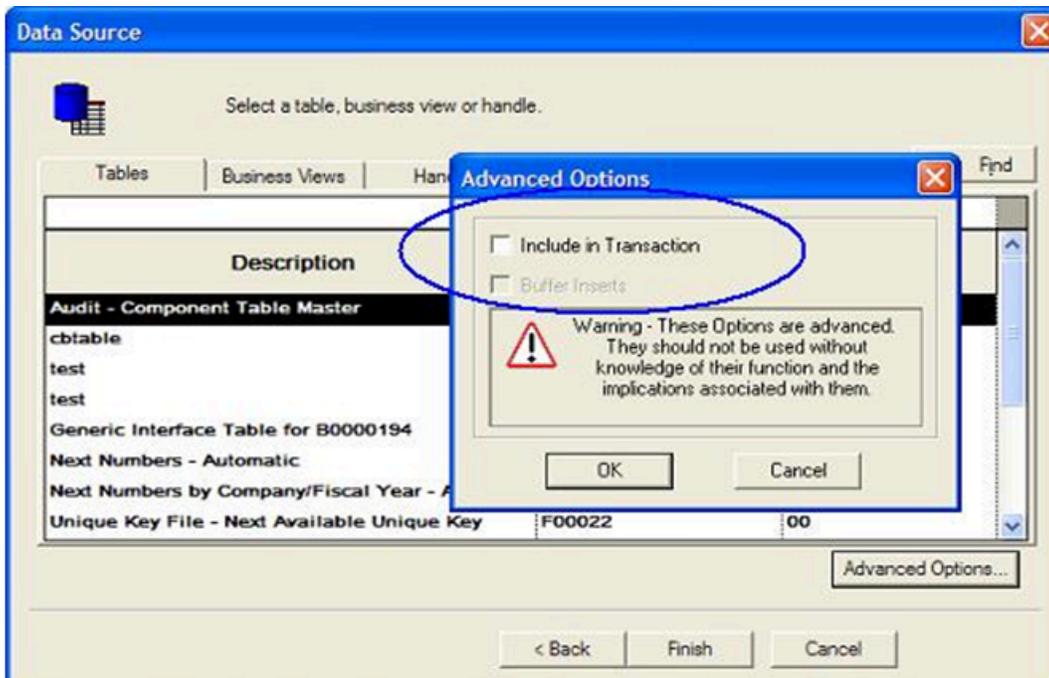


Table Input/Output Scope Extension

Transaction processing can be performed on JD Edwards EnterpriseOne tables. Expand the transaction boundary to include table I/O operations. The Include in Transaction check box is available only on the Open event rule, but this setting will affect all subsequent access to the table that occurs within the defined transaction boundary.

In FDA, the extended transaction scope for table I/O is defined on the Data Source form. The following illustration shows how to access the transaction processing setting for tables from FDA:



Business Function Scope Extension

Transaction processing can be performed on JD Edwards EnterpriseOne business functions. Expand the transaction boundary to include the database operations in a business function. A business function marked asynchronous can be selected as Include in Transaction.

In FDA, the extended transaction scope for a business function marked asynchronous is defined on the Business Functions form, as illustrated in this example:

Business Functions

Function Name : F0411FSEndDoc
Function Description : F0411 End Document
Source Module : D0400047 Data Structure : D0400047D

Transaction Processing
 Include in Transaction

Run Process
 Asynchronously

Available Objects

- FC Job Number - wf
- FC Previous Document
- FC Exchange Rate Edit
- FC Taxable Amount
- FC Tax
- FC Discount Available
- FC Gross Amount
- FC Batch Number
- FC Document Number**
- FC Address Number

Data Structure

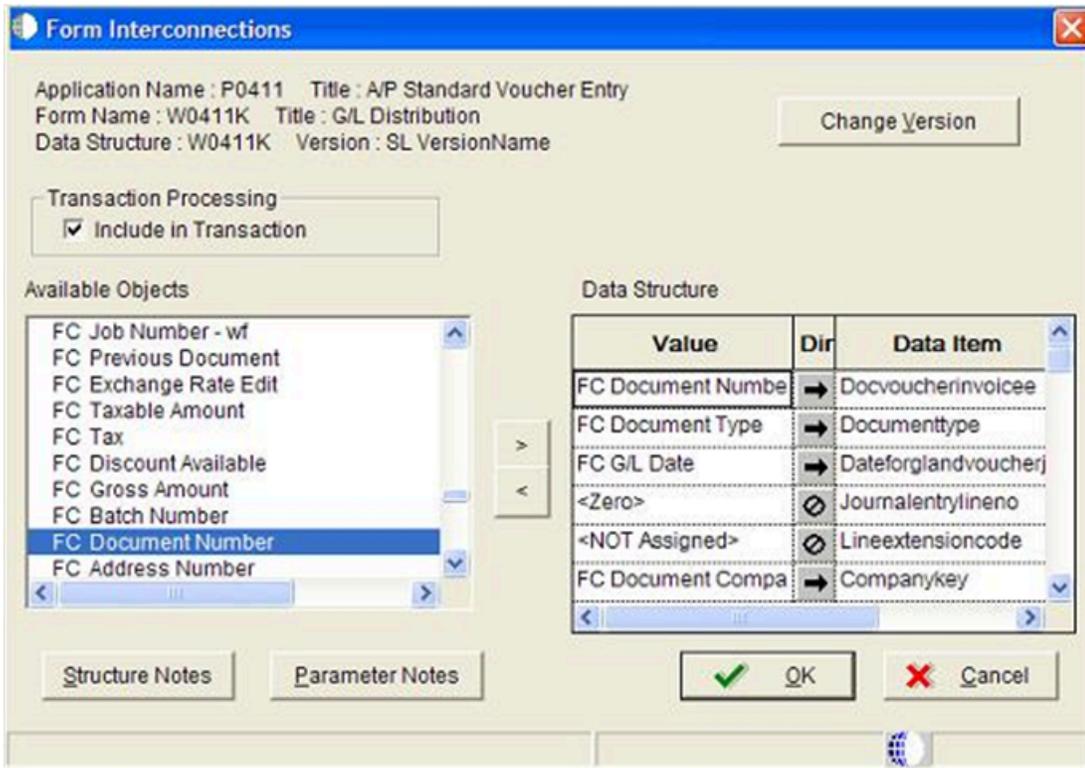
Value	Dir	Data Item
FC Document Number	→	mnDocNumberAssigne
VA frm_ErrorFlag	→	cErrorFlag
VA frm_CurrentAmount	→	mnDocumentAmount
VA frm_PreviousAmou	→	mnPrevDocumentAmo
VA frm_Curconv	→	cCurrencyFlag
VA frm_BatchStatus	→	cPrevBatchStatus

Business Function Notes Structure Notes Parameter Notes OK Cancel

Form Interconnect Scope Extension

Transaction processing can be performed on JD Edwards EnterpriseOne form interconnections. Expand the transaction boundary to include other form adds, updates, and deletes. Transaction processing can be performed only for form interconnects that happen from the events included in the transaction boundary.

In FDA, the extended transaction scope for form interconnections is defined on the Form Interconnections form, as illustrated in this example:

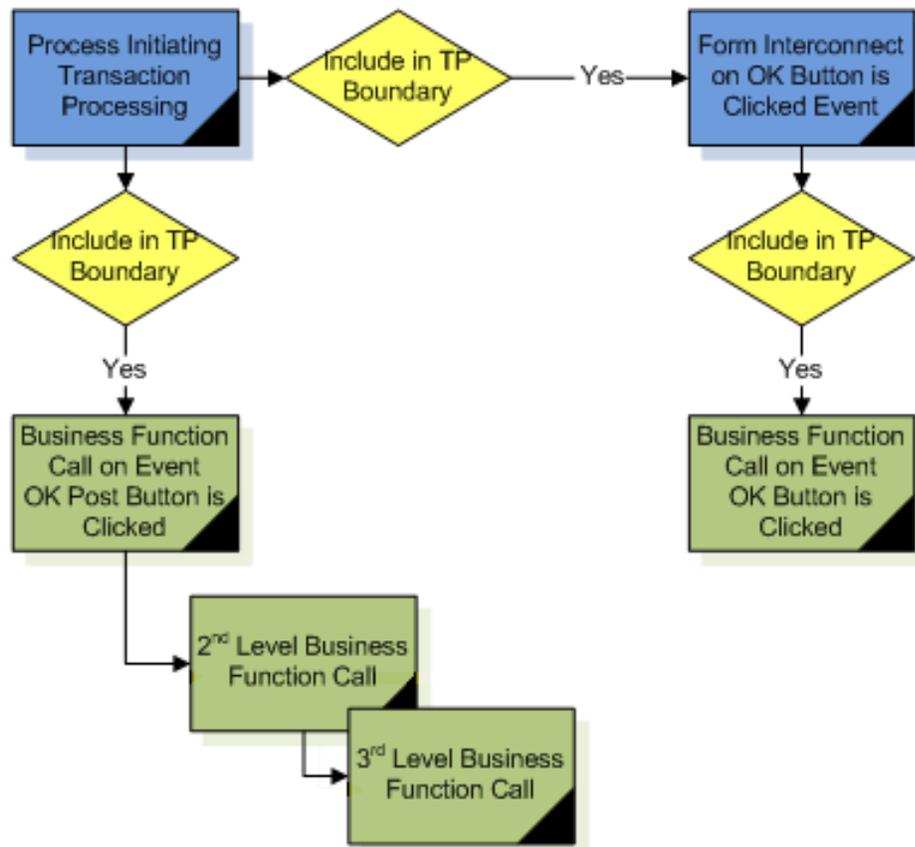


Example of Transaction Scope in an Application

Transactions processing can be started in one form and extended to more forms through form interconnects. Business function calls also can be included in a transaction boundary. If the application calls several business functions, and it needs to be part of the transaction boundary, make sure to include the business functions in the transaction boundary.

The following diagram illustrates setting up transaction processing within an application:

Transaction Processing Boundary



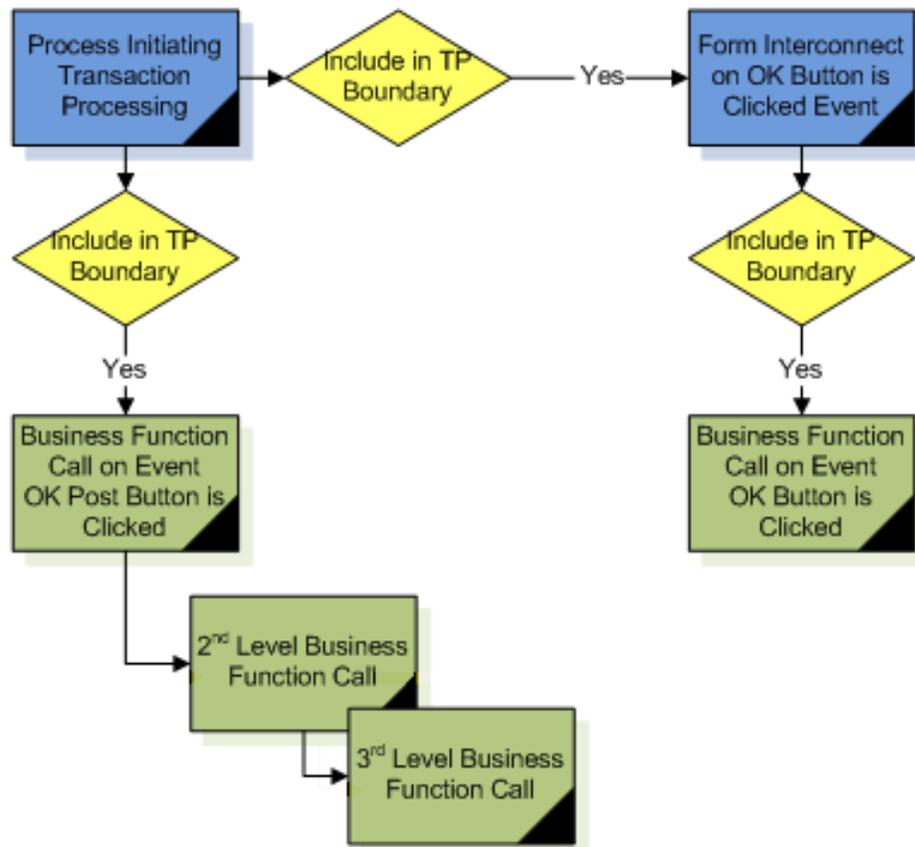
The following setup occurs within the FDA specifications:

- Transaction processing is checked as active on the form initiating the transaction processing boundary.
- The form interconnect is marked as Include in Transaction for the form called on event **OK Button is Clicked**.
- Business function calls are marked as **Include in Transaction** for both forms.

Transactions processing can be started in one form and extended to more forms through form interconnects. Business function calls also can be included in a transaction boundary. If the application calls several business functions, and it needs to be part of the transaction boundary, make sure to include the business functions in the transaction boundary.

The following diagram illustrates setting up transaction processing within an application:

Transaction Processing Boundary



The following setup occurs within the FDA specifications:

- Transaction processing is checked as active on the form initiating the transaction processing boundary.
- The form interconnect is marked as Include in Transaction for the form called on event **OK Button is Clicked**.
- Business function calls are marked as **Include in Transaction** for both forms.

Enabling Transaction Processing in a Business Function

Enabling transaction processing in a business function is not a recommended procedure but can be done where applicable. If you must enable transaction processing in a business function, follow these guidelines:

- At the beginning of the business function, add a call to the JDB_InitUser API with the last parameter set to JDEDB_COMMIT_MANUAL. For example:

```
JDBReturn = JDB_InitUser((HENV) lpBhvrCom->hEnv, &hUser, (JCHAR *)NULL, JDEDB_COMMIT_MANUAL));
```

- Continue to do the normal JDB_InsertTable, JDB_UpdateTable, and JDB_DeleteTable API calls for the table input/output.
- At the end of the transaction boundary, place a call to the JDB_CommitUser API; for example:

```
JDB_CommitUser(hUser);
```

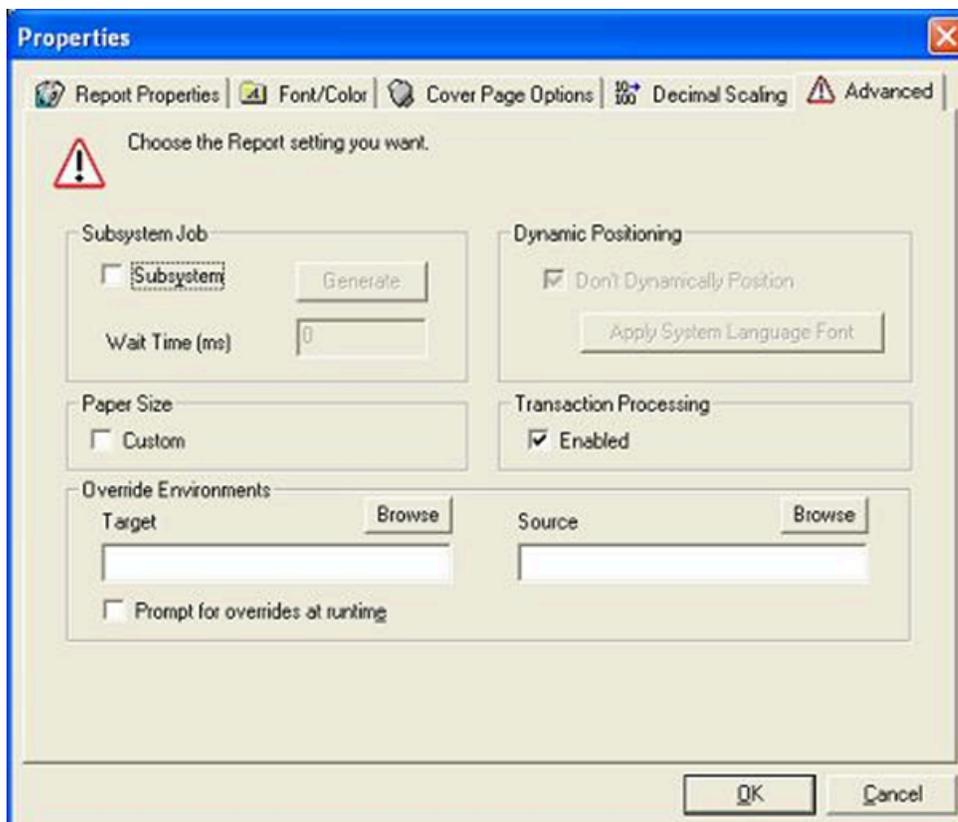
Additional information about the different JD Edwards EnterpriseOne APIs and their usage can be found in the JD Edwards EnterpriseOne Tools API Reference document.

See *JD Edwards EnterpriseOne Tools API Reference* (Document ID 705446.1) on My Oracle Support. <https://support.us.oracle.com/oip/faces/secure/km/DocumentDisplay.jspx?id=705446.1>

Enabling Transaction Processing in a UBE

The scope for transaction processing for a UBE is defined in the report properties. Do not let the transaction grow larger than one full iteration of the driver section of the UBE as this can lead to record locking over multiple records.

In Report Design Aid (RDA), enabling transaction processing in a UBE is defined on the Properties form, as illustrated in this example:



Prior to JD Edwards EnterpriseOne Tools Release 8.98 Update 3, the transaction is defined by these system functions:

- Begin Transaction

- Commit
- Rollback
- Open Table (used to include table I/O within the UBE in the transaction boundary)
- Close Table (used to include table I/O within the UBE in the transaction boundary)

Begin Transaction

Identify the beginning of the table updates that should be part of the transaction. Create a report-level Math Numeric variable for the transaction ID. Because only one transaction boundary can be supported at a time, this value should be set to 1, as illustrated in this example:

```
VA rpt_mnTransactionID_MATH01 = 1
```

```
Begin Transaction(VA rpt_mnTransactionID_MATH01)
```

Commit Transaction

After all of the required table updates are done, call this function:

```
Commit Transaction(VA rpt_mnTransactionID_MATH01)
```

Rollback Transaction

Identify all of the exit points in the UBE that are possible after the Begin Transaction but without hitting the Commit Transaction. This includes updates performed by called business functions that are included in the transaction processing boundary. One exit point should exist for each potential failure to update the database. At each of these exit points, call this function:

```
Rollback Transaction(VA rpt_mnTransactionID_MATH01)
```

Open Table

Tables updated by table I/O in the UBE are included in the transaction processing boundary only if they are implicitly opened with the setting to include them in the transaction processing boundary. Use the following table I/O operator to open the table:

```
Open Fnnnnn.
```

When selecting the table, select the form button for Advanced Options, and then select the Include in Transaction check box.

Close Table

Tables opened for update to be included in the transaction processing boundary must also be closed after updating the table and prior to performing the commit or rollback. Use the following table I/O operator to close the table:

```
Close Fnnnnn.
```

Changes to Transaction Processing in a UBE

JD Edwards EnterpriseOne Tools Release 8.98 Update 3 introduced changes to the UBE transaction processing capabilities. The activation of transaction processing within a UBE is the same, as well as the use of the Begin Transaction; however, the use of the Commit and Rollback system functions have changed significantly.

Commit Transaction

If a database update error occurs within the transaction processing boundary, the Commit Transaction system function results in a rollback of the transaction. This resolves the need to capture every database update error using return variable processing. This includes database update errors that occur several levels deep within the functions that are called from the UBE.

Rollback Transactions

The Rollback Transaction system function can be used to force the rollback of database update errors; however, it is not required for this purpose. The Rollback Transaction is useful to rollback transactions for errors that are not database update related; such as a table fetch fails to return a required record, or a returned table value indicates the transaction should not complete successfully.

The new system variable SV TP_Commit_Status records commitment and rollback activities. The different values assigned to the variable provide status information about the transaction processing boundary. The values and their meanings are listed here:

- CO TP_ACTION_SUCCESS

If SV TP_Commit_Status is equal to CO TP_ACTION_SUCCESS, the last transaction action succeeded. The last transaction action can be either Commit Transaction or Rollback Transaction. If the last transaction is Commit Transaction, the commit succeeded and all database operations are committed. If the last transaction is Rollback Transaction, the rollback operation succeeded.

- CO TP_ACTION_FAIL

If SV TP_Commit_Status is equal to CO TP_ACTION_FAIL, the last transaction action failed. The last transaction action can be either Commit Transaction or Rollback Transaction. If the last transaction is Commit Transaction, the commit failed and no database operation was committed. If the last transaction is Rollback Transaction, the rollback operation failed. No database records were committed.

- CO TP_IN_TRANSACTION

If SV TP_Commit_Status is equal to CO TP_IN_TRANSACTION, the transaction has started but no Commit Transaction or Rollback Transaction system function has been called yet.

- CO TP_NO_TRANSACTION

If SV TP_Commit_Status is equal to CO TP_NO_TRANSACTION, no transaction has started or completed by the transaction processing system functions.

Note: UBE print cycles occur even if (or when) a transaction is rolled back due to database errors encountered during transaction processing boundaries. An example of this is a columnar report section. The line or lines on the report related to the failed transaction will print if the report default is to print one or more lines for each record processed. If the desired result is to not print lines associated with rolled back transactions, logic must be added to the UBE to avoid the printing of transactions that did not successfully process (or to track failed transactions for later reporting). Similar considerations also must be taken into account for any report totals accumulation processing where totals are accumulated based on level break logic or event rule summarization.

Implementing Record Reservation

Record reservation ensures the highest data integrity and quality within the system. It provides a way to prevent the records associated with a business entity from being updated by two users or two separate applications at the same time.

Record reservation should be implemented in all transaction tables where multiple users or applications could update the same record at the same time. For example, record reservation is implemented in Sales Order and Work Order Entry because multiple users might attempt to update the same sales order or work order at the same time.

Record reservation does not need to be implemented in certain setup applications, like Address Book, because this type of information is usually a one-time entry. Also, if an existing JD Edwards EnterpriseOne table is already using record reservation and a new integration point is being added, record reservation must be implemented following the current standards for that table. Two applications that use record reservation are:

- Work Order Entry (P48013)—reserves based on both a Work Order and Engineer to Order (ETO) Project ID)
- Sales Order Entry (P42101)

Use these steps to implement record reservation:

- Update UDC 00|RR to register the application where record reservation is being implemented.
- Enter the identifier of the new application (PXXXZZZZ) in the Codes field, enter the name of the application in the Description field, and then set the Special Code field to 1.
- In the entry event of the application (for example, Dialog is Initialized), call the business function ValidateApplicationId (N0000604).

This function validates that the application is registered in UDC 00|RR. When calling the ValidateApplicationId business function, pass the identifier of your new application (PXXXZZZZ) into the szProgramId field. If the application is registered, the function returns 0. If the application is not registered, the function returns 1. The cValidateApplication_ERR1 field contains the return value of 0 or 1.

- If the ValidateApplicationId returns 0, call F00095ReserveObject (N0000602) to commence the reservation.

This function reserves the record by the key. When calling this function, pass the entity identifier (usually the master table name) into the szNameObject field (for example, FXXXZZZZ). Pass the user ID into the szUserId field, and pass the key value into the szGenericKey field. The value passed into the szGenericKey field is usually the key value of the record being reserved. If there is more than one key for the record, concatenate the values and then pass them into the szGenericKey field.

- The value returned in the szErrorMessageId field from the F00095ReserveObject function can be used to determine if the record has already been reserved by another user or if the record was reserved correctly and the user can now edit it.
 - If a non-blank value is returned in the szErrorMessageId field, then another user has reserved the record and the current user cannot edit it. In order to display this information in the application, the function SetReservationError (B0000603) can be called. Pass the following fields from the F00095ReserveObject business function data structure into the SetReservationError function data structure:
 - szReserversNameAlpha to szReserversNameAlpha
 - szGenericKey to szGenericKey
 - szReserversApplication to szApplication

- If a value of blank is returned in the szErrorMessageID field, the record was reserved properly and the user can edit the information.
- At the end event of the application (for example, End Dialog), you must call the business function F00095RemoveReservation (N0000602) to release the reserved record that was written to the F00095 table. When calling this function, pass in the same values that were passed in when the F00095ReserveObject function was called to initially reserve the record.

Note: If you are implementing record reservation for a table that JD Edwards EnterpriseOne has not already implemented, contact Oracle JD Edwards EnterpriseOne for the proper keys to use for that specific table.

Using Next Numbers

The term next numbers is used to refer to the process of obtaining a unique document number, a unique record ID, and a unique job number.

You use Next Numbers to:

- Assign a unique document number.
- Assign a unique record ID.
- Obtain an internal next number for unique caches.

Assigning a Unique Document Number

You use next numbers to assign a unique document number. Next numbers ensure that business forms, such as sales orders and invoices, are assigned a unique number. Retrieval of the next number for assigning a unique document number should be placed outside of the transaction processing boundary. Placing it within the transaction processing boundary can cause a deadlock situation to occur. Two applications that use next numbers to assign a unique document ID are Work Order Entry (P48013) and Sales Order Entry (P42101).

In the Next Numbers application (P0002), each system code can have up to 10 possible next numbers.

This example illustrates how to assign a unique document number:

A sales order number is setup in the Next Number application in slot 1 for system 42 and the Vessel ID is setup in the Next Number application in slot 6 for system 31B. You can add a record in the Next Number application (P0002) for your new system code (XXX). Then you pick a slot (1 through 10) for the next number.

Once the slot for the next number is decided, you can use the function X0010GetNextNumber (X0010) within the code to retrieve the next number available for the field. For example, to add a new sales, call the X0010GetNextNumber function and pass 42 to the szSystemCode field and 1 into the mnNextNumberingIndexNo field. This function returns the next sales order number in the mnNextNumber001 field and increments the value and stores the value in the F0002 table. If you need to retrieve a new Vessel ID, call the X0010GetNextNumber passing 31B to the szSystemCode field and 6 into the mnNextNumberingIndexNo field.

Note: After you have chosen a use for a next number slot, it should never be changed. Because only 10 slots are available, the usage of this table should be kept to a minimum.

Assigning a Next Number as a Unique Record ID

You might use a unique number in a single table for entries that are only unique to that particular table and will not have multiple entry points. An example of this is the Sales Order Detail Commission Table (F42160) or any other table that needs a line number or unique ID.

This example illustrates assigning a next number as a unique record ID:

The Sales Order Detail Commission table has a commission line number in its key. You call the `GetNextUniqueKeyID` (X00022) function and pass the table name F42160 in the `szObjectName` field. You retrieve the next available unique number for that table in the `mnUniqueKeyID` field. If the table has eight lines, this function returns the number 9 in the `mnUniqueKeyID` field and that is the next number to use.

If there is a new table, `GetNextUniqueKeyID` determines that no records exist and returns a value of 1.

Obtaining an Internal Next Number for Unique Caches

You use a job number to keep cache records unique across a session or a user. A session is an instance of JD Edwards EnterpriseOne that has been opened by a user. A user can open multiple sessions of JD Edwards EnterpriseOne if the user is doing multiple tasks in the same or different applications. A user is the specific person who has opened a particular session of JD Edwards EnterpriseOne. For example, in a company using JD Edwards EnterpriseOne, five different users are signed into two different sessions of JD Edwards EnterpriseOne, each for a total of 10 instances of JD Edwards EnterpriseOne active at one time. A job number helps to identify and make unique the different cached data that needs to be accessed by different programs. This next number is not a next number that needs to be retained in the system after a particular process is finished.

This example illustrates obtaining an internal next number for unique caches:

You use the function `GetInternalNextNumber` (B0000564) to retrieve a next number for this particular purpose. You call this function without passing any key values. Then you use the value passed back in the `mnJobnumberA` field as the next available job number.

Adding a Date and Time Field

When adding date fields to an object, use these guidelines to determine whether you should use a date type of `JDEUTIME` or `JDEDate` to define the date field:

- Use `JDEUTIME` when both date and time information is required.
- Use `JDEUTIME` when date and time information always needs to be presented in a time zone other than the server's time zone.
- Use `JDEDate` if the time is not needed.
- Use `JDEDate` if the date information will always be used in the same time zone as the server.

When you add a field using the date type of `JDEUTIME` to an application, you can set the data dictionary item to display the date only, date and time, or date, time, and offset.

Searching Tree Controls

An essential feature of JD Edwards EnterpriseOne is allowing users to search for a value in an application that has a tree structure. For example, the Bill of Material (BOM) application has a tree structure.

Understanding Wildcard Character Searches

You can use a wildcard character to search for a value in EnterpriseOne applications. The JD Edwards EnterpriseOne software understands an asterisk (*) as the wildcard character. However, the Oracle database understands percent (%) as the wildcard character. Use these steps to convert a wildcard character to search for a value in EnterpriseOne:

1. Enter a search string that uses an asterisk (*) as a wildcard character.
2. Run the `ReplaceCharacterInString (B0400520)` API to convert the search string with the asterisk to a string with a percent (%).
3. The output search string is formatted with a %.

You can use this formatted search string to search for a value in an application with a tree structure.

Searching for a Value in a Tree Structure

You can create a search string that has the appropriate values for searching for a value in a tree structure, or you can use the formatted search string from the previous task to search for a value in a tree structure. Use these steps to search for a value in a tree structure:

1. Run a business function (like `exitCommodityCodeCache (B4004160)`) to clear all of the values in the tree cache.
This business function is coded to terminate the existing cache, if one exists.
2. Create a variable to store the path of the selected node, and clear the variable.
3. Run the `Delete Tree Node` API to delete all of the current nodes in the currently displayed tree structure.
4. Create a variable to store the root value of the tree, clear it, and assign the text `SearchResults` to it.
5. Run the `GetInternalNextNumber (B0000564)` business function to get a unique job number.
6. Create a variable to store the tree cache name (`B4004000`) and concatenate the job number to the variable.
7. Create a business function (like `FindCommodityCodeStructure (B43E4100)`) to perform the search in the tree structure or cache.

The business function performs these activities:

- Forms a query to fetch the appropriate value from the table.
- Fetches the values from the table that match the value provided in the search string.
- Adds the matching results from the table to the tree cache that was created.

Loading the Tree Structure with the Search Results

After the search results are added to the tree cache, use these steps to load the tree structure with the search results:

1. Run the `Bulk Tree Node` API to load the tree from the tree cache.

2. Run the Set Node Information API to allow the system to display the result text of a node and its associated node value.
3. Run the Set Node Text API to set the texts.

Displaying the Path

Use these steps to display the path in the Edit Control when a node is selected.

In the tree node selected event:

1. Clear the path variable.
2. Run the GetNodeInformation API.

This API returns the display text of a node and its associated node value.
3. Set the returned tree node level value in the FC control provided, and set the returned tree node level value in a variable to use as input to processCommodityCodeStructureCache.
4. Run GetNodeLevel to set the current node level
5. Run the processCommodityCodeStructureCache (B4004000) business function to retrieve the tree level.
6. Set the FC control with the path selected using the processCommodityCodeStructureCache.

6 Application Extensibility Design Patterns

Determining an Active System Code

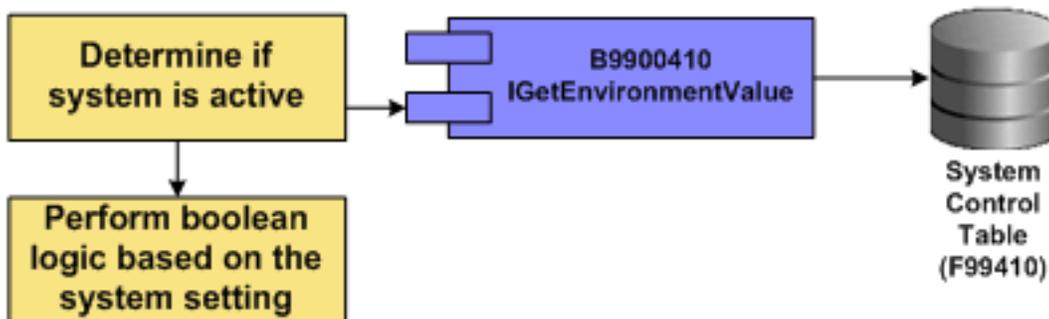
The active system code pattern determines whether a system code is active or inactive. This pattern controls logic flow and audits your enabled systems. For controlling logic flow, the value of the system setting is checked before allowing certain actions or tasks to be performed by the user. The system setting also controls logic to either interface or bypass the interface to the system. For auditing, the system setting can determine which systems you have enabled.

Not all active system codes need to be defined in the System Control table (F99410). You use the System Control application (P99410) to define the active system codes that have the need and logic to use F99410. The use of this system setting is defined in the system user guides that use P99410.

You can determine whether a system code is active by doing the following:

In the EnterpriseOne System Control application (P99410), add a record for SYXXX, where XXX is the system code. Give the system code a description and set the Use Module to a value of Yes to activate the system code. The record will be added to the System Control table (F99410). To prevent the usage of this system, do not add this record or set the Use Module to No.

The following diagram shows a high-level overview of the flow for determining if a system code is active.



To programmatically determine whether a system is activated, wrap the main logic in an if statement to check that the Use Module flag is set to Y. If it is, the logic can be run; if it is not, then a message stating that the system is inactive is issued.

You can call the business function GetEnvironmentValue (B9900410) to retrieve the value from the System Control table (F99410). Pass SYXXX (where XXX is the new system code) into the szDataItem field. Then use the returned value in the cModuleExistence field to determine if the new system is activated. The GetEnvironmentValue business function returns a value of 1 in the cModuleExistence field if the record exists in the table and the Use Module field has a value of Yes. If Use Module is set to No or the record does not exist in the table, the function returns a value of 0 in the cModuleExistence field.

Using a Tag File to Add Fields to the Database

If new fields are needed for an existing table, you should create a new tag table. Never add fields to an existing table. You create a new table by following the naming standards and using the new system code. Then add the same key fields that are in the existing table to the new tag file and then add the additional new fields.

For example, do the following to add a new field to the Sales Order Header table (F4201):

- Create a new table, for example, FXXX4201, where XXX is the new system code (FQ554201).
- Add the same key fields that are in the F4201 table: Order Number, Company Key, and Order Type.
- Add any new fields using these guidelines:
 - Consider the addition of the standard audit fields to the tag table if this information would be valuable in tracking the last modification to the tag file.
 - If amount fields exist, include standard currency fields that are required for conversion to the tag table.
 - If amount fields exist that require currency conversion, choose one of these three methods for converting the amount fields in the tag file:
 - Modify the table triggers in the host table. These changes must be reapplied if a fix or upgrade overlays your changes.
 - Add currency triggers to the tag file.
 - Add logic within the application business logic to perform the currency conversion.
 - Consider adding Table Event Rules (TER) to delete records from the tag file when the corresponding record is deleted from the host table.

Note: You should not add multiple tag files to a single base table because this can create technical challenges in effectively fetching records. As part of technical maintenance, multiple tag files owned by a single provider should be combined.

The following list provides limits on table joins:

- If all joins in the business view are simple joins, the limit is five tables.
- If any one of the joins is an outer join, the limit is three tables.
- For a union business view, a limit of three tables is allowed in the business view.
- A business view cannot be created that combines the use of joins and unions.
- If the tables exist in different data sources and a cross data source join is being created, there is a limit of two tables in the business view.

Adding Processing Options to Existing Applications

You might need to add processing options to extend functionality within an existing application. To avoid possible collisions, you should create a new processing option template, UDC, and dummy application.

Use these guidelines to add processing options to existing applications:

- Create a new processing option template following the naming standards using the assigned system code (TXXXZZZZ). For example, if a new processing option is to be added to Sales Order Entry (P4210), you could create a new processing option template called TQ554210, where XXX is the new system code.
- For all DD items that you add to the new template, name the item using the processing option help text naming convention (SYYYYYYY00). The first processing option help text data dictionary item would be SQ55421001.

Note: Follow the JD Edwards EnterpriseOne Tools Development Guidelines for Application Development Guide when naming the text of the processing option itself and listing the valid values. Also make sure that every processing option has help text associated with it. If an item needs to have a visual assist, the actual added DD item must have the UDC or Search and Select form associated with the item; this cannot be done in the processing option template design.

- Next, create the dummy application using the naming standards (PXXXZZZZ). For the Sales Order Entry example, the dummy application would be PQ554210. Associate this application with the new processing option template, TQ554210. This enables user access to the new processing options. The dummy application has no functionality within it.

In order for the existing application to determine the version of the new dummy application to call to retrieve the new processing option, set up a UDC to cross-reference the version of the existing application with the version of the new dummy application. You must set up a new UDC to cross-reference the version of the existing application with the version of the new dummy application so that the existing application can determine the version of the new dummy application to call to retrieve the new processing option.

These guidelines discuss setting up the UDC:

- Create a new UDC for your system code (XXX|YY), where XXX is the new system code and YY is the code assigned for the processing option cross-reference. After YY is assigned for this use, it should never be changed. The code length size should be 10 so that it is long enough to handle version names.
- In the new UDC, you can set up the version of the existing application that is being run in the Code field and the new version of the dummy application can be listed in the Description field of the UDC. Using the Sales Order Entry example, you set up a new UDC called, Q55|VL with a code length of 10. The actual codes within this UDC consist of version names.

To cross-reference P4210|ZJDE0001 with PQ554210|YQ550001, you set up the UDC in this manner:

- Code: ZJDE0001
- Description 01: YQ550001
- To access and use the new processing option, you write code to use the current version of the existing application and retrieve the Description field from the new UDC by calling the existing function GetUDC (X0005). The new system code is passed into the szSystemCode field (XXX) and the new value given for this use (YY) is passed into the szRecordTypeCode field. Next, the version of the existing application is passed into the szUserDefinedCode field (for example, ZJDE0001). After the call to the GetUDC function, the szDescription001 field contains the version of the dummy application to use.
- After the version of the dummy application is retrieved, use the API AllocatePOVersionData to retrieve the new processing option fields. In the call to this API, the name of the new dummy application is passed into the second parameter of the API (for example, TQ554210) and the version name to be retrieved from the UDC (YQ550001) is passed into the third parameter.
- After the call to the AllocatePOVersionData API, the values from the template can be used for the new functionality.

Adding Hooks to Existing Applications

You can extend the capabilities of the host system by adding hooks to existing applications. This section discusses these hooks that you can add to the application:

- Adding a Button to a Form
- Adding Combo Box Drop-Down Menus
- Adding a New Selection to an Existing Combo Box Drop-Down Menu
- Adding a Row Exit
- Adding Links

Adding a Button to a Form

You can add a button to an application form. Place the new button in the desired location without moving the existing buttons. When you add code to the button, use the least amount of event rule (ER) code possible. For example, just having the button call a new business function is the most desired amount of new ER code.

Note: Any changes made in Form Design Aid (FDA) must be reapplied if any changes are delivered to this application.

Adding Combo Box Drop-Down Menus

Combo box drop-down menus are used frequently throughout JD Edwards EnterpriseOne to create hooks to related applications. The combo box drop-down menus are typically used on forms that do not have a menu bar that provides exits to other applications. Combo box drop-down menus reserve real estate on the form.

Use these guidelines to add a combo box drop-down menu:

- Using FDA, give the new combo box a name using the assigned system code.
 - If the combo box is to be populated by a UDC, select a data dictionary (DD) item that uses that UDC whether from a business view or not.
 - Create the new UDC table following the naming standards.
 - Either add a new DD item or use one created for your system XXX to establish a link to the new UDC table for the drop-down menu.
- Note:** Do not make changes to DD items that are owned by the host system.
- If the combo box is to be manually populated in the application (that is, not using a UDC), select a character DD item that does not use a UDC for the combo box. Add text variables that are not already in use. Use the following naming convention to add text variables for the combo box:
 - Add Item(FC XXX Drop Down, XXX1, TV YYYYYYYY, <Null>)
 - Add Item(FC XXX Drop Down, XXX2, TV YYYYYYYY, <Null>)
 - Add Item(FC XXX Drop Down, XXX3, TV YYYYYYYY, <Null>)

where FC XXX Drop Down represents the new combo box drop-down menu, XXX1, XXX2, XXX3 represent the new keys, and TV YYYYYYYY represents a unique name for the combo box selection.

Note: Always use text variables, never hard code text within the code itself.

- Next add an image for a Go button by selecting Insert on the toolbar, and then selecting Image. Give the image a name. Select Browse in the File Name field, and go to the following location:

Local Drive where B9 is stored\Release Package\res\small-go.gif

- Name the new Go button, preceding the name with the assigned system code. Make the image clickable and maintain the size ratio.
- In the Go button for the new drop-down box, add as little ER as possible.

The above changes must be reapplied if any changes are delivered to the existing application.

Adding a New Selection to an Existing Combo Box Drop-Down Menu

You can also add a new selection to an existing combo box drop-down menu. You can use one of these ways to add a new selection to an existing combo box drop-down menu:

- Text Variable
- UDC Table

Text Variable

To add a new selection to an existing combo box drop-down menu that is defined with text variables, use a key preceded by your system code, as shown in this example:

Add Item (FC XXX Drop Down, XXX1, TV YYYYYYYY, <Null>)

where FC XXX Drop Down represents the new combo box drop-down menu, XXX1 represents the new key, and TV YYYYYYYY represents a unique name for the combo box selection.

Note: Always use text variables, never hard code text within the code itself.

In the Go button for the existing combo box drop-down menu, you can add an ER to check that FC Drop Down is equal to XXX1.

UDC Table

To add a new selection to an existing combo box drop-down menu that has been set up as a UDC, add the new selection to the existing UDC table.

In the Go button for the existing combo box drop-down menu, you can add ER to check that FC Drop Down is equal to XXX1.

Note: Any changes made in FDA must be reapplied if any changes are delivered to this application. There could be contention on the new value added to the UDC table if the host system or another business partner uses the same UDC value for a different purpose in the future.

Adding Row Exits

You can add a new row exit to an existing form; however, you should use the most unique name possible. If an application does not currently use the toolbar, do not add a row exit. Add a new button instead.

Note: Any changes made in FDA must be reapplied if any changes are delivered to this application.

Adding Links

You can display links on an application by making a value in a grid column clickable or by making a static text value clickable.

When the application runs, data must exist in the grid column to be clickable. If this link is needed for every row and not every row will have data in the column, do not make the column clickable. For example, if a grid displays a key field and description but the description is not required, it is better to put a link on the key field and not the description.

Note: Any changes made in FDA must be reapplied if any changes are delivered to this application.

Adding New Versions

You can extend the capabilities of the host system by adding new application versions. You can add a new interactive version or batch version of the application.

When you add a new interactive version or a new batch version, the version name should follow the standard naming conventions (YXXXZZZZ); for example, YQ550001.

Adding a Task List and Task

In the JD Edwards EnterpriseOne system, menus are called task lists; and options on the menus are called tasks. New task lists and tasks can be added for easier access to new applications. All tasks must be added in a pristine environment. If tasks are not added in a pristine environment not all system users will be able to see the new menus.

You can add a new task list to organize a new system by placing the new applications and batch processes in a single common location. Do not change existing task lists and tasks.

To create a new task list (for example, Menu), follow these steps:

- Right-click within the EnterpriseOne Menus, and select Insert New Task.
- In the Task ID field, use the naming convention GXXX, where XXX is the new system code. Assign a task name that you want to appear as the menu description.
- On the Executable tab, select the Folder option.

To add tasks to the Menu GXXX, follow these steps:

- Under the new main task (for example, GQ55), right-click and choose Insert New Task.
- In the Task ID field, assign a number to each of the tasks that will appear in the list. For example, the first task should be 1/GQ55 and the second task should be 2/GQ55.
- Give each task name a description that you want to appear on the menu.
- After the task is added, right-click and choose Task Revisions.
- On the executable tab, select the desired task type. For example, if placing an application on the menu, select Application or if placing a UBE on the menu choose Batch.
- Enter the Application or UBE name along with the version information for the task.

As a last step, you must set up a task between the parent task item (for example, Menu) and the items that will be listed under this new task list. To set up the task relationships, follow these steps:

- On the parent task item, right-click and choose Task Relationships.
- Determine the order of the tasks and how you want them to appear under the menu by changing the presentation sequence.

Changing Displayed EnterpriseOne Carousel Icons

This section contains the following topics:

- Understanding JD Edwards EnterpriseOne Carousel Icons Extensibility
- Managing the Images
- Mapping the Images
- Additional Information

Understanding JD Edwards EnterpriseOne Carousel Icons Extensibility

It is important to understand that JD Edwards EnterpriseOne does not support an upgrade path for any customization of the Carousel icons, which were introduced in EnterpriseOne Tools Release 9.1. What this means is as you take upgrades, it is possible that custom files and folders could be overwritten. Therefore, it is important that you back up any changes you make so they can be reapplied.

Managing the Images

Image files are the pictures used on the icons that represent the various applications and UBEs. To properly relate your images to the applicable icon, these sizing and naming standards should be followed:

1. Applications

- Identify or create the new images. For full support, there must be three versions of each image.

These are the displayed dimensions for the icons. If the images have different dimensions, the browser will do its best to scale them, but for best results, follow these dimensions.

- Large: 76x76 pixels
- Medium: 38x38 pixels
- Small: 20x20 pixels

- There is a required naming standard that must be followed.

These naming standards are case sensitive. For each image file, the naming format should be as follows:

- Large: image1_large.png
- Medium: image1_medium.png
- Small: image1_small.png

2. UBEs

- Identify or create the new images. For full support, there should be 21 versions of each image.

These are the displayed dimensions for the icons. If the images have different dimensions, the browser will do its best to scale them, but for best results, follow these dimensions.

- Large: 76x76 pixels
- Medium: 38x38 pixels
- Small: 20x20 pixels

- The 21 versions are due to the various statuses supported for a UBE; S (Submitted), W (Waiting), H (Held), P (Processing), D (Done), E (Error) and Favorite Report. For favorite report there is not a status indicator in the file name.
- There is a required naming standard that must be followed.

These naming standards are case sensitive. For each image file, the naming format should be as follows:

- Large: image1_P_large.png
- Medium: image1_P_medium.png
- Small: image1_P_small.png

3. JD Edwards EnterpriseOne images are in .png format. It is recommended that any new, customized images are in the same format, but other formats like .jpg, .gif, etc. are supported. To use a different format, the default can be overwritten by adding an attribute called extension to the mapping definition element (see next section). For example "<app appld="Pxxxxx" iconFile="image1" extension="gif"/>".

4. Place new images in the deployment folder, which is located at:

```
"WebClient_WAR\WebContent\share\images\jdeicons".
```

Mapping the Images

After new images are available, you need to map specific applications, UBEs, or system codes to display the images in the carousel.

An .xml file, named icon_mapping.xml that is located in WebClient_WAR\classes\, defines which image to use.

Mapping the images includes these parameters:

- `<app appId="Pxxxxx">`—Identifies a specific application. Use commas to separate multiple applications that use this icon.
- `iconfile="image1">`—Name of the image to be used for this icon. The iconfile parameter should be the name (case sensitive) without the size and extension; for example, use "image1", not "image1_small.png".
- `<appSysCode sysCode="xx">`—Identifies a system code. All applications under that system code use that icon image. Use commas to separate multiple system codes that use this icon.
- `<report reportId="Rxxxx">`—Identifies a specific UBE. Use commas to separate multiple UBEs that use this icon.

Note: There is no supported mechanism for mapping an icon to UBEs at the system code level. For UBEs you can map only to the individual report level.

Mapping Examples

The following are examples for the icon_apping.xml file:

- Mapping one application to an image:
`<app appld="Pxxxxx" iconFile="image1"/>`
- Mapping multiple applications to an image:
`<app appld="Pxxxxx,Pyyyyy,Pzzzzz" iconFile="image1"/>`
- Mapping a system code to an image:
`<appSysCode sysCode="xx" iconFile="image1"/>`
- Mapping multiple system codes to an image:
`<appSysCode sysCode="xx,yy,zz" iconFile="image1"/>`
- Mapping one UBE to an image:
`<report reportId="Rxxxxx" iconFile="image1"/>`
- Mapping multiple UBEs to an image:
`<report reportId="Rxxxxx,Ryyyyy,Rzzzzz" iconFile="image1"/>`

Additional Information

When updating the `icon_mapping.xml` file, it is important to understand how the system determines which image to use for the icon. A hierarchical approach, starting with the most specific to the least specific, is used. The system first looks for a match at the application or UBE level and if not found, it looks for a match at the system code level and if not found, it uses the default image.

Because this customization is not supported through an upgrade path, these images and the updated `icon_mapping.xml` file must be copied to all of the affected JAS servers. The JAS servers must be restarted for them to take effect.

Note: If image files are changed for existing mappings, end users must clear their browser cache to see the new icons.

See "Accessing Carousel Container" in the *JD Edwards EnterpriseOne Foundation Guide* in the Oracle Tools library on Oracle Technology Network. <http://www.oracle.com/technetwork/documentation/jdedent-098169.html>

Using Hover Form Extensibility

This section contains the following topics:

- Understanding Hover Forms
- Creating New Hover Forms
- User Interface Guidelines
- Associating Hover Forms to Form and Grid Fields
- Enabling EnterpriseOne Hover Forms on User-Defined Data Dictionary Items
- Extending the Item Hover Forms
- Configuring Hover Forms for System-Wide Display

Understanding Hover Forms

Hover or pop-up forms are message forms that show context-based information about a form or grid field. This helps users to view information about form and grid fields without having to navigate away from their current application view. These forms were introduced in JD Edwards EnterpriseOne Tools Release 9.1.

Hover forms are simple extensions of message forms that can be designed using Form Design Aid and associated to form and grid fields (data dictionary items) on which the information needs to be shown. Hover forms can be associated with data dictionary items, either manually through Form Design Aid or through hover feature definitions in an Admin application.

Creating New Hover Forms

JD Edwards EnterpriseOne delivers a set of prebuilt hover forms that are preconfigured with Data Dictionary items through the Admin application. Users see these forms popping up on mouse action over hover-enabled controls.

Hover-enabled controls are indicated by a small red square symbol on the top left corner of the control. To see the hover forms, hover the mouse over the red square and click the write pad icon.

Use these steps to create your own hover forms and associate them to required form and grid fields:

1. Create a new interactive application using Form Design Aid.
2. Create a new form of type Message Form.
3. Select the Pop-up Form option in the Message Form Properties.
4. Define a data structure and associate it with the form.

The form interconnect data structure should comply with this defined structure:

Parameter	Type	Description
Parameter 1	String	The first parameter represents the data dictionary alias of the form or grid field on which the hover form should be displayed
Parameter 2	String	The second parameter holds the value of the form or grid field on which hover should be displayed.

Note: The parameters should be defined in the exact manner as indicated in the preceding table to leverage hover form association through the Admin application (P958973/P958974).

JD Edwards EnterpriseOne runtime passes these two values to the hover form upon a mouse hover and click action on the concerned form or grid field defined in P958974. The second parameter should always be defined as a String irrespective of the data type of the form or grid field. These two parameters are essential to enable hover on the form or grid field through P958974; however, users can define additional parameters of their choice and use them through manual Form Design Aid association.

These two parameters do not apply for association through Form Design Aid, and the data structure can be defined in any user defined format.

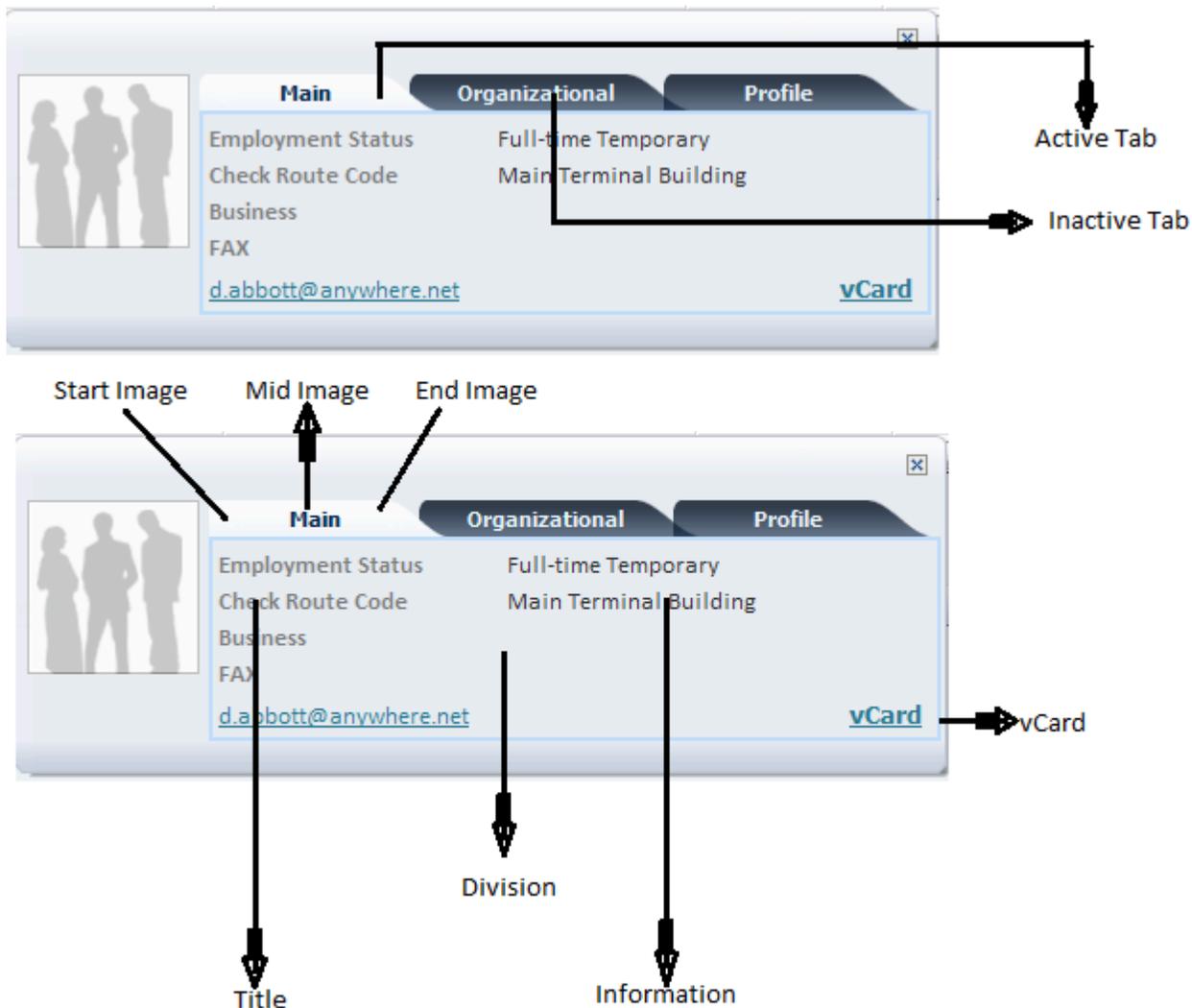
Content on hover forms can be shown using any regular form controls supported by message forms or through HTML.

EnterpriseOne delivered hover forms commonly use HTML for content display.

User Interface Guidelines

The information displayed on the JD Edwards EnterpriseOne-delivered hover forms is created using HTML. User interface (UI) features that are available on EnterpriseOne delivered hover forms are discussed in this section, and you can use these UI guidelines to impart the same look and feel to hover forms that you create.

This diagram shows the different parts of the hover form:



The following topics discuss key UI features that are available on EnterpriseOne delivered hover forms.

Tabs

These UI guidelines were used for creating Tabs:

- Tabs display a large set of information in a more structured way.
Tabs facilitate proper grouping of information and avoids the expansion in size of the hover form.
- Tabs are constructed using multiple images.
- Two types of tabs are available on the hover forms, active and inactive.
- Each tab type is represented by different sets of images.

- Start, mid, and end images are arranged in sequence to give the tabs the required look; this is illustrated in the previous diagram.
- The java script function, `showcontent()`, makes Tabs work.
When the hover form is shown, the first tab is active by default. When another tab is clicked, the function is called to arrange the images. The tab that is clicked is active and other tabs are inactive.
- To implement EnterpriseOne style tabbing, refer EnterpriseOne delivered hover form coding.

Content Display

The following hover forms are delivered with JD Edwards EnterpriseOne:

- Address Book Information Hover Form (P01700)
- Customer Information Hover Form (P03B700)
- Employee Information Hover Form (P080100)
- Supplier Information Hover Form (P04700)
- Item Master Information Hover Form (P41700)

You can open any of these forms in Form Design Aid and review the event rules. You can use the same coding style to create your hover forms.

Image

These UI guidelines were used for creating an image:

- Hover forms support runtime display of images.
- An image control must be attached to the form at design time.
- Set Default MOImage is the function used to display images at runtime.

An image must be attached (as a Media Object attachment) to the associated master record; for example, Employee master record, AB master record, Item master record, and so on. Because a master record can contain more than one image, one of the images must be designated as the default image.

vCard

These UI guidelines were used for creating a vCard:

- The vCard feature facilitates addition of a person's contact information to the email client. For example, on the Employee Hover Form, clicking the vCard link facilitates the creation of the employee's contact information to the email client.
- A vCard can be added to the hover form using `SetvCard()` system function

Associating Hover Forms to Form and Grid Fields

Associating hover forms to form and grid fields can be done in two ways, through the admin application and through Form Design Aid. Association through the Admin application is the recommended way because it removes the overhead of making code changes to target applications on which hover capability should be enabled. Secondly, hover capability can be effortlessly enabled or suppressed over a wider range of applications through the Admin application. But to use the Admin application, the form interconnect data structure on the hover forms should follow the defined pattern discussed in the previous section.

Alternatively, association through Form Design Aid can be used in cases where the input to the hover forms exceeds the number and type restrictions imposed on the form interconnect parameters. You also associate hover forms through Form Design Aid when a data dictionary item appears more than once on a form and has more than one meaning. For example, you might have data dictionary item AN8 on a form that appears more than once. One AN8 field represents an employee and another AN8 field represents a customer. You want to have different hover forms associated with these two fields.

Associating Hover Forms through the Admin Application

Use these steps to define the feature:

1. Open the Admin application P958973.
2. On Work with Feature Revisions, click Add.
3. On Runtime Feature Revisions, enter the feature name.
4. Select the Hover Viewer option.
5. Identify the data item of the form or grid field on which the hover form is to be displayed.
6. Specify the hover form's form and application names.

Use these steps to authorize the feature:

1. Open the Admin application P958974.
2. On Work with Feature Authorizations, click Add.
3. On Feature Authorization Revisions, enter the environment on which this feature should be available.
4. Enter the role information, this feature can be made available at the user, role, or *PUBLIC levels.
5. Enter the feature name from P958973, Step 3.
6. Enter the Form/Object/Product code values. The feature can be made available at the form, object, or product code level.

A sign out and sign in should enable the hover form on applications that are authorized.

Note: JD Edwards ships a set of preconfigured feature and feature authorization data to control the display of EnterpriseOne-delivered hover forms. EnterpriseOne-delivered hover forms are enabled through authorization data and not through Form Design Aid. Users can add their own set of feature and feature authorization data based on their need, which is treated as customization.

Associating Hover Forms through Form Design Aid

The hover forms can also be associated to form and grid fields manually using Form Design Aid. Use these steps to make the association:

1. Open the application on which the hover form needs to be enabled using Form Design Aid.
2. Go to Event Rules of the concerned field and select the event Mouse is Hovered.
3. Under the Mouse is Hovered event call the Show Popup (Web Only) system function.
4. Select the hover form of choice and pass the values to the data structure of the pop-up form.

Associating EnterpriseOne-delivered forms through Form Design Aid is treated as customization.

Enabling EnterpriseOne Hover Forms on User-Defined Data Dictionary Items

EnterpriseOne-delivered Address Book, Customer, Supplier and Employee hover forms can be readily associated with three basic data dictionary items: AN8, ALKY, and SSN. To facilitate association of these hover forms with any other user defined data dictionary item, the data dictionary item must be defined in the UDC, 00/AN.

For example, if the customer hover form needs to be associated with the data dictionary item SHAN, SHAN must be added to the UDC as shown here:

```
Codes: SHAN
Description 01: SHAN Alias
Special Handling: AN
Hardcoded: Y
```

Note: This is applicable only to Address Book, Customer, and Supplier, and Employee hover forms.

Extending the Item Hover Forms

Three item-based hover forms are delivered as part of EnterpriseOne hover forms: Item Sales, Item Purchase, and Item Manufacturing hover forms. These preconfigured hover forms display item information at a generic level and do not take into account information at the branch/plant level.

These forms follow the recommended form interconnect data structure standards (see [Associating Hover Forms to Form and Grid Fields](#)) to leverage runtime invocation of hover forms through the Admin application.

In addition to the first two parameters, from an extensibility standpoint, a third parameter is provided to give users a provision to display information at the branch/plant level. To achieve this, users must associate these hover forms through the Form Design Aid and pass all three parameters manually.

This table shows the data structure of the item hover form:

Parameter	Type	Description
Parameter 1	String	The first parameter represents the data dictionary alias of the form or grid field on which the hover form should be displayed.
Parameter 2	String	The second parameter holds the value of the form or grid field on which the hover should be displayed.
Parameter 3	String	Branch/Plant.

Configuring Hover Forms for System-Wide Display

As discussed in previous sections, hover forms can be configured through P958973 and P958974 Admin applications so that EnterpriseOne runtime automatically enables hover capability on a particular data dictionary item at the form/application/product (system) code or at the system level. By using the Admin applications, code changes to target applications on which the hover forms need to be displayed can be completely avoided. Form Design Aid association requires coding.

The section *Associating Hover Forms to Form and Grid Fields* provides procedures to associate hover forms through the Admin application. This section discusses general considerations to enable hover forms in a more global manner.

If a user wants to enable a hover form:

- The first step is to determine the scope of the hover form functionality and to which data dictionary items the hover forms need to be associated.
- The logic for the hover forms should be devised such that it can support display of information related to a single data dictionary item or multiple data dictionary items. For example, EnterpriseOne-delivered Address Book hover form can be associated with multiple data dictionary items: AN8, ALKY, SSN, and so on.
- Then the user needs to understand whether the hover form will be applicable at the system level, the product code level, or the application/form level.
- When following instructions in *Enabling EnterpriseOne Hover Forms on User-Defined Data Dictionary Items* using P958973, the appropriate feature needs to be defined for the hover form and the associated data dictionary items. One feature must be defined for every hover form and data dictionary item combination. For example; if ABCHover can be associated to ALKY and AN8, then two features such as ABCHover_ALKY and ABCHover_AN8 must be defined.
- Then using P958974 (Feature Authorization), the user can define authorization for every feature where a particular feature should be displayed.
- The user can define a range of authorization records from a more generic authorization record at the system code level to a very specific authorization record at the application/form level. EnterpriseOne runtime gives more precedence to a more specific authorization record compared to a generic authorization record.

This can be approached a little differently as well. Users can first figure out the set of features (hover forms) to associate to a particular data dictionary item. For example, in one system code, AN8 would represent a customer and in a different system code it would represent an employee. Furthermore, AN8 in some of the applications within a particular system code could represent a totally different feature, maybe a supplier. The user needs to understand what representation a particular data dictionary item requires across the system. Based on this analysis, different features need to be defined; for example, Customer_AN8, Employee_AN8, Supplier_AN8, and so on, each representing a different flavor of hover data for AN8 at different points in the system. Then a detailed set of authorization records for AN8 can be prepared in a spreadsheet and imported to P958974 through the EnterpriseOne grid import feature.

- Users can also define a NOHOVER feature for every data dictionary item. This feature suppresses the hover on a data dictionary item. For example, feature authorization records under NOHOVER_ALKY feature suppresses hover forms on the ALKY data dictionary item in all of the specified locations.

Adding New Auto Suggest Functionality

Oracle JD Edwards EnterpriseOne Tools Release 9.1 delivered auto suggest views. You can easily create a new one by following these steps:

1. After you identify a table and field for which you want to provide the auto suggest functionality, create a new business view to support it.
 - o From the Business View Design form, select the Auto Suggest option.
 - o From the Business View Design Aid form, select the table and the fields to be included. It is recommended to keep the number of fields low.
 - o From the Selected Columns window, select which of the columns should be search fields and which should be display fields.

Note: For performance considerations, it is recommended that any search fields be part of an index for that respective table.

2. Next you define the feature.

- o Open the Admin application P958973.
- o On Work with Feature Definitions, click Add.
- o On Runtime Feature Revisions, enter the feature name.
- o Select the Auto Suggest option.
- o Enter the data item.

This is the field on the form to which the auto suggest business view is applied.

- o Enter the business view.
- o Enter the number of characters required.

The number of characters required determines how many characters the user must type before the auto suggest feature is activated.

3. Next authorize the feature.

- o Open the Admin application P958974.
- o On Work with Feature Authorizations, click Add.
- o On Feature Authorization Revisions, enter the environment on which this feature should be available.
- o You can make this feature available at the user, role, or *PUBLIC level.
- o Enter the feature name you created when you defined the feature.
- o You can make this feature available at the form, object, or product code level.

4. Sign out and then sign in and then you should see the auto suggest feature work for the applications you authorized.

See "Enabling Auto Suggestion" the *System Administration Guide* in the *JD Edwards EnterpriseOne Tool library on Oracle Technology Network*. <http://www.oracle.com/technetwork/documentation/jdedent-098169.html>

Understanding Data Model

This section contains the following topics:

- Updating an Existing Data Model
- Creating a New Data Model

Updating an Existing Data Model

The Data Models provided with Oracle JD Edwards EnterpriseOne are created using Oracle SQL Developer Data Modeler, which can be extended to support your customizations and can be saved for future references. You should not update Oracle JD Edwards EnterpriseOne provided data models because if you apply a subsequent Oracle update for that data model, it will be overwritten and your changes will be lost. Instead, you should create a copy of the data model and make your changes to the copy.

Use these steps to extend base data models:

1. Open the Oracle SQL Developer Data Modeler.
2. Go to File Menu -> Open and select the model that you want to extend.
3. Go to File Menu -> Save As and save the model with a new name.
4. Configure the EnterpriseOne JDBC Driver with Oracle SQL Developer Data Modeler; for instructions see *Oracle Data Modeler Setup*.
5. Import the new tables that need to be extended using File -> Import -> Data Dictionary and selecting the desired tables.
6. To make subsequent changes:
 - Resize and place the newly imported tables at the appropriate place.
 - Create foreign keys as necessary between newly imported tables and existing tables.
 - Save the extended model using File->Save As so that it will not overwrite the existing base model.

Oracle JD Edwards EnterpriseOne has developed some data model standards that should be followed when extending an existing data model. This ensures a consistent look and feel across all of the data models. These standards can be found in *Data Model User Interface Guidelines*.

Additional documentation about using the Oracle SQL Developer Data Modeler and Oracle SQL Developer Data Modeler is available on the Oracle Online Documentation web page.

See additional documentation on how to use the Oracle SQL Developer Data Modeler or how to use the full functionality of the Oracle SQL Developer Data Modeler on the Oracle Online Demonstrations web page. <http://www.oracle.com/technetwork/developer-tools/datamodeler/demonstrations-224554.html>

Creating a New Data Model

One of the biggest benefits to the Oracle SQL Developer Data Modeler is how quick and easy it is to create your own customized data models. Use these steps to create a new data model:

1. Start the Oracle SQL Developer Data Modeler.

2. Configure the EnterpriseOne JDBC Driver with Oracle SQL Developer Data Modeler; for instructions see *Oracle Data Modeler Setup*.
3. Import the tables that are needed for data models using File -> Import -> Data Dictionary and selecting the desired tables.
4. Resize and arrange the imported tables at appropriate places.
5. Create foreign keys between the imported tables to denote the relationship between different tables as per the functional/data flow.
6. Save the newly created model using File -> Save.

Oracle JD Edwards EnterpriseOne has developed data model standards that should be followed when extending an existing data model. This ensures a consistent look and feel across all of the data models. These standards can be found in *Oracle Data Modeler Setup* in this document.

See additional documentation on how to use the Oracle SQL Developer Data Modeler or how to use the full functionality of the Oracle SQL Developer Data Modeler on the Oracle Online Demonstrations web page. <http://www.oracle.com/technetwork/developer-tools/datamodeler/demonstrations-224554.html>

7 Understanding Advanced Application Extensibility Design Patterns

Adding Fields to a Cache

You should avoid adding new fields to existing caches. However, if adding fields to a cache is absolutely necessary, follow these guidelines:

- Do not add fields in the middle of the business function data structure or the cache structure. All new fields must be added at the end of the structures.
- Do not change the cache key.
- Do not change the order of the cache fields.
- Rebuild the system.

After adding fields to a cache, the system requires a full rebuild.

Note: Any changes made to existing business functions must be re-applied if any changes are delivered to the business function.

Adding Logic to an Existing Application

Adding logic to event rules (ER) of an existing application is not recommended and should be avoided entirely or limited to the least amount of code as possible. If you are modifying existing ER code, try to place the new code at the end of the button or event. This makes merging or reapplying the code easier.

Delegating Authorization to another Employee

A process (privilege) like authorization is always performed by a particular employee (authorizer) in an organization. If the authorizer is out of the office for an extended period of time, JD Edwards EnterpriseOne authorization may need to be performed by another person.

Use these steps to delegate authorization to another employee:

1. Create a table to store the delegation definition.

Essential values that need to be stored are DelegationSerialNo, DelegatorId, DelegatedFrom, DelegatedTo, DelegatedApplication, EffectiveDate, ExpirationDate, ActiveDelegation, and Audit Fields.

- DelegationSerialNo—the unique number that is stored to identify a delegation entry.
- DelegatorId—the person who created the delegation entry. It can be anyone in the organization who has privilege to access and use this application.

- DelegatedFrom—the address book number of the person whose privileges are delegated.
 - DelegatedTo—the address book number of the person who is delegated with privileges of another person.
 - DelegatedApplication—the application (privilege) that is delegated from one user to another.
 - ActiveDelegation—the flag that indicates whether the delegation is active or inactive.
 - UserId, ProgramId, WorkStationId, DateUpdated, and TimeLastUpdated—the values that need to be stored for audit purposes.
2. Create a new application to query the defined delegations and to add new delegation definitions. This application can be used to add an application that needs to be delegated.

- Add a Find and Browse form that allows the user to verify that a delegation exists for the date range.

Provide header fields and QBE that allows the user to verify the existence of the delegation.

- Add a Headerless Detail form that allows the user to add a delegation definition.

Include these edit controls and grid controls in the design:

- Original Requestor—address book number of the person whose privileges are delegated.
- Delegated Requestor—address book number of the person who is delegated with privileges of another person.
- Delegation Application—the application that is delegated from one user to another. Create text variable and store all of the privileges that need to be delegated. All of the values (privileges) stored need to be displayed in the list box to allow user friendly selection.
- Effective Date and Expiration Date—date range date for which the delegation is valid.
- Delegation Active—flag to indicate whether the delegation is active or inactive.

3. Provide security restricting delegation access to administrative users.

The delegation application needs to be accessed only by administrative users to delegate privileges. Set security in P00950 to provide access to this application for administrative users.

Other required steps to enable use of delegation include:

- Modifying the application design and code.

Add processing options to enable the delegation feature.

- Using P00950 to provide users with required access.

8 Understanding Advanced Functional Extensibility Design Patterns

Understanding Prebuilt Business Function Extension Points for JD Edwards EnterpriseOne

JD Edwards EnterpriseOne provides prebuilt extension points for custom business functions that complement or replace existing system logic. Because the call to the custom business function is handled through configuration data instead of a modification to base code, custom capabilities can be created without changing a single line of JD Edwards EnterpriseOne base code.

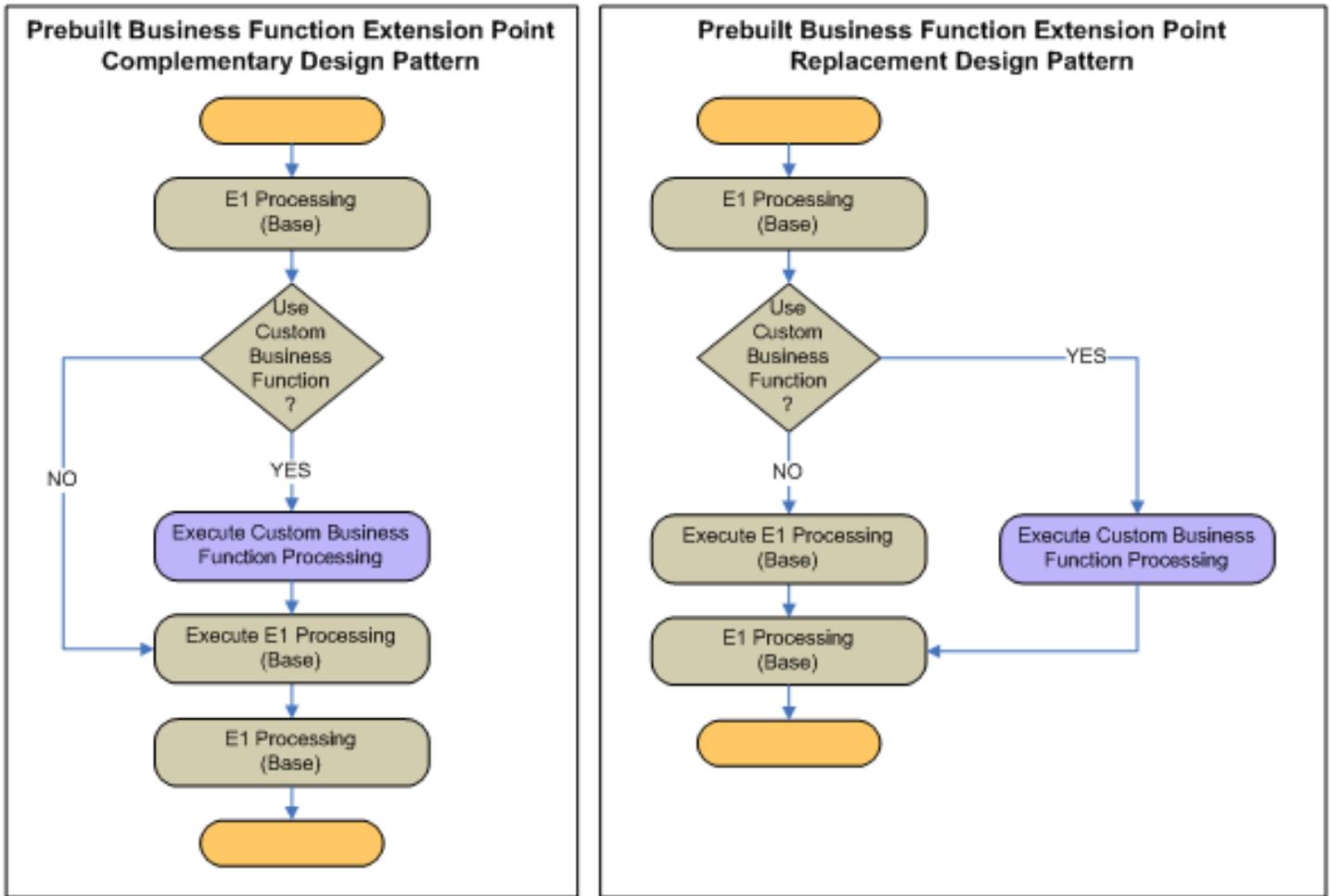
Example of How a Prebuilt Business Function Extension Point Complements Existing System Logic

When prioritizing orders within the Fulfillment Management system, in conjunction with the standard prioritization criteria (Customer Priority, Days to Requested Date, Order Type, and Line Type), it may be necessary to include Customer Category Codes as a prioritization criteria. This can be accomplished by creating a custom function to calculate the score based on the Customer Category Codes and then assigning this custom function name to the Custom Priority Calculation Function processing option value for the Sales Order Score Batch program (R4277702). The returned score is then used with the scores related to the other prioritization criteria to determine the score of the order line.

Example of Transaction How a Prebuilt Business Function Extension Point Replaces Existing Logic

When rating shipments within the Transportation Management system, an existing shipment rating system (third-party or legacy) may already exist. To enable the usage of the existing system, a rate definition that specifies a custom business function that calls the existing shipment rating system may be used. This replaces the shipment rating within JD Edwards EnterpriseOne with your existing system.

The following illustration shows processing for both the complementary and replacement design patterns:



Understanding Prebuilt Business Function Extension Points by System Code

JD Edwards EnterpriseOne provides prebuilt business function extension points that are designed for complementing or replacing standard JD Edwards EnterpriseOne functionality.

When you create a custom business function to use with the extension points listed in the following table, it is important to follow the object and business function naming standards detailed in this document. See *Object Naming Conventions* and *Other Naming Conventions*. Detailed information regarding the implementation of a custom business function for these extension points may be found by following the links listed under "Link to Additional Information" column.

The following table lists some of the prebuilt business function extension points provided by JD Edwards:

System Code	Functionality	Template Function	Template Data Structure	Custom Function Definition	Link to Additional Information
32	Configurator - External Program	Demo External Program Source File (B32EXT)	D3200000	Assembly Inclusion Edit Group Revisions (P3293/W3293A)	White Paper on My Oracle Support, "Calling an External Program from Assembly Inclusion Rules (P3293)" (Doc ID 626090.1). https://support.us.oracle.com/oip/faces/secure/km/DocumentDisplay.jspx?id=626090.1&h=Y
40G	Maturity Date Calculation	Return Maturity Date Sample Function - 57 Days (B40G2155) (Sample Function)	Specified in Add/Edit Maturity Calculation Program Name (P40G50/W40G50B)	Add/Edit Maturity Calculation Program Name (P40G50/W40G50B)	"Setting Up Maturity Date Calculations" in Chapter 2 of the <i>JD Edwards EnterpriseOne Grower Management 9.0 Implementation Guide</i> . http://docs.oracle.com/cd/E13781_01/jded/acrobat/e190AGM-B0908.pdf
41B	Bulk Conversion Tables	American Petroleum Institute Conversion Tables 23A, 23B, 23D,2 (B41B0350)	D41B0300B	Bulk Conversion Table - Conversion Table Interface Revisions (P415004)	"Understanding Conversion Table Setup" in Chapter 6 of the <i>JD Edwards EnterpriseOne Bulk Stock Inventory 9.0 Implementation Guide</i> . http://docs.oracle.com/cd/E13781_01/jded/acrobat/e190ABS-B0908.pdf
42W	Fulfillment Management Score Calculation	Client Custom Scoring Priority (B4277750)	D4277701A	Sales Order Score Batch Processing program (R4277702) processing options	White Paper on My Oracle Support, "Creating Custom Functions for JD Edwards EnterpriseOne Fulfillment Management" (1360676.1). https://support.us.oracle.com/oip/faces/secure/km/DocumentDisplay.jspx?id=1360676.1&recomm=Y
42W	Fulfillment Management Service Level Rule Evaluation	SLR Custom Rule Met Evaluation (B4277788)	D4277777A	Service Level Rule Maintenance (P4277760/W4277760A)	White Paper on My Oracle Support, "Creating Custom Functions for JD Edwards EnterpriseOne Fulfillment Management" (1360676.1). https://support.us.oracle.com/oip/faces/secure/km/DocumentDisplay.jspx?id=1360676.1&recomm=Y
45	Advanced Pricing	Calculate Price Adjustments	D4500210	Price Adjustment Detail Revisions (P4072/W4072A)	White Paper on My Oracle Support, "Creating Custom Functions for JD Edwards

System Code	Functionality	Template Function	Template Data Structure	Custom Function Definition	Link to Additional Information
		Test Function (B4500210)			EnterpriseOne Advanced Pricing" (1291414.1). https://support.us.oracle.com/oip/faces/secure/km/DocumentDisplay.jspx?id=1291414.1
46L	License Plate Segment Definition	N/A	D46L0040C	License Plate Segment Revisions (P46L010/W46L010D)	White Paper on My Oracle Support, "License Plating for 8.11 (P46L10/P46L30/ P46L31/P46L40/P46L20/R46L140/R46140/P46L99)" (1052129.1). https://support.us.oracle.com/oip/faces/secure/km/DocumentDisplay.jspx?id=1052129.1
49	Transportation Management - Rating	External Rating Function Template (B4900920)	D4900920	Rate Definition Revisions - Advanced (P4970/W4970C)	White Paper on My Oracle Support, "External Rating Function in Transportation" (656209.1). https://support.us.oracle.com/oip/faces/secure/km/DocumentDisplay.jspx?id=656209.1
49	Transportation Management - Tracking Number Generation	External Tracking Function Shell (B4901280)	D4901280	Tracking Number Segment Revisions (P49005/W49005B)	White Paper on My Oracle Support, "Pick, Pack, and Ship For ERP8 (P4620/P46091/P46020/P46011/P46471/P46013/R46472/P4621/P4915)" (625609.1). https://support.us.oracle.com/oip/faces/secure/km/DocumentDisplay.jspx?id=625609.1
49	Transportation Management - Tracking Number Check Digit Calculation	External Tracking Function Shell (B4901280)	D4901280	Tracking Number Segment Revisions (P49005/W49005B)	White Paper on My Oracle Support, "Pick, Pack, and Ship For ERP8 (P4620/P46091/P46020/P46011/P46471/P46013/R46472/P4621/P4915)" (625609.1). https://support.us.oracle.com/oip/faces/secure/km/DocumentDisplay.jspx?id=625609.1
49	Transportation Management	Shipment Tracking	D4900360	Carrier Master (P4906/W4906B)	Appendix B in this document, <i>Prebuilt Business Function</i>

System Code	Functionality	Template Function	Template Data Structure	Custom Function Definition	Link to Additional Information
	- Shipment Tracking	Template (B4900470)			<i>Extension Point for Shipment Tracking</i>
49	Transportation Management - Document Number Generation	Set External Document Number (N4900830)	D4900830	Document Setup Revisions (P49190/W49190B)	Appendix B of this document, <i>Understanding Prebuilt Business Function Extension Points by System Code</i>

9 Localization Extensibility

Understanding Localization Extensibility

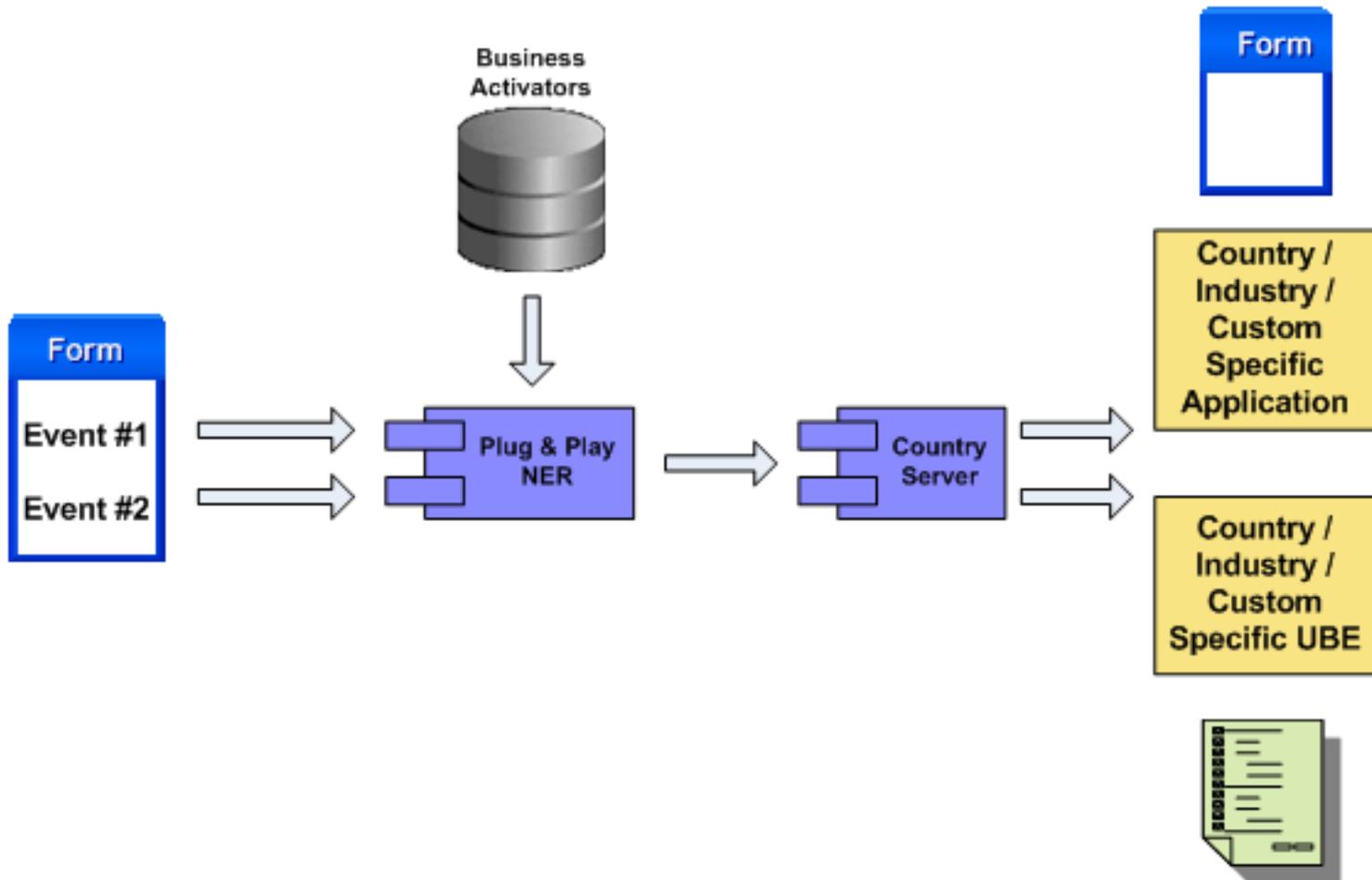
Localization extensibility is related to the functionality that is necessary to add to the software in order to support legal requirements specific to certain countries. The rules to develop a localization extensibility solution are explained in this chapter.

Localization extensibility can be achieved using plug and play and Country Server methodologies. These methodologies allow for the integration of required additional functionality to the base code with only minor modification to the standard software.

Typically, one host application is associated with one plug and play object and one country server object. If more than one provider creates a new country server for a host application, there will be more than one country server object. Using the provided naming conventions prevents object contention if there are more than one country server objects.

Before you implement a new localization extensibility solution, evaluate how your new extended solution will play with the localization solutions that are shipped in the base software. You should consider whether the base software already contains a localization solution. If the base software already contains a localization solution, you should add a new localization extensibility point. If the base software does not contain a localization solution, you should add a new country to an existing extensibility point. These solutions are discussed below.

The following diagram illustrates the basic flow of plug and play and country server methodologies:



In addition to following the information in this document, you should also follow the localization standards in the Country Server Methodology document.

See *Country Server Methodology* (Document ID 1278846.1) on My Oracle Support. <https://support.us.oracle.com/oip/faces/secure/km/DocumentDisplay.jspx?id=1278846.1>

Adding a New Localization Extensibility Point

If the base software already contains a localization solution, you should add a new localization extensibility point.

For example, if the base software already contains a localization solution for Italy and you want to extend the Italian localizations by adding an extensibility point to another host application, then you need to create a new localization extensibility point. If the base software adds an extensibility point to the same application and event in the future, the naming conventions will prevent object contention.

You can add new localization extensibility points in the following ways:

- Create a Plug and Play Data Structure.
- Create a Country Server.

- Create a Plug and Play Object.
- Create the Hook in the Core Business Logic.
- Populate the Business Activators.

Create a Plug and Play Data Structure

The plug and play data structure works in conjunction with the plug and play and country server objects. The host object uses the plug and play data structure as a communication instrument with the plug and play, country server, and country specific objects. The plug and play data structure identifies the form, event, action, and any additional parameters required to call the country-specific functionality.

The localization standards define the mandatory components for creating the plug and play data structure. You create a new plug and play data structure by following the naming standards and using the new system code, as discussed here:

Create a new data structure; for example:

DXXXZZZZ, where XXX is the new system code and ZZZZ is a unique identifier.

The description of the data structure is:

XXX-YYYYYYY—Plug and Play, where YYYYYYY is the name of the host application.

Create a Country Server

Country servers determine the logic to run for each country. A standard design pattern exists within the country server that defines the logic required for establishing the hierarchy in which calls are made. Refer to the localization standards within the Country Server Methodology document.

Use the naming standards and the new system code to create a new country server. Follow these guidelines to create a country server:

Create a new country server NER; for example:

NXXXZZZZ, where XXX is the new system code and ZZZZ is a unique identifier.

The description of the NER is:

XXX-YYYYYYY - CS - Process Localization Requirements, where YYYYYYY is the name of the host application.

Create a Plug and Play Object

The plug and play object identifies the country specified in the User Display Preferences. This enables country-specific functionality related to a host object to be invoked. The plug and play object can be used to invoke a custom or industry-specific functionality. Follow the naming standards and use the new system code to create a new plug and play object, as shown here:

Create a new plug and play NER; for example:

NXXXZZZZ, where XXX is the new system code and ZZZZ is a unique identifier.

The description of the NER is:

XXX-YYYYYYY—plug and play, where YYYYYYY is the name of the host application.

Create the Hook in the Core Business Logic

Connecting the plug and play object requires a modification to the core business logic. You create code change documentation to record the required changes.

Note: Code changes must be reapplied manually whenever the object is modified.

Populate the Business Activators

The business activators are populated through the User Profile Revisions application (P0092). This should be done as part of setup. A user is able to have only one country, industry, custom, and business partner code activated at a time. These codes should be consistent for all users working within a line of business to ensure consistency. The country code is stored in the User Display Preferences table (F00921). The industry, custom, and business partner codes are stored in the User Access Definition table (F00925).

Adding a New Country to an Existing Extensibility Point

If the base software does not contain a localization solution, you should add a new country to an existing extensibility point.

For example, if the base software does not have a solution for Canada, the extended solution can be added by either modifying an existing localization extension or adding a new localization extension point if one does not exist. Modifying an existing localization extension requires a modification to the JD Edwards EnterpriseOne business process logic. Adding a new localization extension point requires a modification to the country server. Adding a new localization extension point is less intrusive.

Adding a new country to an existing localizations extensibility point includes the following steps:

- Modify a Country Server.
- Populate the Business Activators.
- Add New Objects to Support the Extended Country Logic.

Modify a Country Server

Country servers determine the logic to run for each country. You must modify the country server to include logic for the added country. A standard design pattern exists within the country server that defines the logic required for establishing the hierarchy in which calls are made. See the localization standards in the *Country Server Methodology* (Document ID 1278846.1) on My Oracle Support. <https://support.us.oracle.com/oip/faces/secure/km/DocumentDisplay.jspx?id=1278846.1>

Follow these guidelines to modify the country server:

- Modify the existing country server to recognize the logic for the new country code.

- Create code change documentation to record the required changes.

Note: This code change must be reapplied manually whenever the object is modified.

Populate the Business Activators

The business activators are populated through P0092. This should be done as part of setup. A user is able to have only one country, industry, custom, and business partner code activated at a time. These codes should be consistent for all users working within a line of business to ensure consistency.

10 Integration Extensibility

Understanding Integration Extensibility

Extensibility is an important design principle to be considered when two disparate systems are integrated. Services that build the integration layer must be extensible to accommodate any future contract or code-based changes.

Integration extensibility allows for customization of the artifacts used in an integration. Additional processing and additional fields may be needed when implementing an integration between JD Edwards EnterpriseOne and another system. Use of these integration extension patterns helps to ensure that customizations take advantage of provided extension points. When following the suggested patterns, custom code will remain intact after updates.

Business services, real-time events, and application integration architecture (AIA) based connectors are parts of JD Edwards based integrations.

Understanding Business Service Extensibility

JD Edwards EnterpriseOne provides both consumer and provider business service (BSSV) scenarios. Consumer services enable JD Edwards EnterpriseOne to call out to third-party web services. Provider services expose web services for third-party systems to invoke. Both types of services may be extended to expose new data fields or execute new operations.

BSSV extensibility is discussed in detail in the Business Services Development Methodology Guide

See "Customizing a Published Business Service" in the *Business Services Development Methodology Guide* in the JD Edwards EnterpriseOne Applications library on Oracle Technology Network. <http://www.oracle.com/technetwork/documentation/jdedent-098169.html>

BSSV Extension Concepts

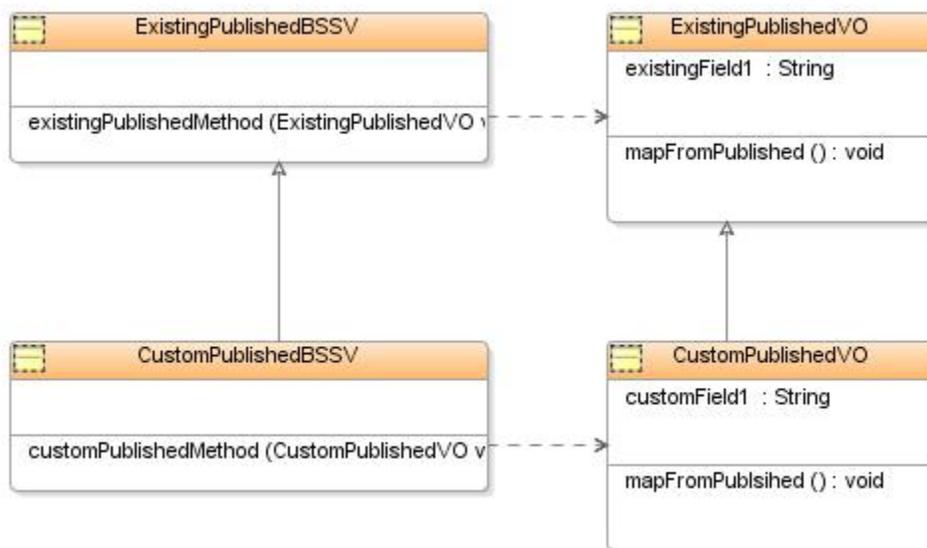
Due to the contractual nature of web services it is important to insulate changes in a manner that does not break the contract with the caller. Use these guidelines when extending JD Edwards EnterpriseOne business services:

- When you add new functionality or make code changes to functionality in existing business services, Oracle JD Edwards recommends that you create a new published business service that extends an existing business service.
- Published classes have an explicit contract. When you extend a published class, you can be sure that your customizations will continue to work when your system is updated because the published business service signature and behavior will not change when JD Edwards EnterpriseOne is updated.
- You use Object Management Workbench (OMW) to create and manage your new published business service.

Field-Level BSSV Extension

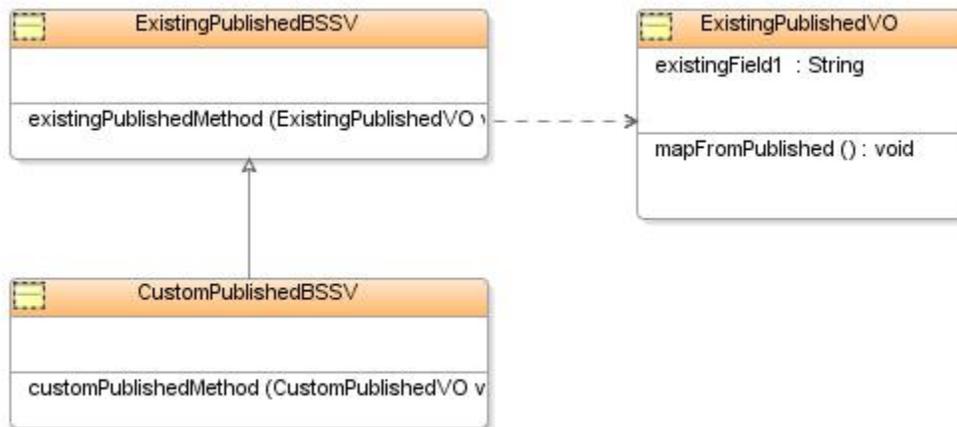
Adding additional fields to a BSSV interface is a common use case for BSSV extensibility. The biggest concern is not disrupting the existing contract for a published service. So adding fields to the interface must be accomplished through a new published business service. The new value object includes the new fields. This does not eliminate reuse of the existing BSSV object.

The following diagram illustrates the extension of a provider BSSV to include new fields. A new published BSSV is created with a value object that extends the existing published BSSV value object and includes a `mapFromPublished` method to perform the new field mapping. The existing published BSSV method is called from the custom published BSSV published method to run original functionality. The new functionality for the new field can be run before or after that call.



Function Only BSSV Extension

If the existing value object already contains the fields necessary for the custom functionality, you may extend only the published function. Any new calls can be made before or after a call to the existing published method.



Understanding Real-Time Event Extensibility

Because existing real-time events (RTE) represent a contract, they should not be modified for customizations directly. If additional published data is required, you must create a new custom event.

Before you create a custom real-time event, you should review the existing real-time events to determine if there is one that you can use as a model for creating your custom real-time event. Detail information about each real-time event can be found in the Real-Time Events Implementation Guide.

See *Real-Time Events Implementation Guide* in the JD Edwards EnterpriseOne Applications library on Oracle Technology Network. <http://www.oracle.com/technetwork/documentation/jdedent-098169.html>

If you require events beyond those that already exist, you can follow the guidelines for creating custom real-time events:

- Determine the type of real-time event (single, aggregate, or composite).
- Create a new data structure.
- Create a new event definition.
- Create a new business function or modify an existing business function to call the API that generates the event.
- Build and promote the business function.
- Add the subscriber, associate the event to the subscriber, and enable the subscription.
- Configure Object Configuration Manager (OCM) for Guaranteed Event Delivery.
- Configure and start your servers (transaction, integration, and enterprise) and test the real-time event.

Details for all of these steps for customization of real-time events are covered in the JD Edwards EnterpriseOne Tools Interoperability Guide in the Creating Custom Real-Time Events section. Concepts and information for creating a version to accommodate a change to an existing real-time event are discussed in the Versioning Real-Time Events section.

<http://www.oracle.com/technetwork/documentation/jdedent-098169.html>

Adding New Filter Criteria for Existing Real Time Events

JD Edwards EnterpriseOne Tools 9.1 is required for adding new filter criteria to existing real-time events. Oracle recommends that you do not update an Oracle JD Edwards EnterpriseOne provided application, because subsequent

Oracle fixes that you apply to that application overwrite your custom changes. Instead Oracle recommends that you create a copy of the application and make your changes to the copy.

Perform these steps to add new filter criteria:

1. Add the new fields as columns in the RT Event Filter Details table (F4231).
2. Add the new fields in the RT Event Filter Details view (V4231A).
3. Select the newly added columns from V4231A and add them in the grid in the Work With Real Time Event Filter form (W4230C) in the Work With RTE Filter application (P4230). In the grid, attach the Associated Description for the newly added fields, if any.
4. Select the newly added columns from V4231A and add them in the grid in the Real Time Event Filter Revisions form (W4230D) in the Work With RTE Filter application (P4230). In the grid, attach the Associated Description for the newly added fields, if any.
5. Add the newly added member in the Validate RTE Filter data structure, D4219030. Create the type definition of the data structure and paste it in the header file of the business function, Validate RTE Filter, B4219030.h.
6. Add the filtering logic for the newly added columns in the business function, Validate RTE Filter, B4219030.c.

Note: Ensure that the intended filters are supported by JD Edwards EnterpriseOne and are available for the target environment in the Event Definition Workbench (P90701A) application.

See the *Real-Time Events Implementation Guide* in the JD Edwards EnterpriseOne Applications library on Oracle Technology Network for information about real-time event filtering. <http://www.oracle.com/technetwork/documentation/jdedent-098169.html>

On the JD Edwards EnterpriseOne documentation page, click the View Library link for Apps Release 9.1, select the Cross-Product tab, and search for the term "filter" in the Real-Time Events Implementation Guide.

Understanding Application Integration Architecture Connector Extensibility

Application Integration Architecture (AIA) connectors delivered with the JD Edwards EnterpriseOne Process Integration Packs (PIPs) are built with specific constructs to accommodate extensions. The Application Business Service Connector (ABCS) provides these two types of extensions:

- ABCS Extension (service extension)
- XSL Extension (field level extension)

Even with this accommodation, it is important to remember these guidelines for extending the connectors.

- The JD Edwards EnterpriseOne schemas shipped with integrations (batch or real-time, as well as published services) are not to be modified in any customization because they are part of the contract for Application Business Message (ABM) to the Enterprise Business Message (EBM) mapping in the AIA connectors.
- In many of the AIA integrations, category codes and user defined fields are already included in the messages. These should be considered for use as a simpler way to customize.
- Filtering and routing is always customizable in an AIA integration and control which messages went through or where specific messages were routed based on any field in the message. This is also a simpler way to accomplish customization with minimal code impact.

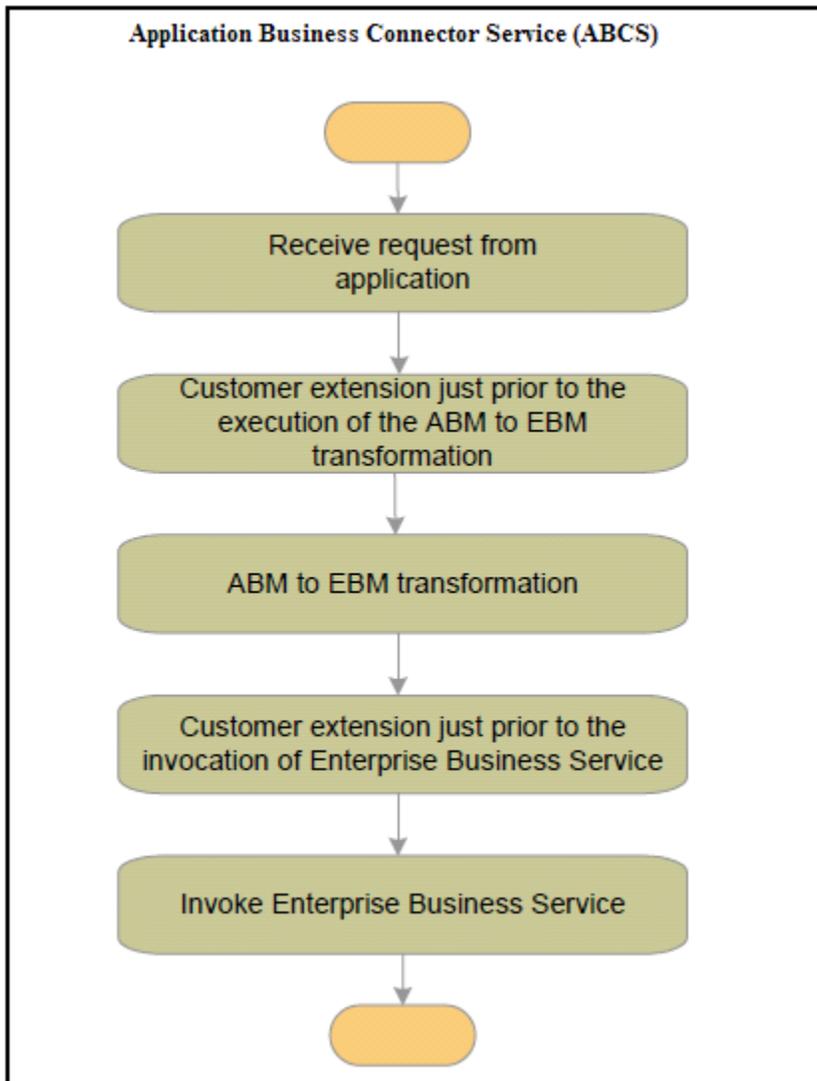
See these guides for additional information:

- For an understanding of AIA extensibility see "Understanding Extensibility" in the *Oracle Fusion Middleware Concepts and Technologies Guide for Oracle Application Integration Architecture Foundation Pack 11g Release (11.1.1.4.0)* on Oracle Technology Network. http://docs.oracle.com/cd/E17904_01/doc.1111/e17363/toc.htm
- For details about how ABCS connectors are built to ensure extensibility, see "Section 15.2 Developing Extensible ABCS" and "Section 23.3 Making Transformation Maps Extension Aware" in the *Oracle® Fusion Middleware Developer's Guide for Oracle Application Integration Architecture Foundation Pack 11g Release 1 (11.1.1.4.0)* on Oracle Technology Network. http://docs.oracle.com/cd/E17904_01/doc.1111/e17364/toc.htm
- For 10g AIA Connectors see the *Oracle Application Integration Architecture - Foundation Pack 2.5 Integration Developer's Guide* on Oracle Technology Network. http://docs.oracle.com/cd/E20059_01/doc.250/e16465.pdf

AIA Service Extension

One of the built-in mechanisms for extending an ABCS connector service is called ABCS Extension. ABCS Extension enables manipulation of the ABM or the EBM by calling custom services before the ABM or EBM are processed by a transformation. Additional processing may also be accomplished within these extensions, even without manipulating the messages.

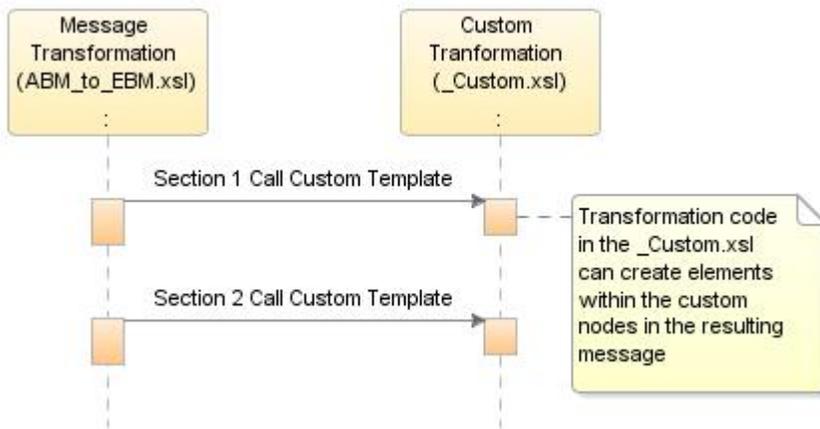
The following illustration shows this flow:



AIA Field-Level Extension

Another built-in mechanism for extending an ABCS connector service is called Extensible Stylesheet Language (XSL) Extension. XSL performs transformations from one message form to another. Extension in XSL allows for manipulation of the ABM or the EBM by calling custom transformations at key locations within each message. You can add fields to the messages within the provided custom sections.

The following illustration shows this flow:



Customization of BSSV within an AIA Integration

A provider ABCS and the BSSV that it invokes are tightly coupled to the message defined in the Web Service Definition Language (WSDL) of the BSSV. Therefore, it is even more important not to modify the JD Edwards EnterpriseOne published BSSVs that are associated with a provider ABCS.

Extensions may be accomplished by providing new published services and invoking them from an ABCS extension point.

11 Appendix A - Agile Product Lifecycle Management

Background

This appendix serves as a reference for back porting Agile-JD Edwards process integration pack (PIP), AIA RV 2.5 to a JD Edwards EnterpriseOne release earlier than 9.0. This appendix identifies known issues and workarounds, and provides troubleshooting tips.

Customer Scenario

Customers adopt the Agile-JD Edwards PIP when they:

- Use Agile PLM and JD Edwards EnterpriseOne products.
- Use Agile PLM to manage product design information.
- Use JD Edwards EnterpriseOne as a manufacturing system.

There is a need to integrate Agile PLM and JD Edwards EnterpriseOne to avoid:

- Delays in product launch.
- Compromise product quality.
- Expensive inventory write-offs.

Integrating Agile PLM and JD Edwards EnterpriseOne enables you to:

- Create products built with right specifications.
- Enable low total cost of ownership (TCO).
- Eliminate excess and obsolete inventory.

Business Flows

This example integration covers these business flows:

- *New Part/Product Release (PREL)*: While Agile is the system of record for item description, design, specifications, and many other pieces of information, the ERP system typically has many more attributes and placeholders for information than the Agile PLM system.

When the engineers finish authoring a part's attributes and design information and are ready to publish it to the manufacturing system, the part is released on an Engineering Change Order (ECO). The part definition and design information contained on the ECO is used to author the part in JD Edwards EnterpriseOne using a business process similar to one that an end user might follow.

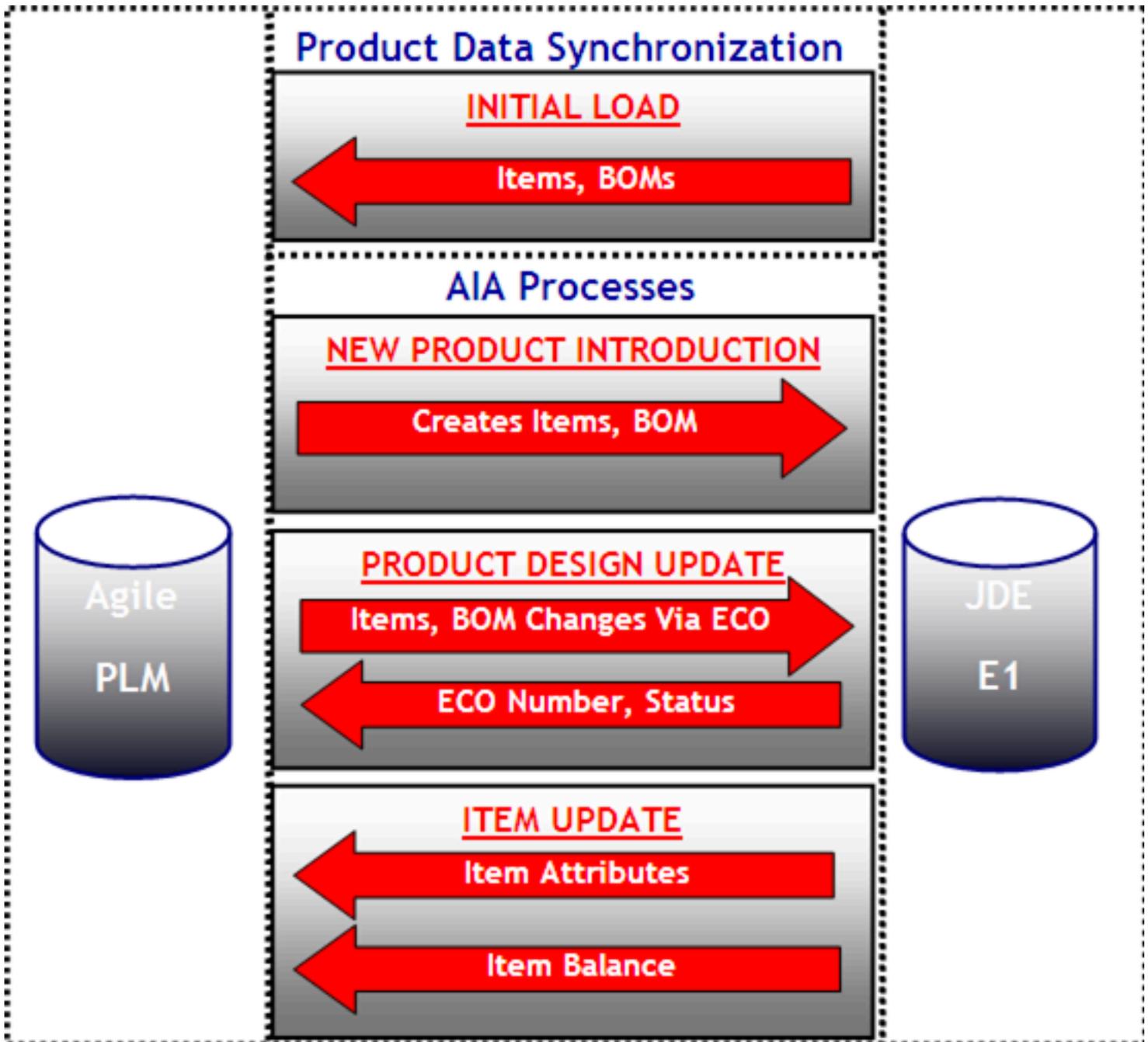
- *Change Order Release (ECO)*: During the design phase, an assembly may go through multiple design changes that are revision controlled by means of ECOs. A design change may be accompanied by a change in specifications, which may be attached as files to the item object. Ideally, this information should be synchronized with the ERP system so that the processes controlled by the ERP system—such as manufacturing planning, costing, procurement, and so on—can take into account the latest design of the assembly.

Therefore, the release of a change order in the PLM system generally acts as a trigger for the synchronization of product design information with the ERP system. Because the Agile PLM system is the system of record for product design data, the synchronization process typically involves transfer of the released revision of product design from Agile PLM to the manufacturing system.

- *Change Order Acceptance*: After an ECO has been synchronized to the ERP system, certain changes to the ECO in the ERP system must be updated back to the PLM system. Most notably, as the status of the ECO changes and moves to an implemented state in the ERP system, PLM should be notified.
- *Update Item Attributes/Unit Costs*: Certain item attributes including item cost will be updated in the PLM system when changed in the ERP system.
- *Update Item Balance*: Item balance information is used in the PLM system to facilitate the design of end items. Item balance information can periodically be synchronized from the ERP system to PLM so that the availability of components is seen in the PLM system.
- *Initial Load: Item and Bill of Material (BOM)*: When integrating Agile PLM and JD Edwards EnterpriseOne, any existing ERP Items and BOMs that will be maintained in the PLM system going forward must be loaded into the PLM System. Then as the items and BOMs are modified, they will be updated in the ERP system.

Architecture

This diagram provides a high-level overview of the integration between Agile PLM and JD Edwards EnterpriseOne:



Environment and Setup

Working in this integration requires these environments:

- JD Edwards EnterpriseOne
- Agile

- Oracle Fusion Middleware

JD Edwards EnterpriseOne

These JD Edwards EnterpriseOne components are required:

- HTML Server
- BSSV Server
- Enterprise Server
- Database
- Deployment Server
- Oracle Application Server or WebSphere Application Server

See these references for JD Edwards EnterpriseOne MTRs:

JD Edwards EnterpriseOne Tools 8.9x Minimum Technical Requirements - Collaborative Portal. https://support.oracle.com/CSP/main/article?cmd=show&type=ATT&id=705327.1:E1_PORTAL.

JD Edwards EnterpriseOne Tools 8.9x Minimum Technical Requirements for Business Services Servers. https://support.oracle.com/CSP/main/article?cmd=show&type=ATT&id=705324.1:E1_BSSV

Agile

This section discusses components that you must set up for the Agile system.

Patch to Expose Agile's AXIS Web Services

There are a couple of web services that Agile uses for the Change Order Acceptance, Update Item Attributes/Unit Costs, and Update Item Balance flows. While the ChangeABS web service is used for the Change Order Acceptance flow, ItemABS web service is used for the Item and item balance updates in Agile.

See the **Oracle Design to Release Integration Pack for Agile Product Lifecycle Management and JD Edwards EnterpriseOne Implementation Guide Release 2.5** for details about the patch.

Turning Multisite Configuration On and Off

When Agile PLM is configured to support multiple sites, the Multisite property on AIAConfigurationProperties.xml should be on for the PIP. The Multisite property can be off in the case of a no site scenario. Use these steps to configure the Multisite property:

- Change AIAConfigurationProperties.xml on the Fusion Middleware (FMW) Server under <AIA_HOME>/config.
- Search for Agile (you should find a module level configuration property called Agile).
- Update the MULTISITE_ENABLED property to TRUE if you want to turn on multisite or set it to FALSE if you want to turn off multisite.
- On the Agile Java Client, under Admin -> Settings -> Server Settings -> Licenses -> Modules tab, set Sites to Yes or No depending on whether you want to turn on/off multisite configuration.

Note: Restart the Agile server for the multisite configurations to take effect.

Setting Up Agile QueueUI through Agile Content Service

Agile QueueUI must be configured through Agile Content Services to enable Agile side messages to reach the Fusion Middleware layer. Use these steps to configure Agile QueueUI:

- Create ECO and SCO events.
- Modify Default Item Filter Filters and set Select the All Levels check box under filter tab.
If this is not checked, BOMs will not flow from Agile to JD Edwards EnterpriseOne.
- Create ECO and SCO subscribers.
- Create a new Java Messaging Service (JMS) destination.

If you are using windows FMW then the runtime ports change every time you bounce the SOA server. The OPMN port does not change. Instead of changing the port every time you restart the FMW, use this command:

```
opmn:ormi://<SOAServer>:<OPMNPort>:<instance-name>
```

User Settings

Set Privileges for ECO and SCO to enable the user to modify the released items and released changes.

Data Settings

Modify flex fields based on the multisite configuration to expose attributes on the Agile's web client to view the cost and the item balance updates from JD Edwards EnterpriseOne to Agile.

Workflow Settings

Create a new workflow for Initial Load Change Orders. Only the Initial loads from JD Edwards EnterpriseOne to Agile PLM make use of this workflow.

For more details on Setting Up Agile, see "Setting Up Agile PLM Applications", in Chapter 7 of the *Oracle Design to Release Integration Pack for Agile Product Lifecycle Management and JD Edwards EnterpriseOne Implementation Guide, Release 2.5*. http://docs.oracle.com/cd/E20059_01/doc.250/e15775.pdf

Fusion Middleware

This section discusses components that you must set up for the fusion middleware environment.

Click on these links to access information about setting up the fusion middleware component:

- Oracle 10g Database <http://www.oracle.com/technetwork/database/database10g/documentation/database10gr2-087366.html>
- Oracle SOA Suite 10g (10.1.3.x) <http://www.oracle.com/technetwork/middleware/toplink/downloads/index.html>
- Oracle SOA Suite 10g (10.1.3.x) http://docs.oracle.com/cd/B31017_01/core.1013/b28937/toc.htm <http://www.oracle.com/technetwork/middleware/soasuite/downloads/soasuitepreviousdownloads-242102.html>
- SOA 10.1.3.4 Patch Set <http://www.oracle.com/technetwork/middleware/soasuite/downloads/soasuitepreviousdownloads-242102.html>
- SOA 10.1.3.4 Patch Set (MLR8) <http://www.oracle.com/technetwork/middleware/soasuite/downloads/soasuitepreviousdownloads-242102.html>

- Oracle Service Registry 10g <http://download.oracle.com/otndocs/tech/soa/OSR103ProductDocumentation.pdf>

Post Install Configuration of Agile PLM-JD Edwards EnterpriseOne PIP

This section discusses other components that you configure for Agile PLM-JD Edwards EnterpriseOne PIP:

- Install Fix Pack 2.5
- Set Up OWSM
- Set Up Debatching
- Set Up Consumer Properties
- Set Up FTP Adapter
- Set Up Resequencer for Initial Loads
- Set Up Transaction Timeout Values
- Set Up AIS and SDK

Installing FP 2.5

Install FP 2.5 + Agile PLM to JD Edwards EnterpriseOne PIP.

Download Patch p7490384_101340_GENERIC.zip (https://support.oracle.com/epmos/faces/PatchHome?_afzLoop=901670986373897&_afzWindowMode=0&_adf.ctrl-state=fss8pglxp_4) , unzip and follow readme.txt to install the OWSM patch on the SOA server.

See "Installing the Design to Release: Agile PLM-JDE E1 PIP" in Chapter 6 of the *Oracle Application Integration Architecture Installation and Upgrade Guide, Release 2.5*. http://docs.oracle.com/cd/E20059_01/doc.250/e18557.pdf

Setting up OWSM

Create Client Agent and Server Agent and restart SOA to update the system with the changes.

Note: Copy (Ctrl + C) and Paste (Ctrl + V) doesn't work properly. For example, "_" becomes "¿" if you copy and paste content into the console.

Setting Up Debatching

Because the initial loads carry bulk data from JD Edwards EnterpriseOne to Agile, it is necessary to split the single file, which contains a large number of records, into smaller files with fewer records so that they can be consumed and processed in a more efficient manner.

In order to achieve this, include jsr173_1.0_ri.jar and jsr173_1.0_api.jar under library oracle.bpel.common in server.xml which is available under <SOA_HOME>/j2ee/<instance_name>/config/.

Setting up Consumer Properties

This list identifies the consumers that are installed as part of the PIP installation.

- ItemInitialLoadExtractJDEE1FileConsumer
- ItemInitialLoadExtractJDEE1FTPConsumer
- BillOfMaterialsInitialLoadExtractJDEE1FileConsumer
- BillOfMaterialsInitialLoadExtractJDEE1FTPConsumer
- ItemListExtractJDEE1FileConsumer
- ItemListExtractJDEE1FTPConsumer
- EngineeringChangeOrderListExtractJDEE1FileConsumer
- EngineeringChangeOrderListExtractJDEE1FTPConsumer
- ItemBalanceListExtractJDEE1FileConsumer
- ItemBalanceListExtractJDEE1FTPConsumer

Consumer artifacts that act as routing services to the actual BPEL services are available at this location:

`<Consumer_Name>_RS` under ESB Console -> AIA System -> JDE E1 (Service Group)

Note: Do NOT modify any properties on the `_RS` artifacts failing, which could cause the adapters to not work as desired.

These three properties are common to all of the consumers:

- FileLocation
 - The FileLocation property for the FTP consumers should match the path specified in the UBE processing option relative to the FTP server default directory. If the files are written to `/slot/ems1754/appmgr/jdedwards/E900/PY900/output/ST2/MultiSite/` and the FTP server default directory is `/slot/ems1754/appmgr` then the FileLocation property should be `/jdedwards/E900/PY900/output/ST2/MultiSite`.
 - After the extract programs are run in JD Edwards EnterpriseOne, the files are automatically picked up by the FTP adapters and fed into the respective BPEL services as defined in the routing rules attached to them.
 - The FileLocation property for the FileConsumers should match the location of `<SOA_HOME>/JDEE1In/`, as illustrated here:
`/slot/ems3945/oracle/product/10.1.3.1/OracleAS_1/JDEE1In.`
 - After the extract programs are run in JD Edwards EnterpriseOne, the files should be moved manually to the JDEE1 folder for these file consumer services to find and process the files.

- Batchsize

This property determines how many records are included in each batch. The default value for this property is 60.

Note: If you use a value larger than 60, the consumer services will not function properly, because Agile Integration Services (AIS) will consume more time and cause time out in the middleware.

- PollingFrequency

This property specifies the time interval in seconds when the file consumers check the fileLocation for new files to process. The default value for this property is 30.

Setting Up FTP Adapter

The FTP adapter enables the JD Edwards EnterpriseOne batch jobs to be processed directly from the JD Edwards EnterpriseOne Enterprise server. If FTP is not set up, the extract xml files must be moved manually to the FMW Server for processing. To set up the FTP adapter, the oc4j-ra.xml file must be modified at the following location:

```
<SOA_HOME>/j2ee/<instance_name>/application-deployments/default/FtpAdapter/
```

The best way to configure this file is to copy an existing connector factory and make the following changes.

- Name: eis/Ftp/JDEE1FtpAdapter.
- Host: <Host Name> (for example, den60002jems.us.Oracle.Com)
- Port: <Port #> (for example, 21)
- Username: <Username> (for example, app1754)
- Password: <Password> (for example, APP1754)
- ftpAbsolutePathBegin: <FTP Absolute path> (for example, /slot/ems1754/appmgr)
- ftpPathSeparator: <FTP Path Separator> (for example, /)
- Server type: <Windows : win, Linux : unix>

Note: Use an FTP client like Putty/WinSCP/FileZilla to connect to the enterprise server. The default path is ftpAbsolutePathBegin.

Setting Up Resequencer for Initial Loads

To set the number of threads for initial load processing, enable Resequencer by changing esb_config.ini under <SOA_HOME>/integration/esb/config/.

Setting Up Transaction Timeout Values

Because initial loads carry bulk data, the default timeout value is quite low and would result in the flow timing out if the process is incomplete at the end of 60 seconds. We recommend that you set the xa_timeout value to 360 in esb_config.ini. This setting is available under <SOA_HOME>/integration/esb/config/.

Also, in the following files, increase the timeout settings to the value indicated:

- <SOA_HOME>\j2ee\home\config\transaction-manager.xml -> Timeout Value: 420
- <SOA_HOME>\j2ee\oc4j_soa\application-deployments\orabpel\ejb_ob_engine\ orion-ejb-jar.xml -> Timeout Value: 380
- BPELConsole -> Configuration Tab -> Domain Tab -> Update syncMaxWaitTime to be 360.

Setting Up AIS and SDK

You use the InvokeAIS and InvokeSDK Java utilities to invoke an ant script to call Agile Integration Service (AIS) and Agile SDK Java utilities. For the invoke utilities to invoke ant scripts properly, the server.xml file, which is located at <SOA_HOME>/j2ee/<instance_name>/config/, should be updated by adding the ant-launcher_1.6.5.jar file to the oracle.bpel.common section.

For more details on Setting Up AIS and SDK, see "Setting Up Agile PLM Applications", in Chapter 7 of the *Oracle Design to Release Integration Pack for Agile Product Lifecycle Management and JD Edwards EnterpriseOne Implementation Guide, Release 2.5*. http://docs.oracle.com/cd/E20059_01/doc.250/e15775.pdf

Code Changes

This section summarizes the list of JD Edwards objects that you create or modify to accomplish the Agile-JD Edwards integration. These flows are discussed:

- Item Initial Load Flow
- Bill of Material Initial Load Flow
- New Part/Product Release and Change Order Release Flow
- Change Order Acceptance Flow
- Update Item Attributes/Unit Costs flow
- Update Item Balance Flow

Item Initial Load Flow

This section discusses the JD Edwards and application integration architecture (AIA) artifacts that are needed for the Item Initial Load Flow.

JD Edwards EnterpriseOne Objects

This table identifies JD Edwards EnterpriseOne objects that must be created:

Object Name	Object Description	Why is this Object Required?
R4101D3	Item Initial Load Extract to XML	Retrieves the items that are to be loaded onto Agile from JD Edwards EnterpriseOne.
V4101D3	Item Initial Load Extract View	View used by the UBE.
T4101D3	Item Initial Load Extract to XML	Processing option template used by the UBE.

This table identifies JD Edwards EnterpriseOne existing objects that are used in this flow:

Object Name	Object Description	Why is this Object Required?
B0000201	XERCES XML wrapper functions	Creates an XML file with the data provided by the UBE.

Fusion Middleware Artifacts

This table identifies AIA artifacts that are related to the Item Initial Load Flow:

Object Type	Object Name	Why is this Object Required?	Where do I find this Object?
XSD	R4101D3.xsd	JD Edwards EnterpriseOne Item Application Business Message (ABM).	http://<servername>:<portname> / AIAComponents/ApplicationObjectLibrary/JDEE1/V1/schemas
XSD	aXML.xsd	Agile Item ABM.	http://<servername>:<portname> / AIAComponents/ApplicationObjectLibrary/Agile/V1/schemas
XSD	ImportAISResult.xsd	Agile Integration Services (AIS) result ABM.	http://<servername>:<portname>/ AIAComponents/ApplicationObjectLibrary/JDEE1/V1/schemas/ AgileInitialLoad
XSD	ReleaseECOSDKResult.xsd	SDK Result ABM.	http://<servername>:<portname> / AIAComponents/ApplicationObjectLibrary/JDEE1/V1/schemas/ AgileInitialLoad
BPEL	InitialLoadItemListJDEE1toAgileImpl	To transform ABM into an XML format and invokes AIS and SDK for pushing items to Agile.	BPEL Console
Consumer	ItemInitialLoadExtractJDEE1File Consumer & ItemInitialLoadExtractJDEE1File Consumer_RS	To pick up and route R4101D3*.xml from <SOA_HOME>/JDEE1 into InitialLoadItemListJDEE1 BPEL Service	ESB Console
Consumer	ItemInitialLoadExtractJDEE1FTP Consumer & ItemInitialLoadExtractJDEE1FTP Consumer_RS	To pick up and route R4101D3*.xml from JD Edwards EnterpriseOne Enterprise Server to InitialLoadItemListJDEE1 BPEL Service	ESB Console

Setting Up Item Initial Load Extract to XML (R4101D3)

Before running the initial loads, the following must be completed:

- Set up versions of each extract program in JD Edwards EnterpriseOne.
See "Setting Up Initial Load Batch Extract Programs" in Chapter 7 of the *Oracle Design to Release Integration Pack for Agile Product Lifecycle Management and JD Edwards EnterpriseOne Implementation Guide, Release 2.5*. http://docs.oracle.com/cd/E20059_01/doc.250/e15775.pdf
- Set up batch processing information and invoke utilities for initial load.
See "Setting Up Batch Processing Information" in Chapter 7 of the *Oracle Design to Release Integration Pack for Agile Product Lifecycle Management and JD Edwards EnterpriseOne Implementation Guide, Release 2.5*. http://docs.oracle.com/cd/E20059_01/doc.250/e15775.pdf
- Create new workflow for initial load change orders and set privileges.
See "Setting up Agile PLM Applications" in Chapter 7 of the *Oracle Design to Release Integration Pack for Agile Product Lifecycle Management and JD Edwards EnterpriseOne Implementation Guide, Release 2.5*. http://docs.oracle.com/cd/E20059_01/doc.250/e15775.pdf

Running Item Initial Load Extract to XML (R4101D3)

When this UBE runs, an XML file in the format of R4101D3_MMDDYY_hhmmss.xml will be generated and placed in the path where the XML file will be written.

After the file has been written to the Enterprise Server, the appropriate consumer service retrieves and debatches the XML file. The routing service then routes each individual XML file to InitialLoadItemJDEE1toAgileImpl. This BPEL service will transform the list of items in R4101D3.xml into aXML.xsd format and write the file with the name agile<BPELInstance>.xml. The BPEL will then invoke a Java utility that will pick up the agile<BPELInstance>.xml file, zip it, and save it as agile<BPELInstance>.axml. Then the utility will invoke an ant script to call an Agile Integration Service (AIS) Java utility to import the data stored in the agile<BPELInstance>.axml file. Then another Java utility will be invoked to call an ant script to invoke yet another Java utility built with Agile SDK to release the change order.

Bill of Material Initial Load Flow

This section discusses the JD Edwards and AIA artifacts that are needed for the Bill of Material (BOM) Initial Load Flow.

JD Edwards EnterpriseOne Objects

This table identifies JD Edwards EnterpriseOne objects that must be created:

Object Name	Object Description	Why is this Object Required?
R3002D	BOM Initial Load Extract to XML	To retrieve the BOM that is to be loaded onto Agile from JD Edwards EnterpriseOne.
V3002D1	BOM Initial Load Extract view	View used by the UBE.
T3002D	BOM Initial Load Extract to XML	Processing option template used by the UBE.

Object Name	Object Description	Why is this Object Required?
Version: XJDE0001 Data Selection: <ul style="list-style-type: none"> Type Bill of Material (F3002-TBM) = M Units Batch Quantity (F3002-BQTY) = 0.0000 	BOM Initial Load Extract to XML	Version recommended when Agile PLMs system Multisite is turned off.
Version: XJDE0002 Data Selection: <ul style="list-style-type: none"> Type Bill of Material (F3002 -TBM) = M Units Batch Quantity (F3002 - BQTY) = 0.0000 Stocking Type (F4101-STKT) != N 	BOM Initial Load-Exclude Non-Stock Components	Version recommended when Agile PLMs system Multisite is turned on.

This table identifies JD Edwards EnterpriseOne existing objects that are used in this flow:

Object Name	Object Description	Why is this Object Required?
B0000201	XERCES XML wrapper functions	To create an XML file with the data provided by the UBE.

Fusion Middleware Artifacts

This table identifies AIA artifacts that are related to the BOM Initial Load Flow:

Object Type	Object Name	Why is this Object Required?	Where do I find this Object?
XSD	R3002.xsd	JD Edwards EnterpriseOne BOM Application Business Message (ABM)	http://<servername>:<portname> /AIAComponents/ ApplicationObjectLibrary/ JDEE1/V1/schemas
XSD	aXML.xsd	Agile ABM	http://<servername>:<portname> / AIAComponents/ ApplicationObjectLibrary/ Agile /V1/schemas

Object Type	Object Name	Why is this Object Required?	Where do I find this Object?
XSD	ImportAISResult.xsd	Agile Integration Services (AIS) result ABM	http:// <servername>:<portname>/ AIAComponents/ ApplicationObjectLibrary/ JDEE1/V1/schemas/ AgileInitialLoad
XSD	ReleaseECOSDKResult.xsd	SDK Result ABM	http:// <servername>:<portname> / AIAComponents/ ApplicationObjectLibrary/ JDEE1/V1/schemas/ AgileInitialLoad
BPEL	InitialLoadBillOfMaterialsListJDEE1toAgileImpl	To transform ABM into XML format and invoke AIS & SDK for pushing BOMs to Agile.	BPEL Console
Consumer	BillOfMaterialsInitialLoadExtractJDEE1File Consumer & BillOfMaterialsInitialLoadExtractJDEE1File Consumer_RS	To pick up and route R3002*.xml from <SOA_HOME>/JDEE1In to InitialLoadBillOfMaterial BPEL Service.	ESB Console
Consumer	BillOfMaterialsInitialLoadExtractJDEE1FTP Consumer & BillOfMaterialsInitialLoadExtractJDEE1FTP Consumer_RS	To pick up and route R3002*.xml from JD Edwards EnterpriseOne Enterprise Server to InitialLoadBillOfMaterial BPEL Service.	ESB Console

Setting Up Item Initial Load Extract to XML (R3002D)

Before running the initial loads, the following must be completed:

- Set up versions of each extract program in JD Edwards EnterpriseOne.

See "Setting Up Initial Load Batch Extract Programs" in Chapter 7 of the *Oracle Design to Release Integration Pack for Agile Product Lifecycle Management and JD Edwards EnterpriseOne Implementation Guide, Release 2.5*. http://docs.oracle.com/cd/E20059_01/doc.250/e15775.pdf

- Set up batch processing information and invoke utilities for initial load.

See "Setting Up Batch Processing Information" in Chapter 7 of the *Oracle Design to Release Integration Pack for Agile Product Lifecycle Management and JD Edwards EnterpriseOne Implementation Guide, Release 2.5*. http://docs.oracle.com/cd/E20059_01/doc.250/e15775.pdf

- Create new workflow for initial load change orders and set privileges.

See "Setting up Agile PLM Applications" in Chapter 7 of the *Oracle Design to Release Integration Pack for Agile Product Lifecycle Management and JD Edwards EnterpriseOne Implementation Guide, Release 2.5*. http://docs.oracle.com/cd/E20059_01/doc.250/e15775.pdf

Setting Up Item Initial Load Extract to XML (R4101D3)

Before running the initial loads, the following must be completed:

- Set up versions of each extract program in JD Edwards EnterpriseOne.

See "Setting Up Initial Load Batch Extract Programs" in Chapter 7 of the *Oracle Design to Release Integration Pack for Agile Product Lifecycle Management and JD Edwards EnterpriseOne Implementation Guide, Release 2.5*. http://docs.oracle.com/cd/E20059_01/doc.250/e15775.pdf

- Set up batch processing information and invoke utilities for initial load.

See "Setting Up Batch Processing Information" in Chapter 7 of the *Oracle Design to Release Integration Pack for Agile Product Lifecycle Management and JD Edwards EnterpriseOne Implementation Guide, Release 2.5*. http://docs.oracle.com/cd/E20059_01/doc.250/e15775.pdf

- Create new workflow for initial load change orders and set privileges.

See "Setting up Agile PLM Applications" in Chapter 7 of the *Oracle Design to Release Integration Pack for Agile Product Lifecycle Management and JD Edwards EnterpriseOne Implementation Guide, Release 2.5*. http://docs.oracle.com/cd/E20059_01/doc.250/e15775.pdf

Running BOM Initial Load Extract to XML (R3002D)

When this UBE runs, an XML file in the format of R3002D_MMDDYY_hhmmss.xml will be generated and placed in the path where the XML file will be written.

After the file has been written to the Enterprise Server, the appropriate consumer service retrieves and debatches the XML file. The routing service then routes each individual XML file to InitialLoadBillOfMaterialsJDEE1toAgileImpl. This BPEL service will transform the list of BOMs in R3002D.xml into aXML.xsd format and write the file with the name agile<BPELInstance>.xml. The BPEL will then invoke a Java utility which will pick up the agile<BPELInstance>.xml file, zip it, and save it as agile<BPELInstance>.axml. Then the utility will invoke an ant script to call an Agile Integration Service (AIS) Java utility to import the data stored in the agile<BPELInstance>.axml file. Then another Java utility will be invoked to call an ant script to invoke yet another Java utility built with Agile SDK to release the change order.

New Part/Product and Change Order Release Flow

This section discusses the JD Edwards and AIA artifacts that are needed for the New Part/Product (PREL) and Change Order Release (ECO) Flow.

JD Edwards EnterpriseOne Objects

This table identifies JD Edwards EnterpriseOne objects that must be created:

Object Name	Object Description	Why is this Object Required?
JP300000	Engineering Change Order Manager	Published Business Service that will be invoked by CreateEngineeringChangeOrderListJDEE1ProvABCSImpl.
J3000010	Engineering Change Orders Processor	Internal BSSV that calls other BSSVs for Item and ECO creation.
J3000020	Engineering Change Order Parts Lists Processor	Creates ECOs with parts lists and related items.
J4100030	Inventory Item Branch Processor	Creates Item Branch records.
B3004100	Create ECO-Wrapper	J3000020 calls this BSFN for creating ECOs.
D3004100A	Create ECO	Data structure used by B3004100.
D3004100B	Add ECO Parts List Line	N/A
D3004100C	Create ECO Related Items	N/A
D3004100D	Free Memory	N/A
B4101071	F4102 Item Branch Net Change	B4101072 calls this BSFN for gathering Item branch net change information.
D4101071A	F4102 Item Branch Net Change	Data structure used by B4101071.
B4101072	Process Item Branch Data - Wrapper	To process item branch records.
D4101072A	Item Process Data-Wrapper	Data structure used by B4101072.

This table identifies JD Edwards EnterpriseOne existing objects that are used in this flow:

Object Name	Object Description	Why is this Object Required?
J4100020	Inventory Items Processor	JP300000 calls this BSSV for inserting/updating items.

Fusion Middleware Artifacts

This table identifies AIA artifacts that are related to the PREL and ECO Flow:

Object Type	Object Name	Why is this Object Required?	Where do I find this Object?
Queue	CreateQueueService	After an ECO is released from Agile, Agile Content Service pushes the ECO to the queue through this service.	BPEL Console
Queue	QueueProcessorServiceImpl	Processes ECOs in queue and passes on the ABM to ProcessEngineeringChangeOrderAgileReqABCS	BPEL Console
EBO (XSD)	EngineeringChangeOrderEBO.xsd	Enterprise business object used in this flow.	http://<servername>:<portname>/AIAComponents/EnterpriseObjectLibrary/Core/EBO/EngineeringChangeOrder/V1/
EBM (XSD)	EngineeringChangeOrderEBM.xsd	CreateEngineeringChangeOrderList EBM operation is used in this flow.	http://<servername>:<portname>/AIAComponents/EnterpriseObjectLibrary/Core/EBO/EngineeringChangeOrder/V1/
BPEL	ProcessEngineeringChangeOrderAgileReqABCS	To transform ABM into EBM format and invoke ECO EBS for pushing ECOs to JD Edwards.	BPEL Console
EBS (WSDL)	EngineeringChangeOrderEBS	Routes EBM from ProcessEngineeringChangeOrderAgileReqABCS to CreateEngineeringChangeOrderListJDEE1ProvABCSImpl.	ESB Console
BPEL	CreateEngineeringChangeOrderListJDEE1ProvABCSImpl	Receives EBM from EngineeringChangeOrder and transforms it into ABM and invokes Engineering Change Order Manager.	BPEL Console
EBM (XSD)	EngineeringChangeOrderEBM.xsd	CreateEngineeringChangeOrderListResponseEBM operation is used for the response from JD Edwards to Agile.	http://<servername>:<portname>/AIAComponents/EnterpriseObjectLibrary/Core/EBO/EngineeringChangeOrder/V1/
EBS (WSDL)	EngineeringChangeOrderResponseEBS	Routes the response EBM from CreateEngineering	ESB Console

Object Type	Object Name	Why is this Object Required?	Where do I find this Object?
		ChangeOrderList JDEE1ProvABCSImpl to ProcessEngineering ChangeOrder AgileReqABCS	

Setting Up the PREL and ECO Flow

Before running the Create ECO Flow, the following must be completed:

- Set up Agile Content Service.

See "Setting Up Agile PLM Applications" in Chapter 7 of the *Oracle Design to Release Integration Pack for Agile Product Lifecycle Management and JD Edwards EnterpriseOne Implementation Guide, Release 2.5*. http://docs.oracle.com/cd/E20059_01/doc.250/e15775.pdf

- Set up JD Edwards EnterpriseOne web services.

See "Setting up JD Edwards EnterpriseOne Web Services" in Chapter 7 of the *Oracle Design to Release Integration Pack for Agile Product Lifecycle Management and JD Edwards EnterpriseOne Implementation Guide, Release 2.5*. http://docs.oracle.com/cd/E20059_01/doc.250/e15775.pdf

- Set up OWSM security information.

See "Setting Up Oracle Web Services Manager Security Information" in Chapter 7 of the *Oracle Design to Release Integration Pack for Agile Product Lifecycle Management and JD Edwards EnterpriseOne Implementation Guide, Release 2.5*. http://docs.oracle.com/cd/E20059_01/doc.250/e15775.pdf

Running Engineering Change Order Manager (JP300000)

When an ECO is created and approved in Agile, a workflow is initiated and the ECO is picked up by the Agile Content Service (ACS) when the workflow reaches the Released state.

The ECO is transferred to the queue controller and CreateECOListABM is provided to the Agile requestor connector. The Agile ReqABCS transforms the ABM to EBM and EngineeringChangeOrderEBS routes the EBM to the JD Edwards provider. The JD Edwards provider then transforms the EBM to ABM and invokes the EngineeringChangeOrderManager published web service. JP300000 published business service internally calls the ECO processor and the ECOPartslistProcessor. The ECO processor re-uses the existing InventoryItemsProcessor and InventoryItemBranchProcessor web services for creating and updating items and branch plant information. For creating parts list and related items, the ECOPartslistProcessor business service uses B3001400 (Create ECO business function). After the business service is processed, the web service provides the confirmEngineeringChangeOrders value object back to the JD Edwards provider. The JD Edwards provider passes the response back to the Agile requestor through EngineeringchangeOrderResponseEBS. Agile Requestor ABCS then passes the status to the queue controller.

Change Order Acceptance Flow

This section discusses the JD Edwards and AIA artifacts that are needed for the Change Order Acceptance Flow.

JD Edwards EnterpriseOne Objects

This table identifies JD Edwards EnterpriseOne objects that must be created:

Object Name	Object Description	Why is this Object Required?
R3013D	ECO Extract	To retrieve the ECOs (originated from Agile) that are implemented in JD Edwards and notify Agile about this change.
V4801A1	Join on F4801\F3013	View used by the UBE.
T3013D	ECO Extract	Processing option template used by the UBE.

This table identifies common objects that must be created for the Change Order Acceptance Flow, Update Item Attributes/Unit Costs Flow, and Update Item Balance Flow.

Object Name	Object Description	Why is this Object Required?
UDC: Product Code: 00 User Defined Codes: IE	Integration Code	To hold AGILEJDE code. Denotes Agile to JD Edwards Integration.
F0095	Integration Timestamp	IntegrationTimeStamp table stores the last successful run date and time of UBEs R3013D, R4102D and R41021D.
JP300010	Integration Timestamp Manager	JDEE1ReqABCS calls this published web service to update the timestamp information.
J3000030	Process Integration Time Stamp	Internal business service called by JP300010.

This table identifies JD Edwards EnterpriseOne existing objects that are used in the Change Order Acceptance Flow, Update Item Attributes/Unit Costs Flow, and Update Item Balance Flow:

Object Name	Object Description	Why is this Object Required?
B0000201	XERCES XML Wrapper Functions	To create an XML file with the data provided by the UBE.

Fusion Middleware Artifacts

This table identifies AIA artifacts that are related to the Change Order Acceptance Flow:

Object Type	Object Name	Why is this Object Required?	Where do I find this Object?
XSD	R3013D.xsd	JD Edwards EnterpriseOne ECO Application Business Message (ABM).	http://<servername>:<portname > / AIAComponents/ApplicationObjectLibrary/JDEE1/V1/schemas
Consumer	EngineeringChangeOrderList Extract JDEE1FileConsumer & EngineeringChangeOrderList Extract JDEE1FileConsumer_RS	To pick up and route R3013D*.xml from <SOA_HOME>/JDEE1In to UpdateEngineeringChar JDEE1ReqABCSImpl BPEL Service.	ESB Console
Consumer	EngineeringChangeOrderList Extract JDEE1FTPConsumer & EngineeringChangeOrderList Extract JDEE1FTPConsumer_RS	To pick up and route R3013D*.xml from JD Edwards EnterpriseOne Enterprise Server to UpdateEngineeringChangeOrderList JDEE1ReqABCSImpl BPEL Service.	ESB Console
EBO (XSD)	EngineeringChangeOrderEBO.xsd	Enterprise Business Object used in this flow.	http://<servername>:<portname>/ AIAComponents/EnterpriseObjectLibrary/Core/EBO/EngineeringChangeOrder/V1/
EBM (XSD)	EngineeringChangeOrderEBM.xsd	UpdateEngineeringChangeOrderList EBM operation is used in this flow.	http://<servername>:<portname>/ AIAComponents/EnterpriseObjectLibrary/Core/EBO/EngineeringChangeOrder/V1/
BPEL	UpdateEngineeringChangeOrderL JDEE1ReqABCSImpl	To transform ABM into EBM format and invokes ECO EBS for pushing the implemented state of ECOs to Agile.	BPEL Console
EBS (WSDL)	EngineeringChangeOrderEBS	Routes EBM from UpdateEngineeringChangeOrderList JDEE1ReqABCSImpl to UpdateEngineeringChangeOrderListAgilePr	ESB Console
BPEL	UpdateEngineeringChangeOrderL	Receives EBM from EngineeringChangeOrderEBS and transforms it into ABM and invokes ChangeABS web service in Agile.	BPEL Console

Object Type	Object Name	Why is this Object Required?	Where do I find this Object?
EBM (XSD)	EngineeringChangeOrderEBM.xsd	UpdateEngineeringChangeOrderListResponseEBM operation is used for the response from Agile to JD Edwards.	http://<servername>:<portname>/AIAComponents/EnterpriseObjectLibrary/Core/EBO/EngineeringChangeOrder/V1/
EBS (WSDL)	EngineeringChangeOrderResponsEBS	Routes the Response EBM from UpdateEngineeringChangeOrderListAgilePr to UpdateEngineeringChangeOrderListJDEE1ReqABCImpl.	ESB Console

Setting Up ECO Extract (R3013D)

Before running the Update ECO flow, the following must be completed:

- Set up versions of each extract program in JD Edwards EnterpriseOne.

See "Setting Up Initial Load Batch Extract Programs" in Chapter 7 of the *Oracle Design to Release Integration Pack for Agile Product Lifecycle Management and JD Edwards EnterpriseOne Implementation Guide, Release 2.5*. http://docs.oracle.com/cd/E20059_01/doc.250/e15775.pdf

- Set up batch processing information and Invoke utilities for initial load.

See "Setting up Batch Processing Information" in Chapter 7 of the *Oracle Design to Release Integration Pack for Agile Product Lifecycle Management and JD Edwards EnterpriseOne Implementation Guide, Release 2.5*. http://docs.oracle.com/cd/E20059_01/doc.250/e15775.pdf

- Set up UDCs.

See "Setting Up UDC" in Chapter 7 of the *Oracle Design to Release Integration Pack for Agile Product Lifecycle Management and JD Edwards EnterpriseOne Implementation Guide, Release 2.5*. http://docs.oracle.com/cd/E20059_01/doc.250/e15775.pdf

- Set up JD Edwards EnterpriseOne web services.

See "Setting up JD Edwards EnterpriseOne Web Services" in Chapter 7 of the *Oracle Design to Release Integration Pack for Agile Product Lifecycle Management and JD Edwards EnterpriseOne Implementation Guide, Release 2.5*. http://docs.oracle.com/cd/E20059_01/doc.250/e15775.pdf

- Set up OWSM security information.

See "Setting Up Oracle Web Services Manager Security Information" in Chapter 7 of the *Oracle Design to Release Integration Pack for Agile Product Lifecycle Management and JD Edwards EnterpriseOne Implementation Guide, Release 2.5*. http://docs.oracle.com/cd/E20059_01/doc.250/e15775.pdf

Running ECO Extract (R3013D)

When this UBE runs, an XML file in the format of R3013D_MMDDYY_hhmmss.xml will be generated and placed in the path where the XML file will be written. After the file has been written to the Enterprise Server, the appropriate

consumer service retrieves and debatches the XML file. The routing service then routes each individual XML file to UpdateEngineeringChangeOrderListJDEE1ReqABCImpl.

The UpdateEngineeringChangeOrderListJDEE1ReqABCImpl transforms the UpdateECOListABM to UpdateECOListEBM and invokes EngineeringChangeOrderEBS, which does the routing to UpdateEngineeringChangeOrderListAgileProvABCImpl. The Agile provider then transforms the UpdateECOListEBM to UpdateECOListABM and calls the Agile web service extensions for updating the change order implementation status. It then invokes EngineeringChangeOrderResponseEBS to pass the response back to the JD Edwards EnterpriseOne requestor. The JD Edwards EnterpriseOne requestor then calls the Integration Timestamp web service to update the IntegrationTimeStamp table with the last successful run date/time.

Update Item Attributes/Unit Costs Flow

This section discusses the JD Edwards and AIA artifacts that are needed for the Update Item Attributes/Unit Costs flow.

JD Edwards EnterpriseOne Objects

This table identifies JD Edwards EnterpriseOne objects that must be created:

Object Name	Object Description	Why is this Object Required?
R4102D	Item And Cost Extract	To retrieve the items that have changed in JD Edwards (originated from Agile) and notify Agile about this change.
V4101DC	Join on F4101\F4102\F4105	View used by the UBE.
T4102D	Item And Cost Extract	Processing option template used by the UBE.

Fusion Middleware Artifacts

This table identifies AIA artifacts that are related to the Update Item Attributes/Unit Costs Flow:

Object Type	Object Name	Why is this Object Required?	Where do I find this Object?
XSD	R4102D.xsd	JD Edwards EnterpriseOne Item Application Business Message (ABM).	http://<servername>:<portname > /AIAComponents/ ApplicationObjectLibrary/JDEE1/V1/schemas
Consumer	ItemListExtractJDEE1File & ItemListExtractJDEE1FileRS	To pick up and route R4102D*.xml from <SOA_HOME>/JDEE1In to UpdateItemJDEE1Req BPEL Service.	ESB Console

Object Type	Object Name	Why is this Object Required?	Where do I find this Object?
Consumer	ItemListExtractJDEE1FTI & ItemListExtractJDEE1FTI RS	To pick up and route R4102D*.xml from JD Edwards EnterpriseOne Enterprise Server to UpdateItemListsJDEE1Re BPEL Service.	ESB Console
XSD	R4102D.xsd	JD Edwards EnterpriseOne Item Application Business Message (ABM).	http://<servername>:<portname > /AIAComponents/ ApplicationObjectLibrary/JDEE1/V1/schemas
Consumer	ItemListExtractJDEE1File & ItemListExtractJDEE1File RS	To pick up and route R4102D*.xml from <SOA_HOME>/JDEE1In to UpdateItemListsJDEE1Re BPEL Service.	ESB Console
Consumer	ItemListExtractJDEE1FTI & ItemListExtractJDEE1FTI RS	To pick up and route R4102D*.xml from JD Edwards EnterpriseOne Enterprise Server to UpdateItemListsJDEE1Re BPEL Service	ESB Console
XSD	R4102D.xsd	JD Edwards EnterpriseOne Item Application Business Message (ABM).	http://<servername>:<portname> /AIAComponents/ ApplicationObjectLibrary/JDEE1/V1/schemas
BPEL	UpdateItemListsAgilePro	Receives EBM from ItemEBS and transforms it into ABM and invokes ItemABS web service in Agile.	BPEL Console
EBM (XSD)	ItemEBM.xsd	UpdateItemListsResponse operation is used for the response from Agile to JD Edwards.	http://<servername>:<portname>/ AIAComponents/ EnterpriseObjectLibrary/Core/ EBO/Item/V2/
EBS (WSDL)	ItemResponseEBS	Routes the Response EBM from UpdateItemListsAgilePro to UpdateItemListsJDEE1Re	ESB Console

Setting Up Item and Cost Extract (R4102D)

Before running the Update Item flow, the following must be completed:

- Set up versions of each extract program in JD Edwards EnterpriseOne.
See "Setting Up Initial Load Batch Extract Programs" in Chapter 7 of the *Oracle Design to Release Integration Pack for Agile Product Lifecycle Management and JD Edwards EnterpriseOne Implementation Guide, Release 2.5*. http://docs.oracle.com/cd/E20059_01/doc.250/e15775.pdf
- Set up batch processing information and invoke utilities for initial load.
See "Setting Up Batch Processing Information" in Chapter 7 of the *Oracle Design to Release Integration Pack for Agile Product Lifecycle Management and JD Edwards EnterpriseOne Implementation Guide, Release 2.5*. http://docs.oracle.com/cd/E20059_01/doc.250/e15775.pdf
- Set up UDCs.
See "Setting Up UDC" in Chapter 7 of the *Oracle Design to Release Integration Pack for Agile Product Lifecycle Management and JD Edwards EnterpriseOne Implementation Guide, Release 2.5*. http://docs.oracle.com/cd/E20059_01/doc.250/e15775.pdf
- Set up JD Edwards EnterpriseOne web services.
See "Setting up JD Edwards EnterpriseOne Web Services" in Chapter 7 of the *Oracle Design to Release Integration Pack for Agile Product Lifecycle Management and JD Edwards EnterpriseOne Implementation Guide, Release 2.5*. http://docs.oracle.com/cd/E20059_01/doc.250/e15775.pdf
- Set up OWSM security information.
See "Setting Up Oracle Web Services Manager Security Information" in Chapter 7 of the *Oracle Design to Release Integration Pack for Agile Product Lifecycle Management and JD Edwards EnterpriseOne Implementation Guide, Release 2.5*. http://docs.oracle.com/cd/E20059_01/doc.250/e15775.pdf

Running Item and Cost Extract (R4102D)

When this UBE runs, an XML file in the format of R4102D_MMDDYY_hhmmss.xml will be generated and placed in the path where the XML file will be written. After the file has been written to the Enterprise Server, the appropriate consumer service retrieves and debatches the XML file. The routing service then routes each individual XML file to UpdateItemJDEE1ReqABCSImpl.

The UpdateItemJDEE1ReqABCSImpl transforms the UpdateItemABM to UpdateItemEBM and invokes ItemEBS, which does the routing to UpdateItemAgileProvABCSImpl. The Agile provider then transforms the UpdateItemEBM to UpdateItemABM and calls the Agile web service extensions for updating the item attributes and cost information. It then invokes ItemResponseEBS to pass the response back to the JD Edwards EnterpriseOne requestor. The JD Edwards EnterpriseOne requestor then calls the Integration Timestamp web service to update the IntegrationTimeStamp table with the last successful run date/time.

Update Item Balance Flow

This section discusses the JD Edwards and AIA artifacts that are needed for the Update Item Balance Flow.

JD Edwards EnterpriseOne Objects

This table identifies JD Edwards EnterpriseOne objects that must be created:

Object Name	Object Description	Why is this Object Required?
R41021D	Item Balance Extract	To retrieve the Item balance for the list of Items in JD Edwards (originated from Agile) and notify Agile about this change.
V41021ZA	Item Balance Enquiry	View used by the UBE.
T41021D	Item Balance Extract to XML	Processing option template used by the UBE.

Fusion Middleware Artifacts

This table identifies AIA artifacts that are related to the Update Item Balance Flow:

Object Type	Object Name	Why is this Object Required?	Where do I find this Object?
XSD	R41021D.xsd	JD Edwards EnterpriseOne Item Balance Application Business Message (ABM).	http://<servername>:<portname> /AIAComponents/ ApplicationObjectLibrary/JDE E1/V1/schemas
Consumer	ItemBalanceListExtract JDEE1File Consumer & ItemBalanceListExtract DEE1File Consumer_RS	To pick up and route R41021D*.xml from <SOA_HOME>/JDEE1In to UpdateItemBalanceListJD ABCSImpl BPEL Service.	ESB Console
Consumer	ItemBalanceListExtract JDEE1FTP Consumer & ItemBalanceListExtract JDEE1FTP Consumer_RS	To pick up and route R41021D*.xml from JD Edwards EnterpriseOne Enterprise Server to UpdateItemBalanceListJD ABCSImpl BPEL Service.	ESB Console
EBO (XSD)	ItemBalanceEBO.xsd	Enterprise business object used in this flow.	http://<servername>:<portname> / AIAComponents/ EnterpriseObjectLibrary/Core/ EBO/ItemBalance/V1/
EBM (XSD)	ItemBalanceEBM.xsd	UpdateItemBalanceListEB operation is used in this flow.	http://<servername>:<portname> / AIAComponents/ EnterpriseObjectLibrary/Core/ EBO/ItemBalance/V1/
BPEL	UpdateItemBalanceListJDE ABCSImpl	To transform ABM into EBM format and invokes ItemBalance EBS for pushing the updated Item balance to Agile.	BPEL Console
EBS (WSDL)	ItemBalanceEBS	Routes EBM from UpdateItemBalanceListJD ABCSImpl to	ESB Console

Object Type	Object Name	Why is this Object Required?	Where do I find this Object?
		UpdateItemBalanceListAg ABCSImpl.	
BPEL	UpdateItemBalanceListAg ABCSImpl	Receives EBM from ItemBalanceEBS and transforms it into ABM and invokes ItemABS web service in Agile.	BPEL Console
EBM (XSD)	ItemBalanceEBM.xsd	UpdateItemBalanceListRe EBM operation is used for the response from Agile to JD Edwards.	http://<servername>:<portname>/ AIAComponents/ EnterpriseObjectLibrary/Core/ EBO/ItemBalance/V1/
EBS (WSDL)	ItemBalanceResponse EBS	Routes the Response EBM from UpdateItemBalanceListAg ABCSImpl to UpdateItemBalanceListJD ABCSImpl	ESB Console

Setting Up Item Balance Extract (R41021D)

Before running the Update Item Balance Flow, the following must be completed:

- Set up versions of each extract program in JD Edwards EnterpriseOne.
See "Setting Up Initial Load Batch Extract Programs" in Chapter 7 of the *Oracle Design to Release Integration Pack for Agile Product Lifecycle Management and JD Edwards EnterpriseOne Implementation Guide, Release 2.5*. http://docs.oracle.com/cd/E20059_01/doc.250/e15775.pdf
- Set up batch processing information and invoke utilities for initial load.
See "Setting Up Batch Processing Information" in Chapter 7 of the *Oracle Design to Release Integration Pack for Agile Product Lifecycle Management and JD Edwards EnterpriseOne Implementation Guide, Release 2.5*. http://docs.oracle.com/cd/E20059_01/doc.250/e15775.pdf
- Set up UDCs.
See "Setting Up UDC" in Chapter 7 of the *Oracle Design to Release Integration Pack for Agile Product Lifecycle Management and JD Edwards EnterpriseOne Implementation Guide, Release 2.5*. http://docs.oracle.com/cd/E20059_01/doc.250/e15775.pdf
- Set up JD Edwards EnterpriseOne web services.
See "Setting Up JD Edwards EnterpriseOne Web Services" in Chapter 7 of the *Oracle Design to Release Integration Pack for Agile Product Lifecycle Management and JD Edwards EnterpriseOne Implementation Guide, Release 2.5*. http://docs.oracle.com/cd/E20059_01/doc.250/e15775.pdf
- Set up OWSM security information.
See "Setting Up Oracle Web Services Manager Security Information" in Chapter 7 of the *Oracle Design to Release Integration Pack for Agile Product Lifecycle Management and JD Edwards EnterpriseOne Implementation Guide, Release 2.5*. http://docs.oracle.com/cd/E20059_01/doc.250/e15775.pdf

Running Item Balance Extract (R41021D)

When this UBE runs, an XML file in the format of R41021D_MMDDYY_hhmmss.xml will be generated and placed in the path where the XML file will be written. After the file has been written to the Enterprise Server, the appropriate consumer service retrieves and debatches the XML file. The routing service then routes each individual XML file to UpdateItemBalanceListJDEE1ReqABCImpl.

The UpdateItemBalanceListJDEE1ReqABCImpl transforms the UpdateItemBalanceListABM to UpdateItemBalanceListEBM and invokes ItemBalanceEBS, which does the routing to UpdateItemBalanceListAgileProvABCImpl. The Agile provider then transforms the UpdateItemBalanceListEBM to UpdateItemBalanceListABM and calls the Agile web service extensions for updating the item balance information. It then invokes ItemBalanceResponseEBS to pass the response back to the JD Edwards EnterpriseOne requestor. The JD Edwards EnterpriseOne requestor then calls the Integration Timestamp web service to update the IntegrationTimeStamp table with the last successful run date/time.

Naming Conventions

This table discusses naming conventions for business services (BSSV), business functions (BSFN) and universal batch engines (UBE):

Object Type	Object Naming Standard	Description
Published Business Service	JP, system code, and numbers (JPXXZZZZ) Example: JP010000 Name must be eight characters.	JP -> Defines the object as a published business service.
Business Service	J, system code, and numbers (JXXZZZZZ) Example: J0100001 Name must be eight characters.	J -> Defines the object as a business service.
Table	FXXZZZ Example: F0095	F -> Defines the object as a file or table.
Business View	VXXZZZZZ Example: VQ551000 The business view name should be the same as the file or table except for the first character. Table FQ5501 will have business	V -> Defines the object as an UBE.

Object Type	Object Naming Standard	Description
	views VQ5501A, VQ5501B, VQ5501C, and so on.	
Processing Option Template	<p>TZZZZZZ</p> <p>Example:TQ551000</p> <p>The processing option name should be the same as the application or UBE except for the first character. Application PQ551000 will have processing option template TQ551000.</p>	<p>T -> Defines the objects as a processing option template.</p> <p>The system code should be incorporated into the name of your application or UBE.</p>
Universal Batch Engine (UBE)	<p>RXXXZZZZ</p> <p>Example: RQ551000</p>	R -> Defines the object as a UBE.
C Business Function	<p>B XXXZZZZ</p> <p>Example: BQ551000</p>	B -> Defines the object as a C business function.
BSFN Data Structure	<p>DXXXZZZZ</p> <p>Example:</p> <p>DQ551000</p> <p>The data structure name should be the same as the business function name except for the first character. Business Function BQ550010 will have data structure DQ550010.</p>	D -> Defines the object as a data structure.
Processing Option Versions	<p>External:</p> <p>YXXXZZZZ</p> <p>Example: YQ550001</p> <p>Internal JDE:</p> <p>ZJDEZZZZ</p> <p>XJDEZZZZ</p> <p>RISZZZZ</p> <p>Example: ZJDE0001</p> <p>When you name versions, do not use a Z or X for the first letter. These versions</p>	<p>External:</p> <p>Y -> Defines the object as an external processing option version.</p> <p>Internal JDE:</p> <p>ZJDE, XJDE, RIS -> Defines the object as an internal processing option version.</p>

Object Type	Object Naming Standard	Description
	<p>are reserved for Oracle. There are also restrictions on the RIS versions that are Owned by Accelerator (formerly Rapid Start).</p> <p>You should create versions using Y and then your system code.</p>	

Note: In this table, XXX is your system code and ZZZ is a sequential number for your file.

If you require additional naming convention information, see these guides in the JD Edwards EnterpriseOne Tools Guides Release 8.98 library. http://docs.oracle.com/cd/E13780_01/jded/html/docset.html

- BSSV

JD Edwards EnterpriseOne Tools 9.98 Business Services Development Guide. http://docs.oracle.com/cd/E13780_01/jded/acrobat/E1_TOOLS898TDE-B0908.pdf

JD Edwards EnterpriseOne Tools 8.98 Business Services Development Methodology Guide. http://docs.oracle.com/cd/E13780_01/jded/acrobat/E1_TOOLS898TME-B0908.pdf

- BSFN

JD Edwards EnterpriseOne Tools 8.98 Development Tools: APIs and Business Functions Guide. http://docs.oracle.com/cd/E13780_01/jded/acrobat/E1_TOOLS898TBF-B0908.pdf

- UBE

JD Edwards EnterpriseOne Tools 8.98 Development Tools: Report Design Aid Guide. http://docs.oracle.com/cd/E13780_01/jded/acrobat/E1_TOOLS898TRD-B0908.pdf

JD Edwards EnterpriseOne Tools 8.98 Development Tools: Batch Versions Guide. http://docs.oracle.com/cd/E13780_01/jded/acrobat/E1_TOOLS898TBV-B0908.pdf

Reference Links

These links provide access to additional documentation:

- JD Edwards EnterpriseOne Tools Guides Release 8.98 library. http://docs.oracle.com/cd/E13780_01/jded/html/docset.html
- *Oracle|Agile Product Lifecycle Management Release v9.2.2.6 Documentation Library.* http://docs.oracle.com/cd/E14190_02/otn/docset.html

You can access the following installation and implementation documents for Agile to JD Edwards EnterpriseOne PIP from the Oracle application Integration Architecture documentation Release 2.5 library (http://docs.oracle.com/cd/E20059_01/index.htm) or from the Oracle Software Delivery Cloud web page (<https://edelivery.oracle.com/>).

- *Oracle Application Integration Architecture Installation and Upgrade Guide* (common for all PIPs which were part of FP 2.5) http://docs.oracle.com/cd/E20059_01/doc.250/e18557.pdf
- *Oracle Design to Release Integration Pack for Agile Product Lifecycle Management and JD Edwards EnterpriseOne Implementation Guide*. http://docs.oracle.com/cd/E20059_01/doc.250/e15775.pdf

The above links are to the respective guide on the Oracle Technology Network Documentation library. The following shows how to download the installation and implementation guides from the Oracle Software Delivery Cloud web page (<https://edelivery.oracle.com/>):

Select Oracle AIA and Linux x86 when prompted and take the link shown below to get the 2.5 guides.

Select a Product Pack ▼ ⓘ

Platform ▼

Results

Select	Description	Release	Part Number	Updated	# Parts / Size
<input checked="" type="radio"/>	Oracle Application Integration Architecture Release 2.5 Media Pack for Linux x86	2.5.0.0.0	B57054-04	FEB-04-2010	174 / 73G

Installation guide: Oracle Application Integration Architecture 2.5: Installation and Upgrade Guide (common for all PIP's which were part of FP 2.5).

<input type="button" value="Download"/>	Oracle Application Integration Architecture 2.5: Installation and Upgrade Guide	V19322-01	3.3M
---	---	-----------	------

Implementation Guide: Oracle Design to Release Integration Pack for Agile Product Lifecycle Management and JD Edwards EnterpriseOne 2.5 - Implementation Guide

<input type="button" value="Download"/>	Oracle Design to Release Integration Pack for Agile Product Lifecycle Management and JD Edwards EnterpriseOne 2.5 - Implementation Guide	V18232-01	3.1M
---	--	-----------	------

Known Issues and Workarounds

This section documents known issues and workarounds. For more details refer to *Known Issues and Workarounds for Oracle Design to Release Integration Pack for Agile Product Lifecycle Management and JD Edwards EnterpriseOne 2.5* (Document ID 958495.1) on My Oracle Support. <https://support.us.oracle.com/oip/faces/secure/km/DocumentDisplay.jspx?id=958495.1>

Known Issue:

A bug occurs when using the WAS transformation and connectors for business services running on WebSphere. The bug occurs only when sourcing an ECO from Agile PLM that contains more than one new item that does not exist in the JD Edwards EnterpriseOne Item Master table. When new items are created as part of the create ECO process, the connector must create proper XREF records for the newly created items in EnterpriseOne. If more than one new item is

created, the transformation creates a combination XREF record for the new items, which causes future updates of these items to fail during XREF lookup.

Workaround: A patch has been created for this bug, download Patch - **8982299**. https://updates.oracle.com/Orion/Services/download/p8982299_25_Generic.zip?aru=12018874&patch_file=p8982299_25_Generic.zip

Troubleshooting

These troubleshooting tips are provided to help you resolve issues when implementing Agile-JD Edwards EnterpriseOne PIP.

- If you find any of the Agile provider connectors (UpdateEngineeringChangeOrderListAgileProvABCImpl, UpdateItemBalanceListAgileProvABCImpl, UpdateItemBalanceListAgileProvABCImpl) faulting, it could be due to the fact that the Agile's web services aren't exposed. To expose the web services, apply Agile's patch.
- Agile Patching: If you are unable to apply the Agile patch for exposing Agile's web services, it could be due to the fact that your system is running an older version of ANT. Perform the following steps to install the patch successfully:

```
<PROMPT> export ANT_HOME=/slot/ems3780/oracle/product/10.1.3.1/OracleAS_1/ant (Agile's SOA Home)
```

```
<PROMPT> export ANT_OPTS="-Xms512M -Xmx1024M -XX:MaxPermSize=256M"
```

```
<PROMPT> export JAVA_HOME=/slot/ems3780/oracle/product/10.1.3.1/OracleAS_1/jdk (Agile's Java Home)
```

```
<PROMPT> export PATH=.:$ANT_HOME/bin:$PATH
```

```
<PROMPT>ant -version
```

It should return : Apache Ant version 1.6.5 compiled on June 2 2005

- When you create an ECO in Agile and if you cannot find that ECO in the queue, test the JMS Destination and ensure that it is successful. If it is successful, reset the destination and reset the subscribers. If that doesn't help restart Agile and SOA Server.
- If you find instances of "CreateQueueService" on the BPELConsole it ensures that the queue is working. However if you cannot find instances of "ProcessEngineeringChangeOrderAgileReqABC", it could be due to the fact that the queue is in "Suspend" mode. Click on "Resume" to solve this issue.
- If "CreateEngineeringChangeOrderListJDEE1ProvABCImpl" faults with "Bad response: 404 Not Found" it would mean that the "Routing.EngineeringChangeOrderManager.JDEE1_01.EndpointURI" property under AIAConfigurationProperties.xml isn't working. Update the correct URL to get rid of this error.
- If "CreateEngineeringChangeOrderListJDEE1ProvABCImpl" faults with "EnterpriseOne Authentication Failure" it would mean that the OWSM Configuration isn't proper. You could remove and re-configure the client and server agents.
- "InitialLoadItemBalanceListJDEE1toAgileImpl" and "InitialLoadBillOfMaterialsListJDEE1toAgileImpl" services look for Workflow for "Initial Load Change Orders" on the agile system. If this workflow isn't available these BPEL services will fault with the appropriate error messages.
- The branch plants which are part of the Initial load payloads should be available under the DVM "Agile_target_site_mapping". If any of these values aren't available, the initial load BPEL services will fault with the appropriate error messages.

- If you have Initial load services faulting during invocation of "InvokeAIS" or "InvokeSDK" it could be due to the fact that the reference to "ant-launcher_1.6.5.jar" isn't available on the server.xml. Add a reference in the server.xml as documented and it would get rid of the invocation error.

12 Appendix B - Prebuilt Business Function Extension Points for Transportation Management System

Prebuilt Business Function Extension Point for Shipment Tracking

Utilizing shipment tracking requires the implementation of custom business functions and therefore is considered a strictly custom process that is not supported by Oracle Support Services. This section provides a high-level overview of the external business function functionality.

Overview

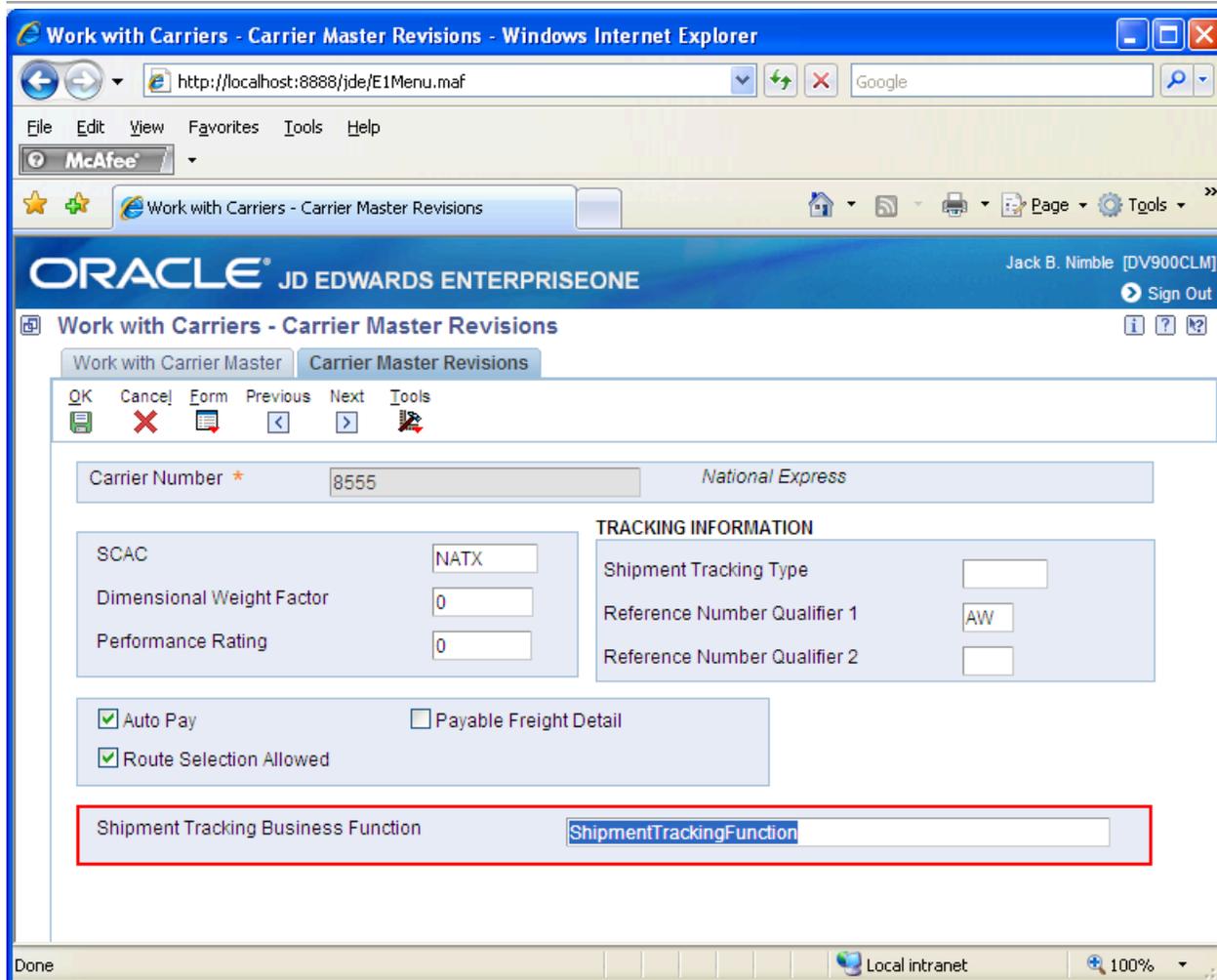
The Transportation Management system enables users to track their shipments by exiting to the URL associated with their carrier.

A custom function must be created to pass the shipment reference number (and any other information required by the carrier) to the carrier's URL to allow display of the shipment tracking information.

Setup Carrier Master Revisions Form

From the Carrier Setup menu (G49414), select Work with Carriers (P4906), and then either click Add or select a carrier.

At the bottom of the form, enter the custom shipment tracking business function name in the Shipment Tracking Business Function field, as illustrated here:



Note: On Carrier Master Revisions, the Shipment Tracking Business Function field requires the name of the external business function, not the business function ID.

Parameter and Business Functions

B4900350 - CallExternalTrackingProgram is called from N4901340 - Track Shipment By Carrier. This function is the bridge to the custom shipment tracking function.

B4900470 - ShipmentTrackingTemplate is a template function that can be used to create the custom function and has the data structure (DSTR) the system expects to come from B4900350.

B4900360 - Track Fed Ex Shipment and B4900680 - Internet Shipment Tracking are additional sample functions that you may use as a model for creating custom shipment tracking functions.

Prebuilt Business Function Extension Point for Document Number Generation

Utilizing the document number generation functionality requires the implementation of custom business functions and therefore is considered a strictly custom process that is not supported by Oracle Support Services. This section provides a high-level overview of the external business function functionality.

Overview

The Transportation Management system enables users to generate custom document numbers for their Transportation Delivery Documents (Bill of Lading, Manifest, Master Bill of Lading, and so on).

You must create a custom function to generate the custom document numbers.

Setup Document Setup Revisions Form

From the Transportation Setup menu (G4941), select Work with Document Setup (P49190), and then either click Add or select a document.

At the bottom of the form, enter the document number generation function name in the Program ID External Doc # field, as illustrated here:

The screenshot displays the 'Work with Document Setup - Document Setup Revisions' window in Internet Explorer. The browser address bar shows the URL 'http://localhost:8888/jde/E1Menu.maf'. The Oracle logo and 'JD EDWARDS ENTERPRISEONE' are visible at the top. The user 'Jack B. Nimble [DV900CLM]' is logged in. The main form contains the following fields and options:

- Document Code: BOL1
- Document Type: DL (Delivery Ticket)
- Sequence Number: 2.00
- Program Name: R49110 (Demand Schedule Bill of lading)
- Version: ZJDE0002

Document Print Level

- Line Level
- Shipment Level
- Load Level

Primary Delivery Document Document Freight Update
 Primary Invoice Document Include Miscellaneous Lines
 Document Recreate

External Document Number

Reference Number Qualifier: []

Program ID External Doc #: SetExternalDocNbr [Search]

Note: On Document Setup Revisions, the Program ID External Doc# field requires the name of the external business function and not the business function ID. Use the Search feature to select the function name.

Parameter and Business Functions

B4900820 - CallExternalDocumentNbrProgram is called from N4900590 - Retrieve Document Next Number. This function is the bridge to the custom document number generation function.

N4900830 - SetExternalDocNbr is a template function that can be used to create the custom function and has the data structure (DSTR) the system expects to come from N4900590.

This appendix assumes an understanding of the JD Edwards EnterpriseOne design tools, Object Management Workbench (OMW), applications standards, business function coding standards, and usability standards.

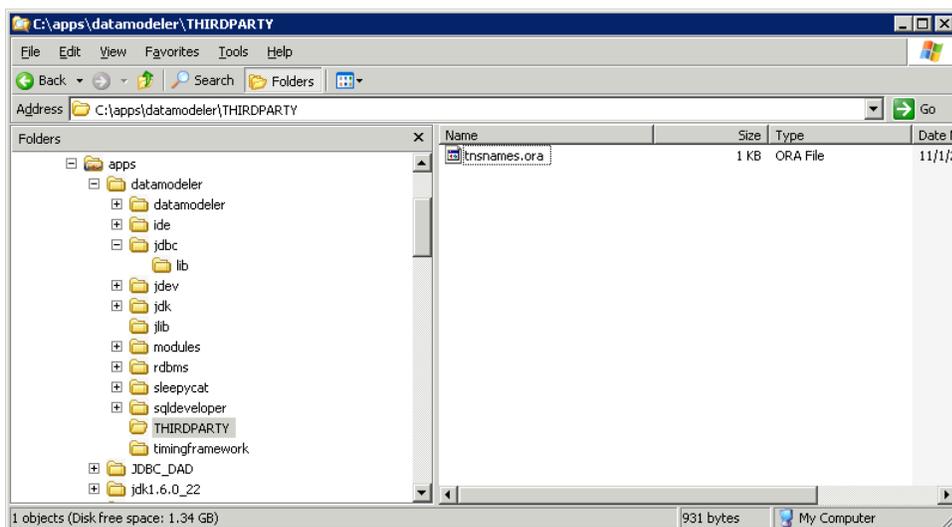
13 Appendix C - Oracle Data Modeler Setup

Oracle Data Modeler Setup

This appendix provides instructions for setting up the JD Edwards EnterpriseOne JDBC Driver for the Oracle SQL Developer Data Modeler.

Use these steps to setup Oracle Data Modeler:

1. Download the JD Edwards EnterpriseOne Data Access Driver (DAD) driver. The DAD driver is available for download from My Oracle Support | Patches & Updates | Software and Patch Search Sites.
 - a. Navigate to JDEdwards | Type: EnterpriseOne Tools Releases | Release: --All Releases-- | Platform: Multi-platform | Description (*text*): *Data Access Driver* | check the License Agreement check box | click the Search button.
 - b. Click the plus sign (+) to add the desired tools release specific driver to the Download Basket.
 - c. Click the Item(s) hyperlink, and download the tools release specific item.
2. Navigate to the Data Modeler install location, C:\apps\datamodeler and create a folder named THIRDPARTY, and copy the native database JDBC driver for your JD Edwards EnterpriseOne database (identified in the following list) into the THIRDPARTY folder.
 - o DB2/400 requires jt400.jar
 - o Microsoft SQL Server 2008 requires sqljdbc.jar
 - o Oracle database requires the tnsnames.ora file.



3. The JD Edwards EnterpriseOne Data Access Driver must be registered (installed) using Server Manager. The combined Server Manager Agent (installed on the Data Modeler server) and Server Manager Console are used to register the JD Edwards EnterpriseOne Data Access Driver. This process generates the jas.ini, jdbj.ini,

and `jdelog.properties` files and extracts the JD Edwards EnterpriseOne Data Access Driver jar files to a target location.

See the *Server Manager Guide* in the JD Edwards EnterpriseOne Installation and Upgrade for Apps 9.0 & Apps 9.1 library on Oracle Technology Network. <http://www.oracle.com/technetwork/documentation/jdedent-098169.html>

- a. From the Server Manager Console, select the Managed Home Location named for your Data Modeler Server. Click the Create New Managed Instance button. From the panel that follows, select the EnterpriseOne Data Access Driver option, and then click the Continue button.

- b. Complete the following form fields:

Management Dashboard | dendv003.mlab.jdedwards.com [C:\UDE_HOME]

Create/Register A Managed Instance

Shown below are all the instance properties that are required to create/register a new instance of the selected type. Complete the required fields and select 'Continue' to proceed to the next step. Unique instance names within this management domain may only contain the characters [a-zA-Z_0-9]; spaces or other special characters are not permitted.

Instance Type Instance Properties Confirmation Finish

Server Group: default

Instance Name: JDEJDBC_DAD

Usage Type: Generic

Install Location: C:\apps\JDBC_DAD

Software Component: EnterpriseOne Data Access Driver 9.1.0.0 10-26-2011_08_04

Server Group

Select the desired group, for example, default.

Instance Name

Provide the name of the instance you are creating.

Usage Type

Select Generic.

Install Location

Provide a valid path into which to install the JD Edwards Data Access Driver jar files.

Software Component

Select the desired tools release version, for example, 9.1.0.1.

- c. Click the Continue button.

Note: Click the Continue button only ONCE and wait for the panel to change.

- d. Complete the Create/Register A Managed Instance form fields based on your environment's configuration attributes, and then click the Continue button.

Create/Register A Managed Instance

Shown below are the configuration items that must be manually confirmed. Please validate or update, as appropriate, the configuration items. Once complete select 'Continue' to the installation/registration step.

Instance Type Instance Properties Confirmation Finish

Bootstrap User

Bootstrap User Password

Bootstrap Role

Bootstrap Environment

Primary Security Server

Outgoing JENET Port

Incoming JENET Port

System Datasource Name

Database Type

Database Name

Database Server Name

Database TCP/IP Port

Physical Database

Object Owner

Supports Large Objects (LOBs)

Unicode Database

e. On the next page, click the Create Instance button.

ORACLE JD Edwards EnterpriseOne

Server Manager Documentation
EnterpriseOne Documentation
Logout

Management Dashboard ▶ denv003.mlab.jdedwards.com [C:\IDE_HOME] ▶

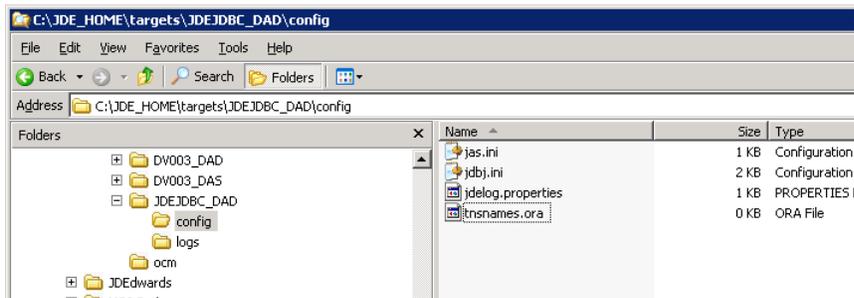
Create/Register A Managed Instance

Please wait while the managed instance is created/registered. Once complete you will be redirected to the management page for the newly created instance.

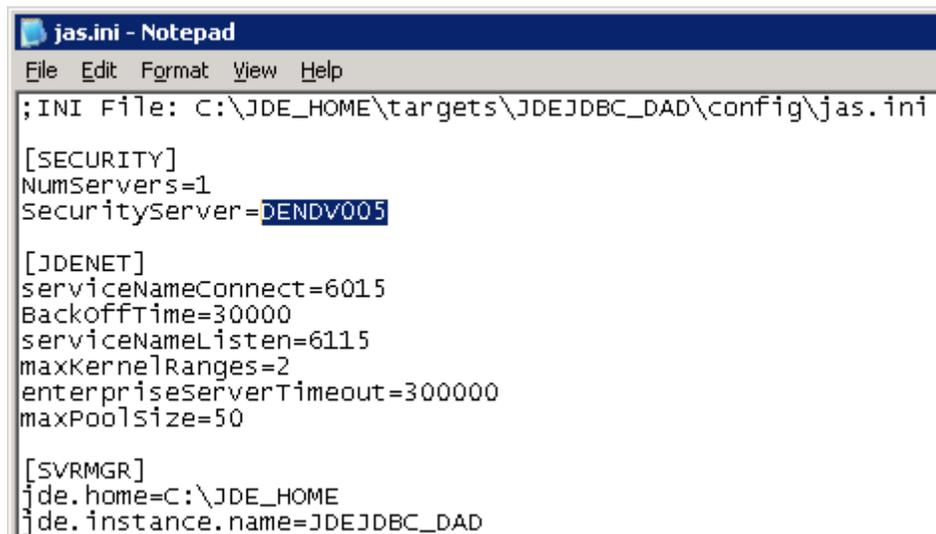
Instance Type Instance Properties Confirmation Finish

Select 'Create Instance' to finish the install of EnterpriseOne Data Access Driver. After installation is complete, distribute any needed jdbc drivers before using the Data Access Driver.

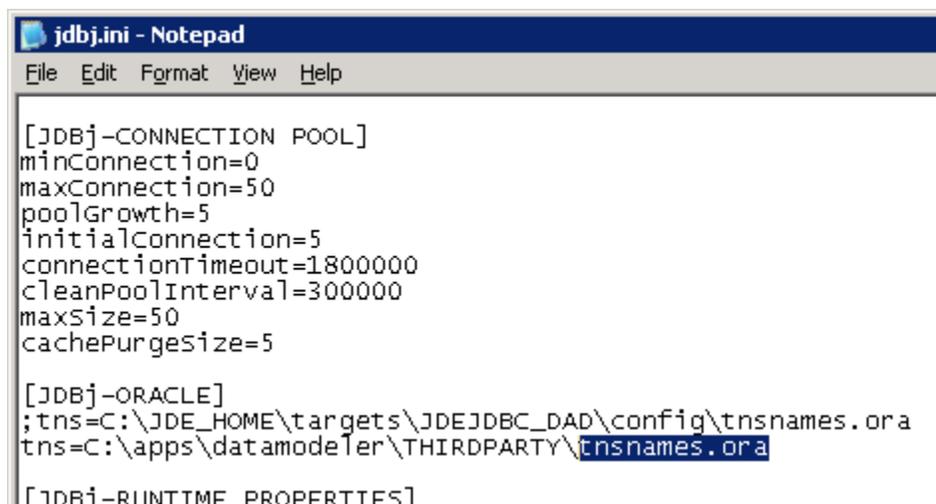
4. Navigate to the recently created Server Manager Agent JDBC driver configuration install location; for example, C:\JDE_HOME\targets\JDEJDBC_DAD\config, which is illustrated here:



- a. Ensure that the jas.ini file security server is correctly set.



- b. Edit the jdbj.ini file setting the location to the tnsnames.ora file.



- c. The password depicted in the following graphic is encrypted. This password can be set from the Server Manager Console only.

```

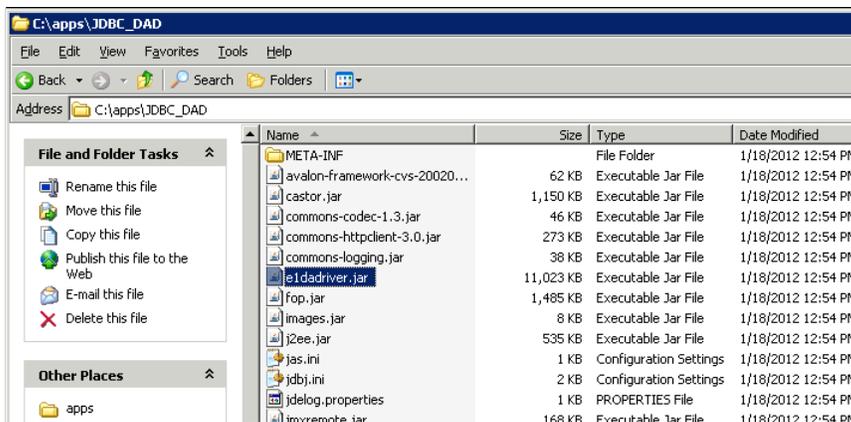
jdbj.ini - Notepad
File Edit Format View Help
;INI File: C:\JDE_HOME\targets\JDEJDBC_DAD\config\jdbj.ini

[JDBj-JDBC DRIVERS]
ORACLE=oracle.jdbc.driver.OracleDriver
AS400=com.ibm.as400.access.AS400JDBCdriver
SQLSERVER=com.microsoft.sqlserver.jdbc.SQLServerDriver
UDB=COM.ibm.db2.jdbc.app.DB2Driver

[JDBj-BOOTSTRAP SESSION]
role=*ALL
user=JDE
password=00hyhIKcvpu=
environment P0000
    
```

Note: It is imperative that the JDBJ.INI | [JDBj-BOOTSTRAP SESSION] stanza | password value be encrypted.

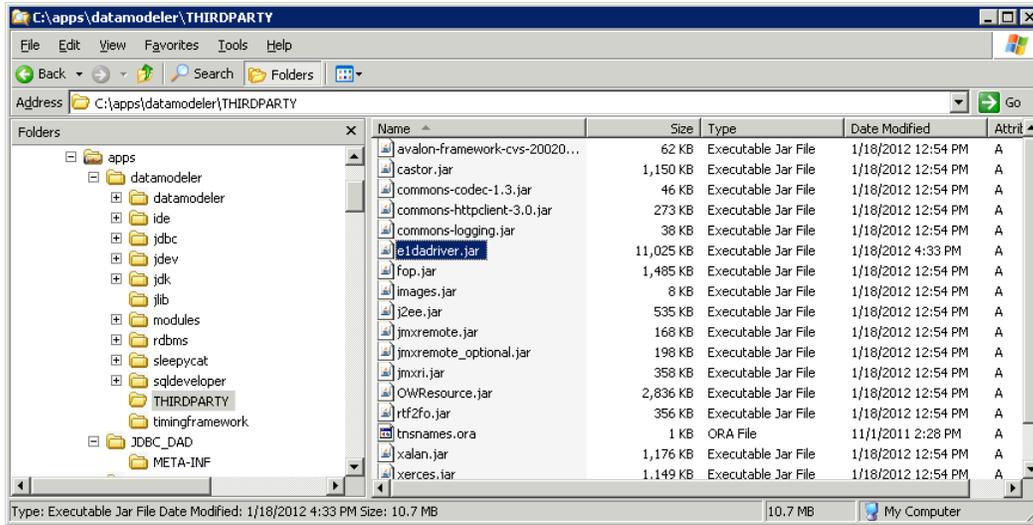
- d. Navigate to the JD Edwards EnterpriseOne Data Access Driver jar files install location. Add the jas.ini, jdbj.ini, and jdelog.properties found in the JDBC driver configuration install location (for example, C:\JDE_HOME\targets\JDEJDBC_DAD\config) to the e1dadriver.jar file.



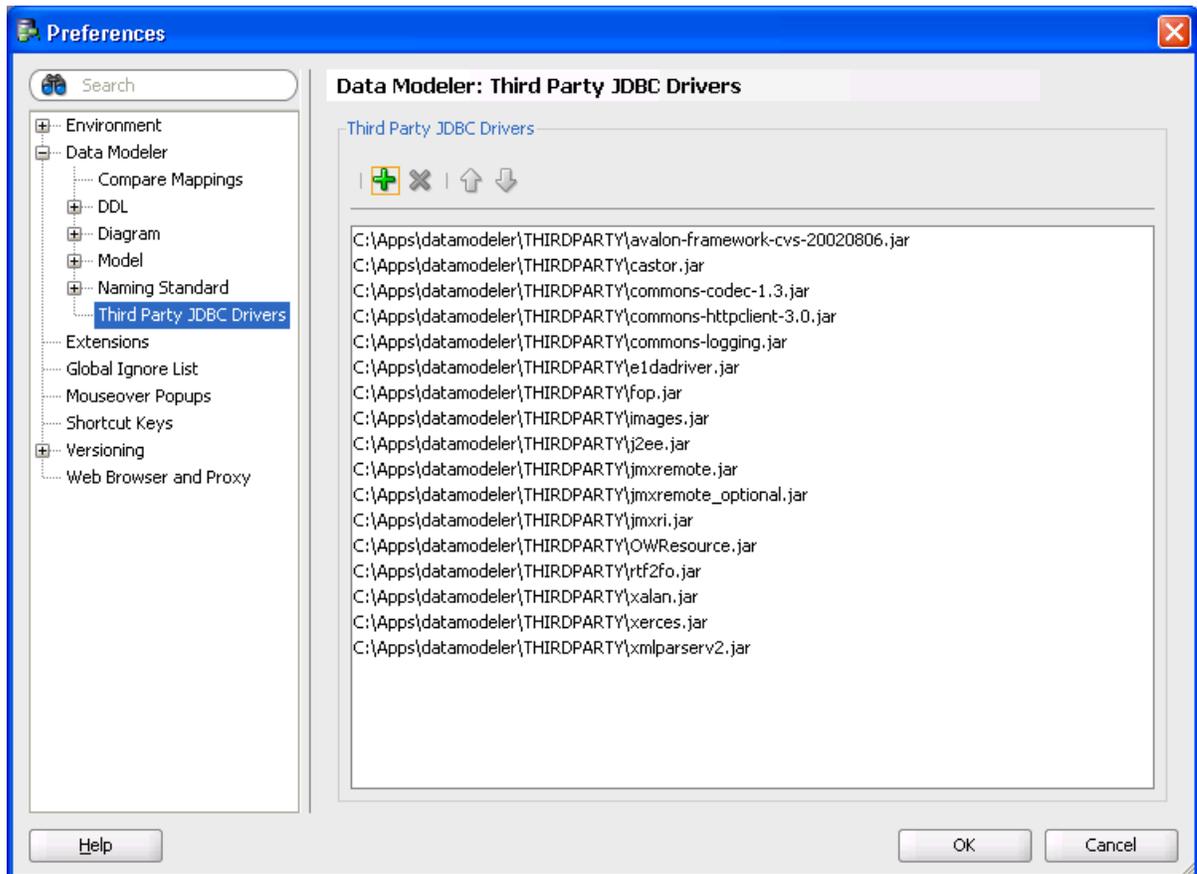
Adding the configuration files to the e1dadriver.jar file can be accomplished using an archive utility, or you can use the following steps.

- a. Unpack the e1dadriver.jar.
- b. Add the jas.ini, jdbj.ini, and jdelog.properties files to unpackaged e1dadriver.
- c. Pack the e1dadriver folder again to form e1dadriver.jar.

Copy ONLY the JAR files found in the JD Edwards EnterpriseOne Data Access Driver install location (for example, C:\apps\JDBC_DAD) into the Data Modeler third party folder (for example, C:\apps\datamodeler\THIRDPARTY).

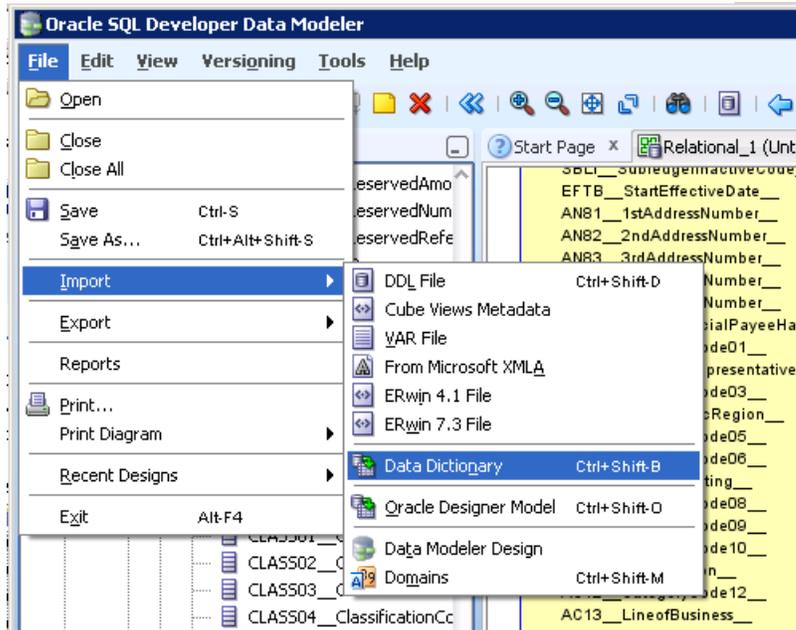


5. Configure the Data Modeler to use the JD Edwards EnterpriseOne Data Access Driver.
 - a. Start | Run | C:\apps\datamodeler\datamodeler.exe.
 - b. From the Menu bar select Tools | Preferences.
 - c. Expand the Data Modeler node and click Third Party JDBC Drivers.
 - d. Third Party JDBC Drivers are registered as shown in the following diagram.



If the e1dadriver.jar is not registered, add it by clicking the plus sign and navigating to the e1dadriver.jar file and other jar files.

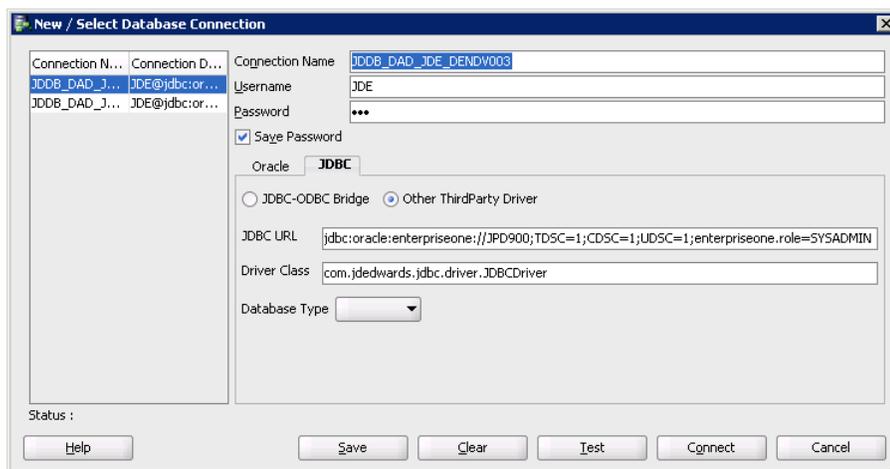
- e. Select File Menu | Import | Data Dictionary; and click the Add button to add connections.



6. Complete the fields on the New/Select Database Connection form, using one of the following:
 - o JDE JDBC Driver Definition *WITH* Column Descriptions and Specified EnterpriseOne Role (see following diagram and data entry descriptions)
 - o JDE JDBC Driver Definition *WITHOUT* Column Descriptions or a specified EnterpriseOne Role (see following diagram and data entry descriptions)

JDE JDBC Driver Definition *WITH* Column Descriptions and Specified EnterpriseOne Role

Complete the form fields:



- Connection Name: JDDB_DAD_JDE_DENDV003
- Username: JDE
- Password: JDE

Click the JDBC tab and complete these fields:

- Radio Button: Other ThirdParty Driver

- JDBC URL: jdbc:oracle:enterpriseone://JPD900;TDSC=1;CDSC=1;UDSC=1;enterpriseone.role =SYSADMIN
- Driver Class: com.jdedwards.jdbc.driver.JDBCdriver

The JDBC URL Connection String

Specifying the JD Edwards EnterpriseOne Role—You can add information that defines the JD Edwards EnterpriseOne role that will be used when connecting to the database to the end of the connection string; for example, enterpriseone.role=SYSADMIN. If the role is not specified in the connection string, the *ALL role is used.

Retrieving Table Descriptions—You can add information to the end of the connection string that enables the JDBC driver to retrieve table descriptions in addition to table names. To display table descriptions, add TDSC=1 to the connection string. If the value is 0 or the TDSC tag is not in the connection string, table descriptions will not be retrieved from the database.

Retrieving Column Descriptions—You can add information to the end of the connection string that enables the JDBC driver to retrieve column descriptions in addition to column names. The column description is the long column name from the data dictionary. Additionally, the column description is retrieved in the language of the user who is building the query in BI Publisher.

To display the column description, add CDSC=1 to the connection string. If the CDSC value is 0 or the CDSC tag is not in the connection string, column descriptions will not be retrieved from the database.

Retrieving UDC Descriptions—You can add information to the end of the connection string that enables the JDBC driver to retrieve the UDC description for table columns that have an associated UDC. Each table column is based on a data dictionary item, which could have a UDC assigned to it.

The UDC description is retrieved in the language of the user who is building the query in BI Publisher. Without the UDC description, the report developer must know which table columns have UDCs associated with them. The report developer can override the column name when designing the report.

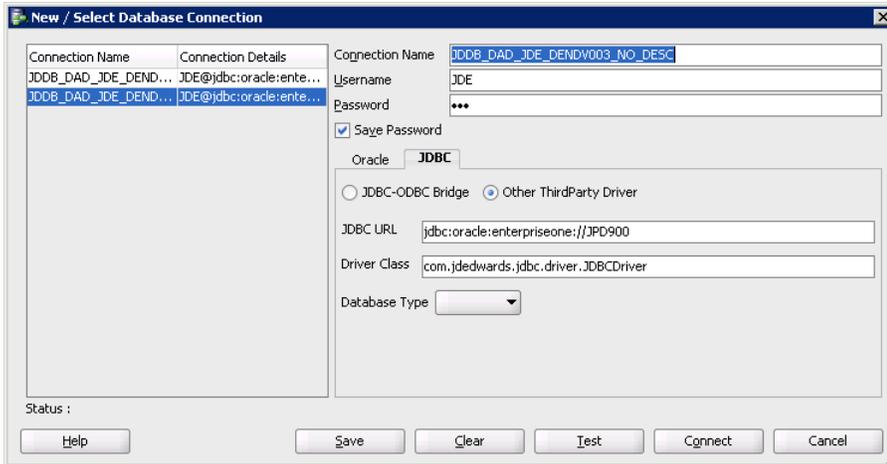
To display UDC descriptions, add UDSC=1 to the connection string. If the value is 0 or the UDSC tag is not in the connection string, UDC descriptions will not be retrieved from the database.

Note: Use a semi-colon between tags when appending them to the connection string. For example, the following connection string shows the appended information for retrieving the table description, column description, and UDC description:

```
jdbc:oracle:enterpriseone://JDV900;TDSC=1;CDSC=1;UDSC=1;
```

JDE JDBC Driver Definition ***WITHOUT*** Column Descriptions or a Specified EnterpriseOne Role

Complete the form fields:



- Connection Name: JDDB_DAD_JDE_DENDV003_NO_DESC
- Username: JDE
- Password: JDE

Click the JDBC tab and complete these fields:

- Radio Button: Other ThirdParty Driver
- JDBC URL: jdbc:oracle:enterpriseone://JPD900
- Driver Class: com.jdedwards.jdbc.driver.JDBCdriver

14 Appendix D - Creating an EnterpriseOne Page Example

Creating an EnterpriseOne Page Example

This appendix discusses how to create a new EnterpriseOne Page using JDeveloper. The example EnterpriseOne Page is from an Order Management use case.

Note: The Sample zip file, EnterpriseOne_Pages_SimpleOrderMgmt_Sample.zip, available on My Oracle Support (Document ID 1199953.1), has a link to a chart that contains links to files where you can cut and paste information that is contained in this appendix. <https://support.us.oracle.com/oip/faces/secure/km/DocumentDisplay.jspx?id=1199953.1>

Use these steps to create a new EnterpriseOne Page:

1. Open JDeveloper and create a new file called home.html or home.htm.
 - o This is the minimum requirement for an EnterpriseOne Page, the name is required.
 - o This step creates a basic html file with no content.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252"></meta>
<title>home</title>
</head>
<body></body>
</html>
```

- o Add the script element to include the e1pagehelper.js so that calls to runE1App and runE1UBE functions can be used. The e1pagehelper.js file is delivered with the EnterpriseOne HTML code base and is used when testing and running the content.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252"></meta>
<title>home</title>
<script src="../e1pagehelper.js" type="text/javascript"></script>
</head>
<body>
</body>
</html>
```

2. Create a folder named images in the same location where the html file exists.



Name ▲	Size	Type
 home.html	1 KB	Firefox Document
 images		File Folder

- Populate this images folder with the needed png, gif, or jpg (jpeg) files.

These files exist within the sample source zip file, EnterpriseOne_Pages_SimpleOrderMgmt_Sample.zip.

3. Go back to the home.html file and add a table using the design view.

- o Add content and styles to match this source code:

```
<table border="0" cellpadding="0" cellspacing="0" style="border-collapse: collapse" width="550">
  <tr>
    <td height="40" style="vertical-align:bottom;">
      <h2>Order Management</h2>
    </td>
    <td width="76px">
      </td>
    </tr>
  </table>
```

- o Add another table.
- o Add attributes and javascript calls to match this html code:

```
<table class="appLinks" border="0" cellpadding="0" cellspacing="0" style="border-collapse: collapse">
  <tr class="iconRow">
    <td class="appIcon">
      <a href=javascript:runElApp('P4210','W4210E','ZJDE0001')>
        </a>
      </td>
    <td class="spacer"></td>
    <td class="appIcon">
      <a href=javascript:runElApp('P4210','W4210A','ZJDE0001')>
        </a>
      </td>
    <td class="spacer"></td>
    <td class="appIcon">
      <a href=javascript:runElApp('P4310','W4310I','ZJDE0001')>
        
      </td>
    <td class="spacer"></td>
    <td class="appIcon">
      <a href=javascript:runElApp("P48013","W48013J","ZJDE0001")>
        
      </td>
    <td class="spacer"></td>
    <td class="appIcon">
      <a href=javascript:runElApp('P01012','W01012B','ZJDE0001')>
        
      </td>
  </tr>
  <tr class="linkRow">
    <td><a href=javascript:runElApp('P4210','W4210E','ZJDE0001')>Sales Order Home</a></td>
    <td></td><td><a href=javascript:runElApp('P4210','W4210A','ZJDE0001')>Enter Sales Order</a></td>
    <td></td><td><a href=javascript:runElApp('P4310','W4310I','ZJDE0001')>Enter Purchase Order</a></td>
    <td></td><td><a href=javascript:runElApp("P48013","W48013J","ZJDE0001")>Enter Work Order</a></td>
    <td></td><td><a href=javascript:runElApp('P01012','W01012B','ZJDE0001')>Address Book Home</a>
    </td><td></td></tr>
  <tr class="descRow">
    <td class="appDesc">View or change an existing Sales Order and submit it for approval</td>
    <td></td><td class="appDesc">Create a new Sales Order</td>
    <td></td><td class="appDesc">Create a new Purchase Order</td>
    <td></td><td class="appDesc">Create a new Work Order</td>
    <td></td><td class="appDesc">Select and print a hardcopy of an Expense Report</td>
    <td></td>
  </tr>
</table>
```

Notice we are using the exposed runElApp javascript function. This is using only the three required parameters.

- o Create two more tables to match the following html code:

```

<table border="0" cellpadding="0" cellspacing="0" style="border-collapse: collapse;margin-top:50px;" width="300">
  <tr>
    <td height="30">
      <h3 style="border-bottom-style: double; border-bottom-width: 1px">Navigation to Additional Tasks</h3>
    </td>
  </tr>
</table>
<table>
  <tr>
    <td class="navIcon">
      <a href="javascript:goToTabByLabel(new Object(), 'Customer Relationship Management')"></a>
    </td>
    <td class="navLink">
      <a href="javascript:goToTabByLabel(new Object(), 'Customer Relationship Management')">Customer Relationship Management</a>
    </td>
  </tr>
  <tr>
    <td class="navIcon">
      <a href="javascript:goToTabByLabel(new Object(), 'Sales Order Management')"></a>
    </td>
    <td class="navLink">
      <a href="javascript:goToTabByLabel(new Object(), 'Sales Order Management')">Sales Order Management</a>
    </td>
  </tr>
  <tr>
    <td class="navIcon">
      <a href="javascript:goToTabByLabel(new Object(), 'Human Capital Management')"></a>
    </td>
    <td class="navLink">
      <a href="javascript:goToTabByLabel(new Object(), 'Human Capital Management')">Human Capital Management</a>
    </td>
  </tr>
</table>

```

Notice the use of the goToTabByLabel javascript function. This call will take you to the Customer Relationship Management tab.
NOTE: For this to work the user MUST have that tab/content published to them

Note: These files exist within the sample source zip file, EnterpriseOne_Pages_SimpleOrderMgmt_Sample.zip.

- Review the page by open home.html file in a browser.

The coding of your html is now done.

Note: The javascript functionality cannot be tested at this time; to be tested, you must be within EnterpriseOne.

- Package the file into a zip or jar file.

This example uses the zip format called SimpleOrderMgmt.zip.

- It is important that all of the HTML content be relative to the root home.html (or home.htm) file. Because this zip or jar file will be deployed on the JD Edwards HTML Server at runtime, make sure that all resources that the html file requires are self-contained within the bundled zip or jar file.
- If there are folders (like images folder), they must be relative to the home.html or home.htm file.

Now you are ready to add the EnterpriseOne Page using the User Generated Content Manager (P982400) application, and then test within the context of EnterpriseOne.

- Access the User Generated Content Manager application and add a new EnterpriseOne Page. It is recommended to use the same name as the zip file.
 - Enter Object Name, Product Code, Page Title, and Page Description.
 - Upload the content from SimpleOrderMgmt.zip.
- Validate and test the new page.

8. Activate the content by taking the EnterpriseOne Page record through the status flow defined for the logged in user.
9. Publish the content to users and roles.

For more details, see "System Administrator and Designer Tasks for EnterpriseOne Pages" in the System Administration Guide in the JD Edwards EnterpriseOne Tools library on Oracle Technology Network. <http://www.oracle.com/technetwork/documentation/jdedent-098169.html>

Under the library, use the following key words to search: System Administration Guide or EnterpriseOne Pages.

