
JD Edwards EnterpriseOne Tools 8.97 Web Services Gateway: Integration Development Methodology Guide

October 2007

The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. This document is not warranted to be error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

If the Programs are delivered to the United States Government or anyone licensing or using the Programs on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS

Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are “commercial computer software” or “commercial technical data” pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software–Restricted Rights (June 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee’s responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and we disclaim liability for any damages caused by such use of the Programs.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

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

Open Source Disclosure

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

This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>). Copyright © 1999-2000 The Apache Software Foundation. All rights reserved. THIS SOFTWARE IS PROVIDED “AS IS” AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Contents

| | |
|--|-------------|
| General Preface | |
| About This Documentation Preface | ix |
| JD Edwards EnterpriseOne Application Prerequisites..... | ix |
| Application Fundamentals..... | ix |
| Documentation Updates and Printed Documentation..... | x |
| Obtaining Documentation Updates..... | x |
| Downloading Documentation..... | x |
| Additional Resources..... | x |
| Typographical Conventions and Visual Cues..... | xi |
| Typographical Conventions..... | xii |
| Visual Cues..... | xii |
| Country, Region, and Industry Identifiers..... | xiii |
| Currency Codes..... | xiv |
| Comments and Suggestions..... | xiv |
| Common Fields Used in Implementation Guides..... | xiv |
| Preface | |
| JD Edwards EnterpriseOne Tools Web Services Gateway Integration Development | |
| Methodology Preface..... | xvii |
| JD Edwards WSG Integration Development Methodology Companion Documentation..... | xvii |
| Chapter 1 | |
| Getting Started with JD Edwards EnterpriseOne Tools Web Services Gateway | |
| Integration Development Methodology..... | 1 |
| JD Edwards EnterpriseOne WSG Integration Development Methodology Overview..... | 1 |
| JD Edwards EnterpriseOne WSG Integration Development Methodology Integrations..... | 1 |
| JD Edwards EnterpriseOne WSG Integration Development Methodology Implementation..... | 2 |
| JD Edwards EnterpriseOne WSG Integration Development Methodology Implementation | |
| Steps..... | 2 |
| Chapter 2 | |
| Managing Integration Server Components..... | 3 |
| Understanding Naming Standards for Integration Server Components..... | 3 |

| | |
|---------------------------------------|----|
| Name Lengths..... | 3 |
| Packages..... | 4 |
| Deployment Packages..... | 5 |
| Folders..... | 5 |
| Interface Documents..... | 7 |
| Adapter Services..... | 11 |
| Common Services..... | 12 |
| Transformation Services..... | 13 |
| Triggers..... | 18 |
| Integration Flow Services..... | 19 |
| Editing XML Namespaces..... | 20 |
| Converting XBPs to Flow Services..... | 21 |

Chapter 3

| | |
|---|-----------|
| Versioning Integrations..... | 23 |
| Versioning Interface Documents..... | 23 |
| Versioning Flow Services, Adapter Services, and Utility Services..... | 24 |
| Deprecating Artifacts..... | 24 |
| Modifying Delivered Integrations..... | 25 |

Chapter 4

| | |
|--|-----------|
| Understanding Development Conventions..... | 27 |
| Coding Standards..... | 27 |
| Integration Server Development Conventions..... | 27 |
| Error Handling..... | 28 |
| Transactions and Flow Service Design..... | 32 |
| Invoking Flow Services with Explicit Transaction Boundaries..... | 34 |
| Transactions and Stepping/Tracing in Developer..... | 35 |
| Cross-Referencing..... | 36 |
| Integration Options..... | 36 |
| Documentation..... | 37 |
| Testing..... | 37 |
| Test Script Development and Maintenance..... | 37 |
| Guidelines for Testing..... | 38 |

Appendix A

| | |
|----------------------------|-----------|
| Source Control..... | 39 |
|----------------------------|-----------|

| | |
|---|----|
| Clear Case Structure..... | 39 |
| Views..... | 40 |
| WSG EnterpriseOne Adapter Services Package Structure..... | 40 |
| Manifest Files..... | 40 |
| Testing Scripts..... | 41 |

Appendix B

| | |
|-----------------------|-----------|
| Family..... | 43 |
| Defined Families..... | 43 |

Appendix C

| | |
|---|-----------|
| PSFT_XrefAndSoftCoding Services..... | 45 |
| PSFT_XRefAndSoftCoding.XRef:createCodeXReference..... | 45 |
| PSFT_XRefAndSoftCoding.XRef:createKeyXReference..... | 45 |
| PSFT_XRefAndSoftCoding.XRef:deleteCodeXReference..... | 46 |
| PSFT_XRefAndSoftCoding.XRef:deleteKeyXReference..... | 47 |
| PSFT_XRefAndSoftCoding.XRef:getCanonicalCode..... | 47 |
| PSFT_XRefAndSoftCoding.XRef:getCanonicalKey..... | 48 |
| PSFT_XRefAndSoftCoding.XRef:getCodeLatch..... | 49 |
| PSFT_XRefAndSoftCoding.XRef:getKeyLatch..... | 49 |
| PSFT_XRefAndSoftCoding.XRef:getNativeCode..... | 50 |
| PSFT_XRefAndSoftCoding.XRef:getNativeKey..... | 51 |
| PSFT_XRefAndSoftCoding.XRef:setCodeLatch..... | 51 |
| PSFT_XRefAndSoftCoding.XRef:setKeyLatch..... | 52 |
| PSFT_XRefAndSoftCoding.SoftCoding:getIntegrationOption..... | 53 |
| PSFT_XRefAndSoftCoding.SoftCoding:getOptionalIntegrationOption..... | 53 |
| PSFT_XRefAndSoftCoding.Utills:concatErrorMessage..... | 54 |
| PSFT_XRefAndSoftCoding.Utills:createKeyDebugMessage..... | 55 |
| PSFT_XRefAndSoftCoding.Utills:generateCanonicalID..... | 55 |
| PSFT_XRefAndSoftCoding.Utills:getBrokerSettings..... | 56 |

Appendix D

| | |
|--|-----------|
| PSFT_Utills Services..... | 57 |
| PSFT_Utills.Conversion:booleanToString..... | 57 |
| PSFT_Utills.Conversion:charToString..... | 57 |
| PSFT_Utills.Conversion:dateToString..... | 58 |
| PSFT_Utills.Conversion:dateToStringWithOffset..... | 58 |

| | |
|--|----|
| PSFT_Utills.Conversion:doubleToLong..... | 59 |
| PSFT_Utills.Conversion:doubleToString..... | 59 |
| PSFT_Utills.Conversion:doubleToStringNoDecimal..... | 60 |
| PSFT_Utills.Conversion:intToString..... | 60 |
| PSFT_Utills.Conversion:longToString..... | 61 |
| PSFT_Utills.Conversion:stringToBoolean..... | 61 |
| PSFT_Utills.Conversion:stringToChar..... | 62 |
| PSFT_Utills.Conversion:stringToDate..... | 62 |
| PSFT_Utills.Conversion:stringToDouble..... | 63 |
| PSFT_Utills.Conversion:stringToInt..... | 63 |
| PSFT_Utills.Conversion:stringToLong..... | 64 |
| PSFT_Utills.Error:customizedHandleError..... | 64 |
| PSFT_Utills.Error:getErrorListForE1..... | 65 |
| PSFT_Utills.Error:getErrorListForERP8..... | 65 |
| PSFT_Utills.Error:handleError..... | 65 |
| PSFT_Utills.Error:testErrorHandler..... | 67 |
| PSFT_Utills.File.Utills:closeFileWriter..... | 68 |
| PSFT_Utills.File.Utills:deleteFile..... | 68 |
| PSFT_Utills.File.Utills:doesFileExist..... | 69 |
| PSFT_Utills.File.Utills:fileType..... | 69 |
| PSFT_Utills.File.Utills:flushFileWriter..... | 70 |
| PSFT_Utills.File.Utills:getFileSeparator..... | 70 |
| PSFT_Utills.File.Utills:getLineSeparator..... | 71 |
| PSFT_Utills.File.Utills:openFileWriter..... | 71 |
| PSFT_Utills.File.Utills:renameFile..... | 72 |
| PSFT_Utills.File.Utills:writeFileWriter..... | 72 |
| PSFT_Utills.File:docToFileWithDate..... | 73 |
| PSFT_Utills.File:stringToFileWithDate..... | 74 |
| PSFT_Utills.File:writeToFile..... | 74 |
| PSFT_Utills.FlatFiles.FormatServices:formatAndTrimInStrings..... | 75 |
| PSFT_Utills.FlatFiles.FormatServices:formatInStrings..... | 76 |
| PSFT_Utills.FlatFiles.FormatServices:leftPadOutNumWithBlanks..... | 76 |
| PSFT_Utills.FlatFiles.FormatServices:trimAndReturnNullIfEmpty..... | 77 |
| PSFT_Utills.FlatFiles.Utills:getNullValue..... | 77 |
| PSFT_Utills.Flow:dynamicServiceInvocation..... | 78 |
| PSFT_Utills.Format:formatBU..... | 78 |
| PSFT_Utills.Format:trimLength..... | 79 |
| PSFT_Utills.Integrations:convert_ActionType_To_AUDIT_ACTN..... | 79 |
| PSFT_Utills.Integrations:convert_AUDIT_ACTN_To_ActionType..... | 80 |
| PSFT_Utills.Math:addInts..... | 80 |

PSFT_Utils.Time:AddTimes.....81

PSFT_Utils.Time:CompareTimes.....81

PSFT_Utils.Time:GetCurrentTime.....82

Glossary of JD Edwards EnterpriseOne Terms.....83

Index99

About This Documentation Preface

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

This preface discusses:

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

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

JD Edwards EnterpriseOne Application Prerequisites

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

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

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

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

Application Fundamentals

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

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

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

Documentation Updates and Printed Documentation

This section discusses how to:

- Obtain documentation updates.
- Download documentation.

Obtaining Documentation Updates

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

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

See Also

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

Downloading Documentation

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

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

Additional Resources

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

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

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

Typographical Conventions and Visual Cues

This section discusses:

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

Typographical Conventions

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

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

Visual Cues

Implementation guides contain the following visual cues.

Notes

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

Note. Example of a note.

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

Important! Example of an important note.

Warnings

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

Warning! Example of a warning.

Cross-References

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

Country, Region, and Industry Identifiers

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

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

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

Country Identifiers

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

Region Identifiers

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

- Asia Pacific
- Europe
- Latin America
- North America

Industry Identifiers

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

- USF (U.S. Federal)

- E&G (Education and Government)

Currency Codes

Monetary amounts are identified by the ISO currency code.

Comments and Suggestions

Your comments and suggestions are important to us. We encourage you to send us your feedback about our PeopleBooks and other reference and training materials. Please include the release numbers for the PeopleTools and applications that you are currently using. Email your comments to PSOFT-INFODEV_US@ORACLE.COM.

Common Fields Used in Implementation Guides

| | |
|----------------------------|--|
| Address Book Number | Enter a unique number that identifies the master record for the entity. An address book number can be the identifier for a customer, supplier, company, employee, applicant, participant, tenant, location, and so on. Depending on the application, the field on the form might refer to the address book number as the customer number, supplier number, or company number, employee or applicant ID, participant number, and so on. |
| As If Currency Code | Enter the three-character code to specify the currency that you want to use to view transaction amounts. This code enables you to view the transaction amounts as if they were entered in the specified currency rather than the foreign or domestic currency that was used when the transaction was originally entered. |
| Batch Number | Displays a number that identifies a group of transactions to be processed by the system. On entry forms, you can assign the batch number or the system can assign it through the Next Numbers program (P0002). |
| Batch Date | Enter the date in which a batch is created. If you leave this field blank, the system supplies the system date as the batch date. |
| Batch Status | <p>Displays a code from user-defined code (UDC) table 98/IC that indicates the posting status of a batch. Values are:</p> <p><i>Blank</i>: Batch is unposted and pending approval.</p> <p><i>A</i>: The batch is approved for posting, has no errors and is in balance, but has not yet been posted.</p> <p><i>D</i>: The batch posted successfully.</p> <p><i>E</i>: The batch is in error. You must correct the batch before it can post.</p> <p><i>P</i>: The system is in the process of posting the batch. The batch is unavailable until the posting process is complete. If errors occur during the post, the batch status changes to <i>E</i>.</p> |

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

| | |
|-------------------------|---|
| Branch/Plant | Enter a code that identifies a separate entity as a warehouse location, job, project, work center, branch, or plant in which distribution and manufacturing activities occur. In some systems, this is called a business unit. |
| Business Unit | Enter the alphanumeric code that identifies a separate entity within a business for which you want to track costs. In some systems, this is called a branch/plant. |
| Category Code | Enter the code that represents a specific category code. Category codes are user-defined codes that you customize to handle the tracking and reporting requirements of your organization. |
| Company | Enter a code that identifies a specific organization, fund, or other reporting entity. The company code must already exist in the F0010 table and must identify a reporting entity that has a complete balance sheet. |
| Currency Code | Enter the three-character code that represents the currency of the transaction. JD Edwards EnterpriseOne provides currency codes that are recognized by the International Organization for Standardization (ISO). The system stores currency codes in the F0013 table. |
| Document Company | <p>Enter the company number associated with the document. This number, used in conjunction with the document number, document type, and general ledger date, uniquely identifies an original document.</p> <p>If you assign next numbers by company and fiscal year, the system uses the document company to retrieve the correct next number for that company.</p> <p>If two or more original documents have the same document number and document type, you can use the document company to display the document that you want.</p> |
| Document Number | Displays a number that identifies the original document, which can be a voucher, invoice, journal entry, or time sheet, and so on. On entry forms, you can assign the original document number or the system can assign it through the Next Numbers program. |
| Document Type | <p>Enter the two-character UDC, from UDC table 00/DT, that identifies the origin and purpose of the transaction, such as a voucher, invoice, journal entry, or time sheet. JD Edwards EnterpriseOne reserves these prefixes for the document types indicated:</p> <p><i>P</i>: Accounts payable documents.</p> <p><i>R</i>: Accounts receivable documents.</p> <p><i>T</i>: Time and pay documents.</p> <p><i>I</i>: Inventory documents.</p> <p><i>O</i>: Purchase order documents.</p> <p><i>S</i>: Sales order documents.</p> |
| Effective Date | Enter the date on which an address, item, transaction, or record becomes active. The meaning of this field differs, depending on the program. For example, the effective date can represent any of these dates: |

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

Fiscal Period and Fiscal Year

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

G/L Date (general ledger date)

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

JD Edwards EnterpriseOne Tools Web Services Gateway Integration Development Methodology Preface

This preface discusses Web Services Gateway (WSG) integration development methodology companion documentation.

JD Edwards WSG Integration Development Methodology Companion Documentation

Additional, essential information describing the setup and design of JD Edwards EnterpriseOne Tools resides in companion documentation. The companion documentation consists of important topics that apply to JD Edwards EnterpriseOne WSG integration development methodology as well as other JD Edwards EnterpriseOne Tools. You should be familiar with the contents of these companion guides:

- JD Edwards EnterpriseOne Tools 8.97 Web Services Gateway: Order Promising Adapter Programmer's Guide
- JD Edwards EnterpriseOne Tools 8.97 Web Services Gateway: EnterpriseOne Adapter Programmer's Guide
- JD Edwards EnterpriseOne Tools 8.97 Web Services Gateway: Configuration Editor Guide
- JD Edwards EnterpriseOne Tools 8.97 Web Services Gateway: Dispatcher Guide

See Also

JD Edwards EnterpriseOne Tools 8.97 Web Services Gateway: Order Promising Adapter Programmer's Guide, "Getting Started with JD Edwards EnterpriseOne Tools Web Services Gateway Order Promising Adapter"

JD Edwards EnterpriseOne Tools 8.97 Web Services Gateway: EnterpriseOne Adapter Programmer's Guide, "Getting Started with JD Edwards EnterpriseOne Tools Web Services Gateway EnterpriseOne Adapter"

JD Edwards EnterpriseOne Tools 8.97 Web Services Gateway: Configuration Editor Guide, "Getting Started with JD Edwards EnterpriseOne Tools Web Services Gateway Configuration Editor"

JD Edwards EnterpriseOne Tools 8.97 Web Services Gateway: Dispatcher Guide, "Getting Started with JD Edwards EnterpriseOne Tools Web Services Gateway Dispatcher"

CHAPTER 1

Getting Started with JD Edwards EnterpriseOne Tools Web Services Gateway Integration Development Methodology

This chapter discusses:

- JD Edwards EnterpriseOne Web Services Gateway (WSG) Integration Development Methodology overview
- JD Edwards EnterpriseOne Integration Development Methodology integrations
- JD Edwards EnterpriseOne Integration Development Methodology implementation

JD Edwards EnterpriseOne WSG Integration Development Methodology Overview

Oracle's JD Edwards EnterpriseOne WSG Integration Development Methodology describes a set of practices and steps to follow to create integrations using the WSG toolset. This document does not preclude the use of other standard development methodologies.

JD Edwards EnterpriseOne WSG Integration Development Methodology Integrations

JD Edwards EnterpriseOne WSG Integration Development Methodology works with other JD Edwards EnterpriseOne tools to ensure that all information is fully integrated. JD Edwards EnterpriseOne WSG Integration Development Methodology integrates with these JD Edwards EnterpriseOne tools from Oracle:

- JD Edwards EnterpriseOne Web Services Gateway Foundation
- JD Edwards EnterpriseOne Web Services Gateway Developer
- JD Edwards EnterpriseOne Web Services Gateway Adapters
- JD Edwards EnterpriseOne Web Services Gateway Integration Server

JD Edwards EnterpriseOne WSG Integration Development Methodology Implementation

This section provides an overview of the steps that are required to implement JD Edwards EnterpriseOne WSG Integration Development Methodology.

In the planning phase of your implementation, take advantage of all JD Edwards EnterpriseOne sources of information, including the installation guides and troubleshooting information. A complete list of these resources appears in the preface in *About This Documentation* with information about where to find the most current version of each.

JD Edwards EnterpriseOne WSG Integration Development Methodology Implementation Steps

This table lists the steps for the JD Edwards EnterpriseOne WSG Integration Development Methodology implementation.

| Step | Reference |
|---|--|
| 1. Install JD Edwards EnterpriseOne Web Services Gateway. | <i>JD Edwards EnterpriseOne 8.97 Web Services Gateway Installation and Setup Guide</i> |
| 2. For Integrating PeopleSoft Enterprise and JD Edwards EnterpriseOne systems, create a JD Edwards EnterpriseOne node on PeopleSoft Enterprise Integration Gateway. | <i>JD Edwards EnterpriseOne Tools 8.97 Web Services Gateway: Dispatcher Guide</i> , “Using the Dispatcher,” Configuring the Inbound Dispatcher Service |
| 3. Set up integration options using the JD Edwards EnterpriseOne Web Services Gateway Configuration Editor. | <i>JD Edwards EnterpriseOne Tools 8.97 Web Services Gateway: Configuration Editor Guide</i> , “Using the Configuration Editor,” Using Integration Options |
| 4. Verify that the JDBC adapter is connected to the Integrations Options table. | <i>JD Edwards EnterpriseOne Tools 8.97 Web Services Gateway: Configuration Editor Guide</i> , “Using the Configuration Editor,” Accessing the Configuration Editor |

CHAPTER 2

Managing Integration Server Components

This chapter provides an overview of naming standards for integration server components and discusses how to:

- Edit XML namespaces.
- Convert XBPs to flow services.

Understanding Naming Standards for Integration Server Components

JD Edwards EnterpriseOne WSG uses the Integration Server to integrate data from a variety of systems. As with any production systems environment, consistency of implementation from one integration project to the next is critical to ensure the evolution of a maintainable WSG environment.

This section provides an overview of name lengths and discusses naming standards for these Integration Server elements:

- Packages
- Deployment packages
- Folders
- Interface documents
- Adapter services
- Common services
- Transformation services
- Triggers
- Integration flow services

Name Lengths

The size of a path is limited in Microsoft Windows. The path cannot exceed a length of 248 characters. When you use C:\Program Files\webMethods61\IntegrationServer\packages\ as a constant value for all integrations, the length of the name of services, triggers, and so on, including the package and folders names and \s, cannot exceed 192 characters. This example uses the webMethods default install location:

```
C:\ProgramFiles\webMethods61\IntegrationServer\packages\  
PSFT_EnterpriseOne_AdapterServices\ns\AdapterServices\Busness_Function\  
Financials\getNextNumber
```

This path has a total of 152 characters.

Note. The webMethods install location is longer than the default PeopleSoft\wsg location.

Please keep this limitation in mind when determining the names of integration server elements.

Packages

A package is a logical container for a set of services and related files. A package exists as a single physical directory on the file system.

Use this format when naming the package:

```
<provider prefix>|<resource identifier>|<group name>
```

This table describes the elements of package names:

| Element | Description |
|-----------------------|--|
| <provider prefix> | Identifies the provider for the services. Current prefixes are: <ul style="list-style-type: none"> • PSFT: PeopleSoft • wm: webMethods |
| <resource identifier> | Identifies the resource providing the services. Current resource identifiers are: <ul style="list-style-type: none"> • EnterpriseOne: EnterpriseOne • Enterprise: Enterprise • Blank: not specific to a resource |
| <group name> | Identifies the functional area for the service(s). Examples of current group names are: <ul style="list-style-type: none"> • Utils: Common utilities • Family: Address Book, Sales, Procurement, and so on See Appendix B, “Family,” page 43. • Integration roadmap name: SCM, CRM and so on • AdapterNameServices: EnterpriseOne_AdapterServices, JDBC_AdapterServices |

Examples

These are examples of package names:

- PSFT_EnterpriseOne_SCM
- PSFT_Enterprise_SCM
- PSFT_EnterpriseOne_AdapterServices

Deployment Packages

A deployment package is the Zip file that is created for deployment on the Integration Server.

Use this format when naming the deployment package:

```
<package name>_<update #>
```

This table describes the elements of deployment package names:

| Element | Description |
|----------------|---|
| <package name> | Identifies the package being deployed. |
| <update #> | Identifies the update number for delivering fixes. The initial package does not contain an update number. |

Examples

These are examples of deployment package names:

- PSFT_EnterpriseOne_SCM
- PSFT_EnterpriseOne_SCM_SP1

Folders

This section discusses the naming conventions for:

- Primary folders
- Other folders
- Adapter services packages

Primary Folders

The primary function of a folder is to organize and house related services and files.

To avoid naming conflicts, the primary folder is named the same as the package name. Under the primary folder is a folder named the same as the group name. If you want to group further, you may have another folder with a subgroup name. Generally, use initial caps (the first letter is capitalized, the remaining letters are lower case) for each word in the group or subgroup name. If the subgroup is an integration, follow the naming standards for the integration name to name the folder.

These are examples of folders and subfolders:

```
PSFT_EnterpriseOne_SCM
  PSFT_EnterpriseOne_SCM
    Inventory
PSFT_Utils
  PSFT_Utils
    Conversion
PSFT_EnterpriseOne_HCM
  PSFT_EnterpriseOne_HCM
    Employee
    Payroll
```

```
El_Company_To_E_BusinessUnit
```

Other Folders

All other folders use this format:

```
<Artifact Type>
```

This table describes the elements of folder names:

| Element | Description |
|-----------------|--|
| <Artifact Type> | <p>Identifies the type of object that is stored in the folder.</p> <p>The folder name is lower case.</p> <p>Current values are:</p> <ul style="list-style-type: none"> utils: common services for the package maps: transformation services docs: interface documents (includes schemas, data dictionary) triggers: triggers |

Note. The utils and docs folders can reside at both the package and the grouping level. The docs folder at the package level contains interface documents, and the docs folder at the grouping level contains internal documents. The utils folder at the package level contains common services for that package. The utils folder at the grouping level contains common services for the group.

Maps and triggers are associated with the integration and reside in the integration subgroup folder.

This is an example of a package with docs and utils folders:

```
PSFT_EnterpriseOne_HCM
  PSFT_EnterpriseOne_HCM
    docs
    utils
  Employee
  Payroll
    docs
    utils
  El_Company_To_E_BusinessUnit
    maps
    triggers
```

Adapter Services Package

The adapter services package does not have a primary folder. This folder is removed to enable the JD Edwards ERP 8.0 system, the JD Edwards OneWorld Xe system, and the JD Edwards EnterpriseOne system to use the same flow services. The first folder in the adapter services packages is AdapterServices. Removing the uniqueness of the primary folder enables you to enable only one of the PSFT_EnterpriseOne adapter services packages at a time.

These are examples of adapter services packages:

```

PSFT_EnterpriseOne_AdapterServices
  AdapterServices
    BusinessFunction
      [Family]
    Database
      [Family]
    Notification
      [Family]
  EnterpriseOne_AdapterServices_Listener
    EnterpriseOne_Listener
PSFT_ERP8_XE_AdapterServices
  AdapterServices
    BusinessFunction
      [Family]
    Database
      [Family]
    Notification
      [Family]
  ERP8_XE_AdapterServices_Listener
    ERP_XE_Listener

```

See [Appendix B, “Family,” page 43](#).

Interface Documents

Interface documents provide the input and output to the flow services.

Naming Conventions

Interface documents follow a verb and noun naming convention.

Structure

Documents and document lists are collections of related fields. Any grouping within the interface document is defined as a document or document list. This is an example of a document and a document list:

- Header - document
- Line - document list

Fields are the lowest level elements that are defined. Fields are fundamental elements that are used to create documents and document lists (for example, description, name, and so on). You must use data types that are supported universally when creating interface messages. Avoid using complex data types when possible and always use arrays instead of collection classes.

The interface document is used as the input and output for the web service generated from the flow. The webMethods Developer requires that all fields defined in the document are a string type. When defining the interface document, define the field as a string and then update the properties to set the constraint to the correct XML schema data type. When the WSDL is generated, the XML references the correct data type so the consumer knows what data type is required for that field in the interface.

All fields in an interface document must have a value set for the XML namespace property. For all fields, the XML namespace value is <http://www.peoplesoft.com/>.

See [Chapter 2, “Managing Integration Server Components,” Editing XML Namespaces, page 20](#).

This table lists valid XML schema data types to use when setting the constraint for the string field in the interface document:

| XML Schema Data Type | Usage |
|--|--|
| boolean(http://www.w3.org/2001/XMLSchema) | Use for Boolean (true or false, Y or N, 1 or 0) values. |
| decimal(http://www.w3.org/2001/XMLSchema) | Use for currency values and any other fixed precision/scale decimal data. |
| date(http://www.w3.org/2001/XMLSchema) dateTime(http://www.w3.org/2001/XMLSchema) | Use for time, date, and timestamp data. |
| integer(http://www.w3.org/2001/XMLSchema) | Use for integral data. |
| long(http://www.w3.org/2001/XMLSchema) | Use for integral data that spans a range larger than an int can represent. |
| string(http://www.w3.org/2001/XMLSchema) | Use for string and character data. |

These data types are *not* allowed for defining fields:

| Glossary | Description |
|----------|---|
| Vector | Does not work well when used as part of a Web Service because the World Wide Web Consortium (W3C) has not defined a vector data type in its schema. Web Services Interoperability Organization (WS-I) advises against using any collection data types that are not supported by the W3C schema. |
| char | Not acceptable for Web Services as determined by the Web Service Interoperability Organization (WS-I). Not all Web Services vendors support the use of the char data type. |

Interface Policies

You can create these interfaces:

- **Public:** An immutable interface exposed by a component.
- **Published:** A public interface that is registered with an interface registry such that it can be discovered by other systems.
- **Component Scoped:** An interface that is not exposed and is internal to the component.

This table lists the policies that govern the different types of interfaces:

| Policy | Description |
|--------|--|
| Public | Once a public interface is delivered, modifications to the interface are not allowed. Any changes to a public interface require that a new interface be provided. The existing interface may be deprecated if it will no longer be used. |

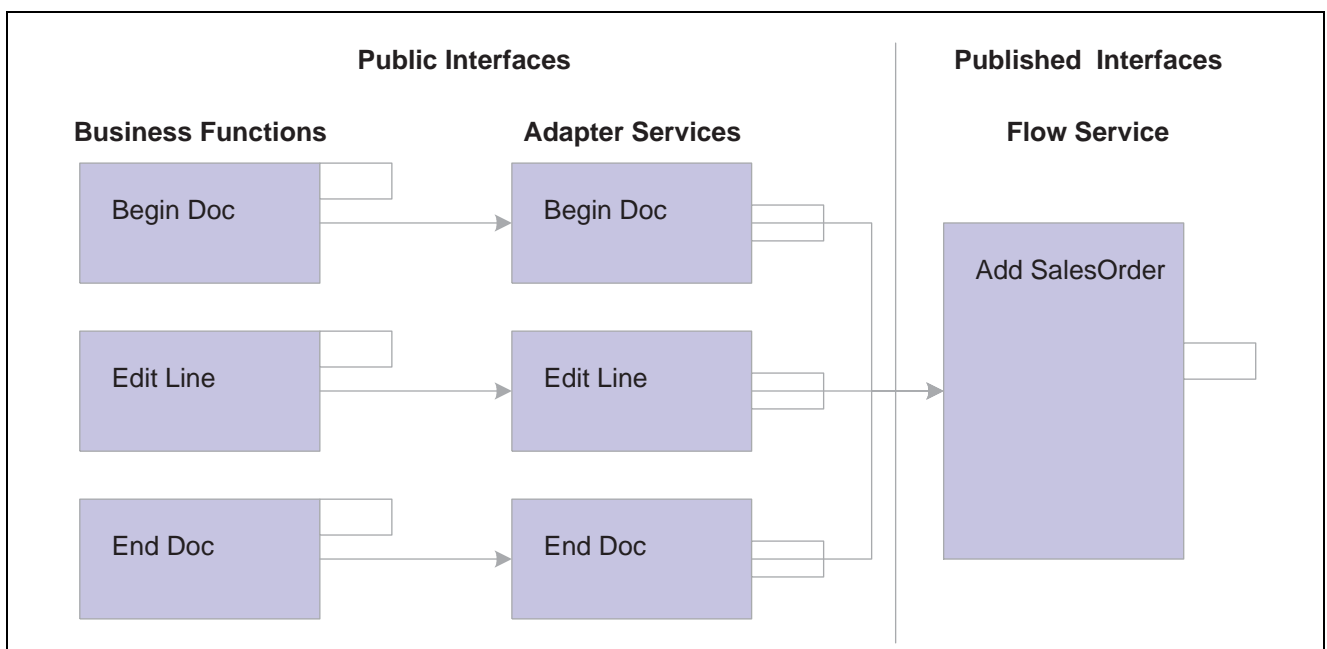
| Policy | Description |
|------------------|---|
| Published | Once an interface is published, modifications to the interface are not allowed. Any changes to a published interface require that a new interface be provided. The existing interface may be deprecated if it will no longer be used. |
| Component Scoped | A component scoped interface can be added, modified, or deleted without breaking the public or published interface; therefore, it does not need to be versioned. |

See Chapter 3, “Versioning Integrations,” [Deprecating Artifacts](#), page 24.

These diagrams show how the different interface policies affect the development of flow services that invoke the WSG EnterpriseOne adapter services.

For the first scenario, a flow and associated interface document are needed because the JD Edwards EnterpriseOne business function interface is at too low of a level of granularity. For example, to add a sales order, several business functions must be called to complete the task. Adding a sales order requires a call to BeginDoc, EditLine, and EndDoc. The interface that is exposed to external consumers is at a higher level.

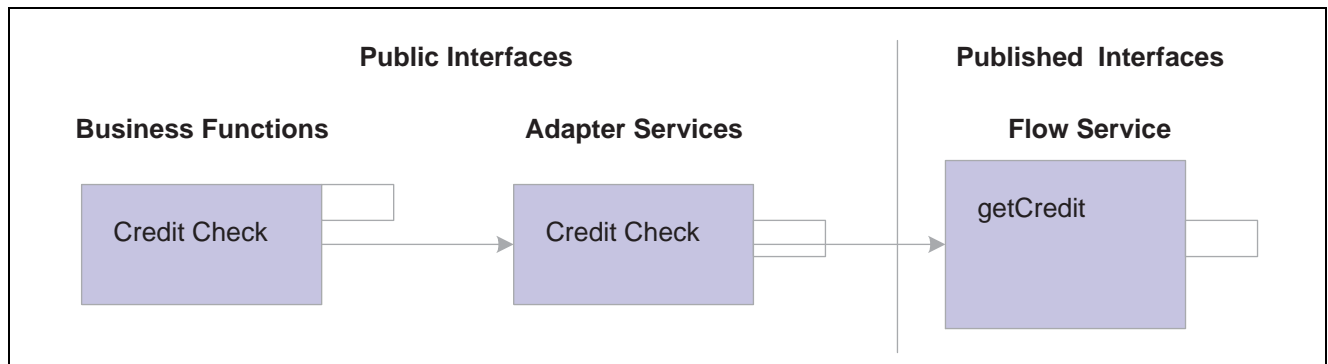
This diagram shows the interfaces that are associated with multiple business functions:



Interfaces for multiple business functions

For the second scenario, a single business function is needed to complete the task. You can publish the business function interface and not provide a flow or a flow interface document. If you do not provide a flow and the interface changes for the business function, the interface policy is broken. This happens in the JD Edwards EnterpriseOne system because an interface policy is not being followed for the business function interfaces. To follow the interface policy for integrations, a flow is required. The flow maintains the published interface and absorbs the changes to the business function interface.

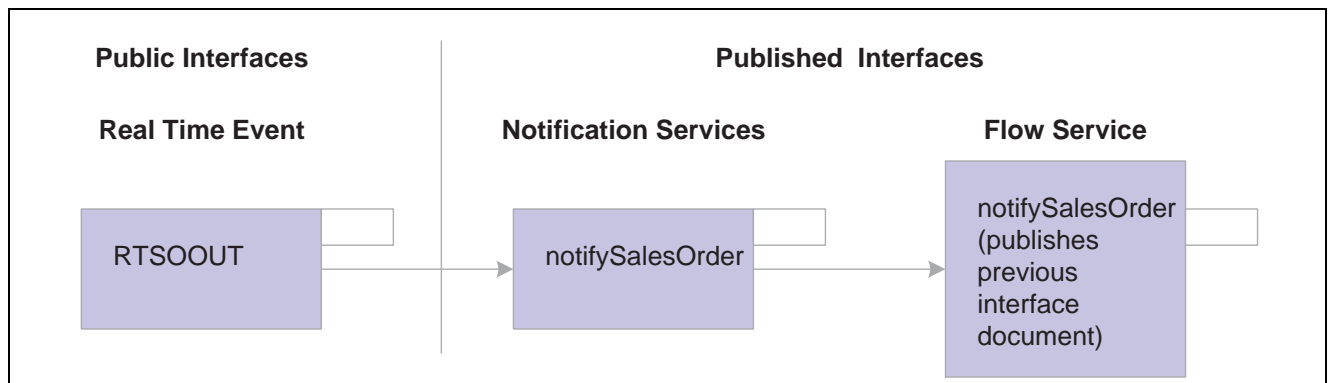
This diagram shows the interfaces for a single business function:



Interfaces for a single business function

For the third scenario, a notification interface document is used. Initially, a flow wrapping the notification service is not delivered. The notification service interface document is a published interface. The JD Edwards EnterpriseOne system does not follow an interface policy for the realtime event interfaces. To follow the interface policy for integrations when the realtime event interface changes, a flow is required to support the previous interface.

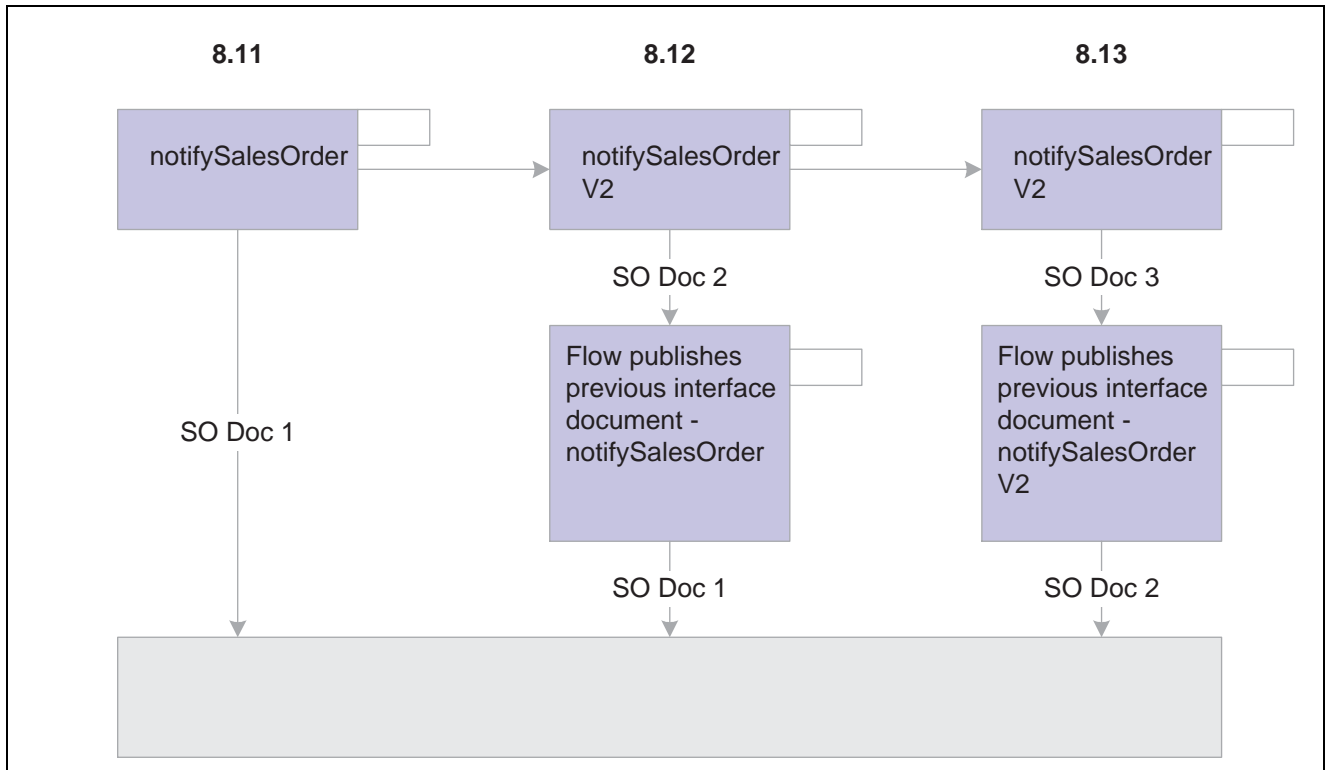
This diagram shows the interfaces associated with a realtime event:



Interfaces for a realtime event

Supporting Past Releases

This diagram shows how to support past releases of an interface:



Versioning example

The example shows a notification interface that is created for JD Edwards EnterpriseOne 8.11 release. In the JD Edwards EnterpriseOne 8.12 release the realtime event interface changes, forcing a new interface to be created. In the JD Edwards EnterpriseOne 8.13 release, the realtime event interface changes again, forcing a new interface for the JD Edwards EnterpriseOne 8.13 release.

Adapter Services

An adapter service is created from an operation template to perform a specific operation on a resource. For example, the JDBC adapter provides templates for working with a database.

You must create an adapter service within the context of an adapter. The adapter name provides additional information for the adapter service; therefore, the service does not need to be as fully qualified as an adapter's name.

In the case of application resources, an adapter service name should identify the native application function that is being performed.

Use this format when naming adapter services:

<verb><Noun>

This table describes the elements of adapter service names:

| Element | Description |
|---------|--|
| <verb> | <p>Describes the action. These values are recommended verbs. This list is not comprehensive. You can add to this list.</p> <ul style="list-style-type: none"> • get • update • create • delete • select • insert • process • calculate • parse • concatenate |
| <Noun> | Describes the object being manipulated (for example, Account Ledger, Exchange Rate). |

Note. When you create adapter services using the business function template of the WSG EnterpriseOne adapter, name the adapter service the same as the business function name. This helps to identify the service when selecting it to include in a flow.

Examples

This table lists examples of adapter services:

| Adapter Service | Description |
|-----------------------------|--|
| getCanonicalCode | JDBC adapter |
| setCodeLatch | JDBC adapter |
| getNextNumber | WSG EnterpriseOne adapter - business function template |
| insertAccountLedger | WSG EnterpriseOne adapter - database template |
| selectExchangeRate | WSG EnterpriseOne adapter - database template |
| notifyGLAccountMasterEvent. | WSG EnterpriseOne adapter - notification template |

Common Services

A common service provides general purpose functions. You can create these functions within the context of a package, for use by a group, and at the highest level for use by any package. Place the common service at the highest level of reusability. Common services that provide general function for any package are stored in the PSFT_Utils package.

Unlike adapter services, common services are not created from an operation template. You create them as Java methods that are wrapped within a flow that can have a pipeline as input and update the pipeline within the code.

Use this format when naming common services:

`<verb><Noun>`

This table describes the elements of common service names:

| Element | Description |
|---------------------------|---|
| <code><verb></code> | Describes the action. These values are recommended verbs. This list is not comprehensive, and you can add to this list. <ul style="list-style-type: none"> • get • update • create • delete • process • calculate • parse • concatenate |
| <code><Noun></code> | Describes the object being manipulated (for example, SystemDate, InvoiceNumber). |

Note. The verb is lower case and the noun is initial caps.

These are examples of common service names:

- concatenateValue
- getSystemDate

Transformation Services

A transformation provides the mapping between disparate interface documents.

Use this format when naming transformation service flows:

`map_<From doc>_To_<To doc >`

This table describes the elements of transformation service names:

| Element | Description |
|-------------------------------|---|
| <code><From doc></code> | Identifies the document from which you are mapping. |
| <code><To doc></code> | Identifies the document to which you are mapping. |

These are examples of transformation service names:

- map_E1_Company_To_E_BusinessUnit
- map_E_SalesOrder_To_E1_SalesOrder

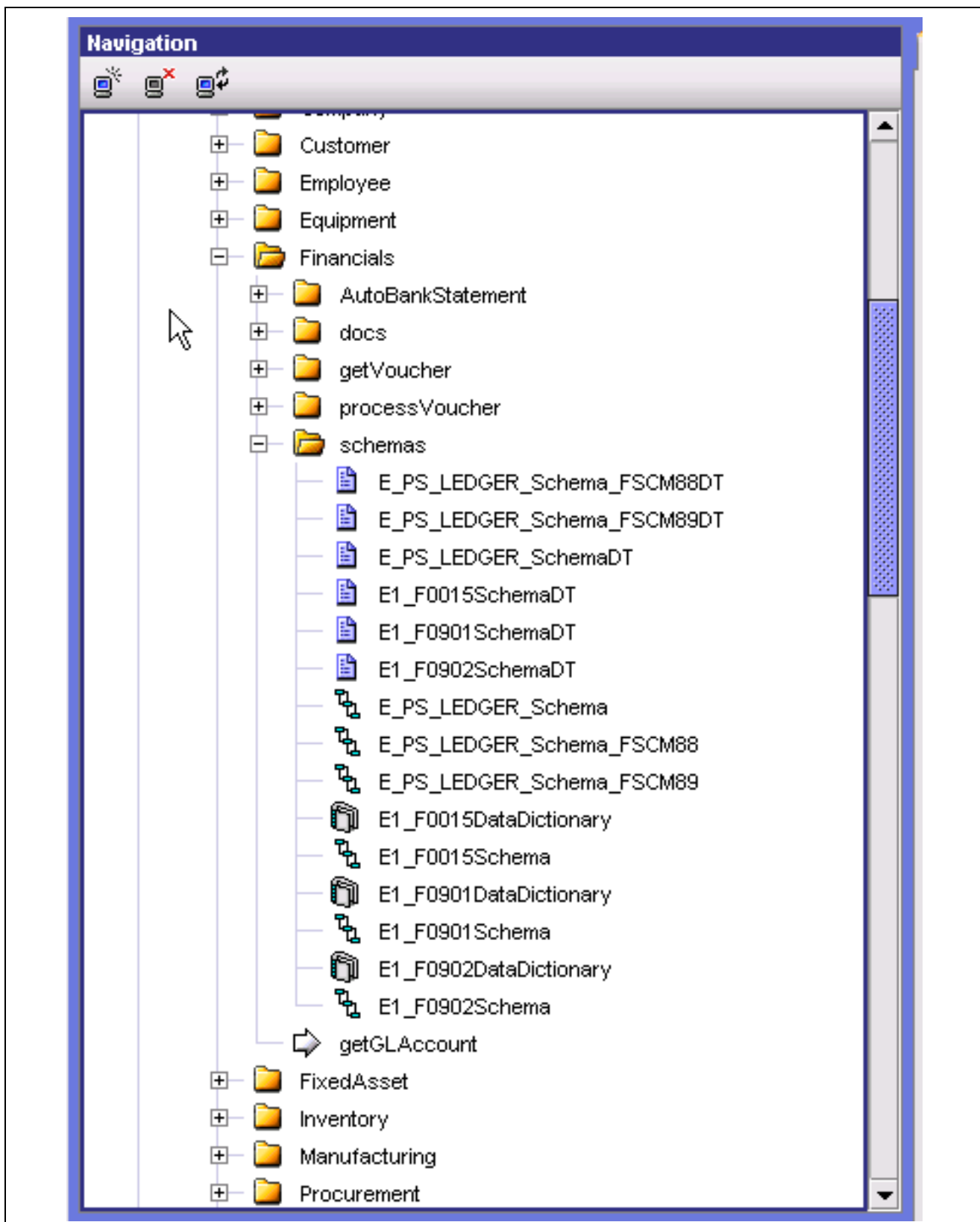
Flat Files

When performing transformations from or to flat files, you must create flat file schemas and possibly WSG flat file data dictionary artifacts.

Package Structure

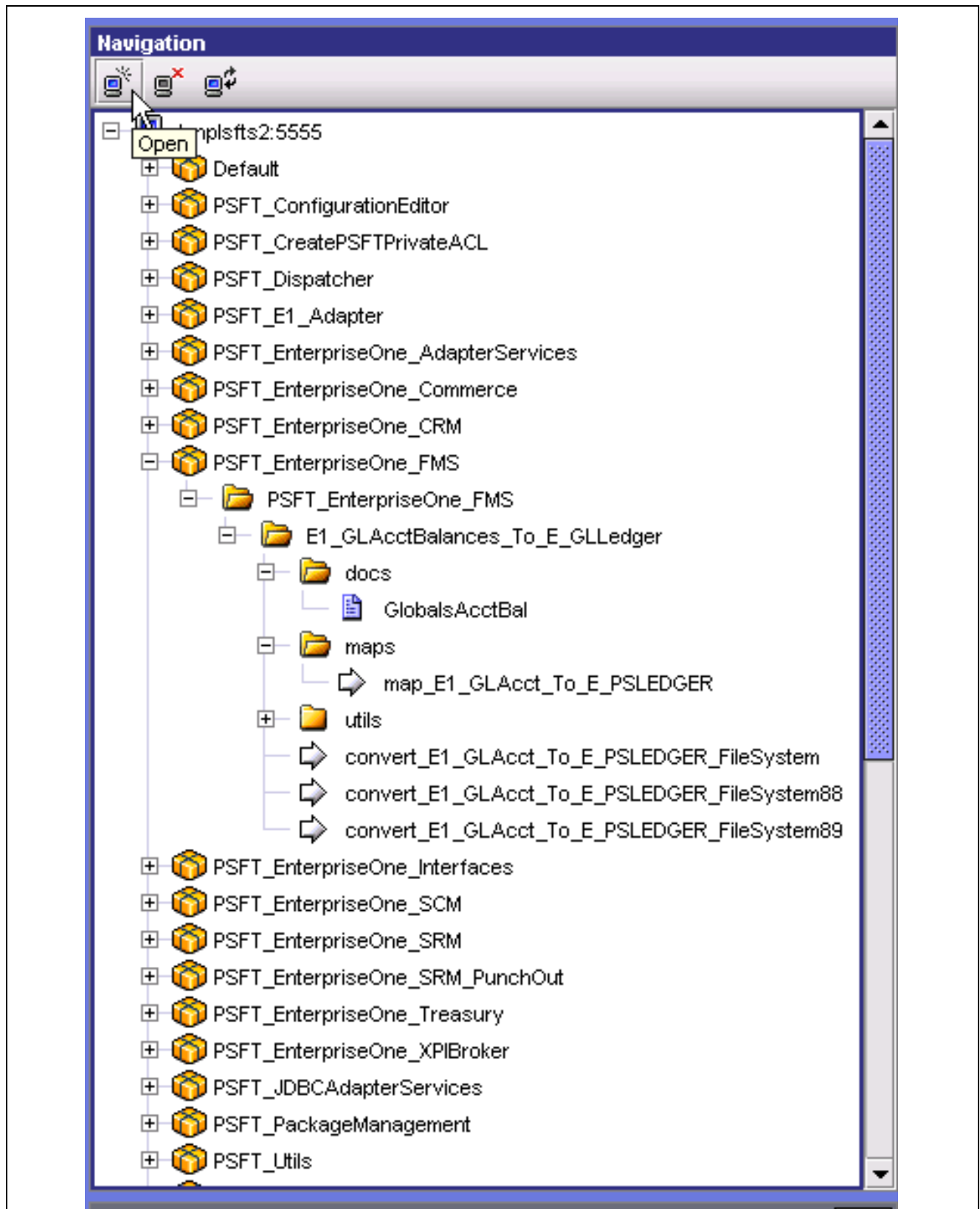
Because the schemas and document types are the external interfaces for the integrations using flat files, these artifacts are stored in the PSFT_EnterpriseOne_Interfaces package. Schemas, data dictionary objects and document types are all stored in the proper family under a schemas folder. The initial batch load integrations using these artifacts are stored in the appropriate package based on functionality, for example, PSFT_EnterpriseOne_FMS.

This screen shows an example of a schemas folder:



Example of a schemas folder

This screen shows an example of the structure of a package:



Example of package structure

Schemas

Use this format when naming flat file schemas:

```
<resource identifier >|<table name >|Schema
```

This table describes the elements of flat file schema names:

| Element | Description |
|-----------------------|--|
| <resource identifier> | Identifies the resource providing the schemas. Current resource identifiers are: <ul style="list-style-type: none"> • E:- Enterprise One • E: Enterprise • Blank: not specific to a resource |
| <table name > | Identifies the table for the schema. JD Edwards EnterpriseOne examples are F0092, F0401, and so on. |

This is an example of a flat file schema name:

E1_F0902Schema

Document Types

When naming document types based on flat file schemas, use the name of the schema appended by DT, as shown here:

```
<SchemaName>|DT
```

This is an example of the name of a document type that is based on a flat file schema:

E1_F0902SchemaDT

XPI Flat File Data Dictionary

Use this format when naming a flat file data dictionary:

```
<Resource Identifier >|<Table Name >| DataDictionary
```

This table describes the elements of flat file data dictionary names:

| Element | Description |
|-----------------------|--|
| <Resource Identifier> | Identifies the resource providing the schemas. Current resource identifiers are: <ul style="list-style-type: none"> • E1: Enterprise One • E: Enterprise • Blank: not specific to a resource |
| <Table Name> | Identifies the table for the data dictionary. JD Edwards EnterpriseOne examples are F0092, F0401, and so on. |

This is an example of the name of a flat file data dictionary:

E1_F0902DataDictionary

Flat File Data Dictionary Fields

Use descriptive names when you define flat file data dictionary fields. Users should be able to easily identify the field. For a JD Edwards EnterpriseOne flat file, use these naming convention:

<description>_<alias>_<table name>

This table describes the elements of flat file data dictionary field names:

| Element | Description |
|---------------|---|
| <description> | The description from the JD Edwards EnterpriseOne data dictionary for the data item in the flat file with spaces removed. Use the Java naming convention of lower and upper case to format the description. |
| <alias> | This is optional. Use an alias from the JD Edwards EnterpriseOne data dictionary for the data item in the flat file. An example for not having an alias would be a flat file that has separated the name into first, middle, and last. The JD Edwards EnterpriseOne table has a single field, name. |
| <table name> | This is optional and is only needed multiple tables are joined into the same flat file. This designates the table the field is coming from and eliminates duplicate field names. |

These are examples of flat file data dictionary field names:

- addressNumber_AN8
- addressNumber_AN8_F0101
- firstName

Triggers

Triggers execute a flow when a document is received.

Use this format when naming triggers:

```
trigger_<DocumentName>
```

The Name field in the configuration defaults to Condition1. Overwrite the default to invoke_<main flow name>.

These are examples of trigger names:

- trigger_AddressBookMasterManageNotify_2_0
- trigger_NotifyBusinessUnitEvent
- invoke_convert_E1_Response_To_E_And_Dispatch

Integration Flow Services

An integration flow describes the interaction between the integration tasks. It ties the integration logic together. An integration flow is where the actual logic is created. The integration flow is used to generate the WSDL for the web service.

Naming of flows follows the standards for naming services.

Use this format when naming integration flows:

```
<verb><Noun>
```

This table describes the elements of integration flow names:

| Element | Description |
|---------|---|
| <verb> | Describes the action. These values are recommended verbs. This list is not comprehensive, and you can add to this list. <ul style="list-style-type: none"> • Get • Update • Create • Delete • Add • Process |
| <Noun> | Describes the object being manipulated (such as Sales Order, Purchase Order, and so on). |

These are examples of integration flow names:

- addSalesOrder
- processSalesOrder
- getSalesOrder

The integration flow may also contain a transformation of data, like the integration flows between PeopleSoft Enterprise and JD Edwards EnterpriseOne.

Use this format when naming this type of integration flow:

```
<verb>_ <From>_<Noun>_To_<To>_<Noun>_And_<Return message action>
```

This table describes the elements of the name of integration flows between the PeopleSoft Enterprise system and the JD Edwards EnterpriseOne system:

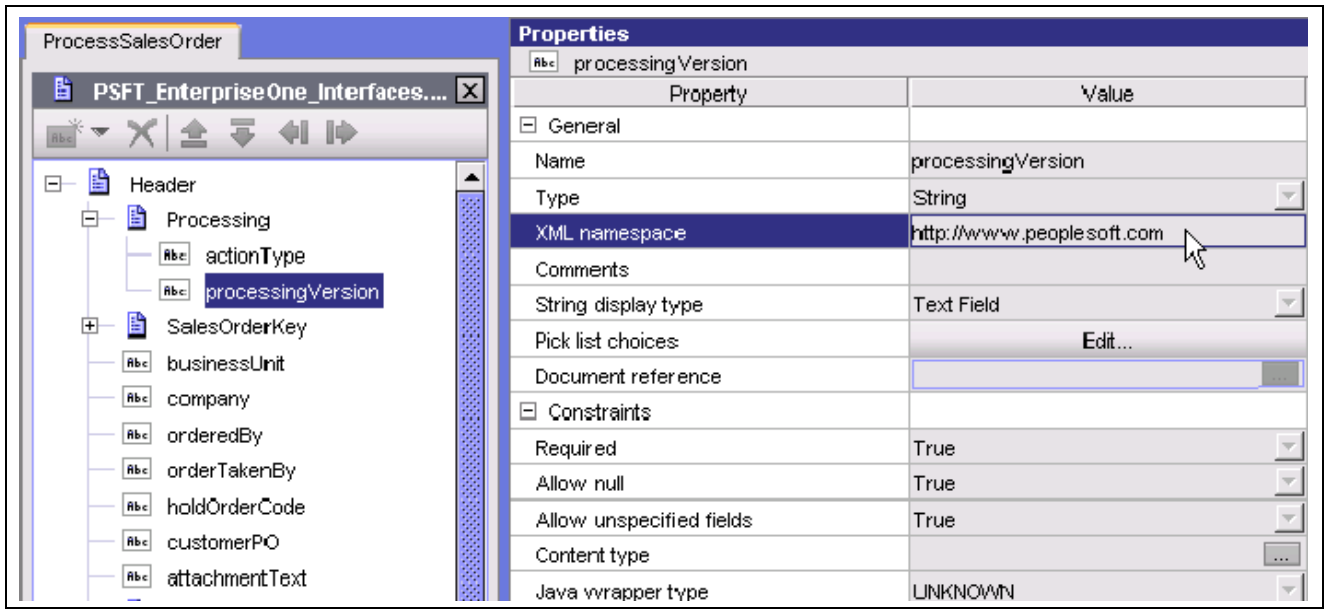
| Element | Description |
|------------------|--|
| <verb> | Describes the action. These values are recommended verbs. This list is not comprehensive, and you can add to this list. <ul style="list-style-type: none"> • convert • process • query • insert • update • delete |
| <from> | Identifies the resource from which you are transforming. |
| <Noun> | Describes the object being manipulated (such as Sales Order, Purchase Order, and so on). |
| <to> | Identifies the resource you are transforming to. |
| <Noun> | Describes the object being manipulated (such as Sales Order, Purchase Order, and so on). |
| <Return Message> | Indicates that a return message will be dispatched from the flow. This is optional and is only included if there is a return message. These values are recommended. This list is not comprehensive, and you can add to this list. <ul style="list-style-type: none"> • Response • Dispatch |

These are examples of the names of integration flows between the PeopleSoft Enterprise system and the JD Edwards EnterpriseOne system:

- convert_E1_Company_To_E_Business_Unit
- convert_E_Employee_To_E1_Employee_And_Response
- query_E_MarketRate_To_E1_ExchangeRate_And_Response
- insert_E_GlobalPayroll_To_E1_BatchJournalEntry_And_Response
- convert_E1_Response_To_E_Status_And_Dispatch

Editing XML Namespaces

Access the Properties for the object.



Properties

Converting XBPs to Flow Services

When converting XBPs to flow services, you must provide a flow and the documents that match the name of the existing XBP and canonical. This is in addition to any new documents and flow service that you create. This provides backwards compatibility for existing customers.

Understanding Service WSDL

The WSDL service document is equivalent to a business service as published to a registry.

Use this naming convention for the service WSDL:

```
<Flow Name>.wsdl
```

These are examples of WSDL service documents:

```
addSalesOrder.wsdl
```

```
updatePurchaseOrder.wsdl
```

To convert an XBP to a flow service, use the generation option in Developer to generate the WSDL, and then save the WSDL to the pub/services directory for the package. In the file system, go to this location:

```
x:\PeopleSoft\xpi\IntegrationServer\packages\package name\pub
```

where *x* is the drive where you installed WSG. Create a services directory under the pub directory for the package.

CHAPTER 3

Versioning Integrations

This chapter discusses how to:

- Version interface documents.
- Version flow services, adapter services, and utility services.
- Deprecate artifacts.
- Modify delivered integrations.

Versioning Interface Documents

Interface documents are a published interface; therefore, modification to the interface is not allowed. Any changes to a published interface require that a new interface be provided. The existing interface may be deprecated if it is no longer used.

We have identified these scenarios for modifying an interface document:

- A field needs to be added to a published interface.

This field does not constitute new functionality but is a necessary field that does not currently exist in the interface.

For this scenario, you create a new interface document that contains the existing interface fields plus the additional fields. The new interface document is named the same as the original document with a version appended. In this scenario, the original interface is deprecated because you are creating a new version of the interface that replaces the original. The new interface document name is `InterfaceDocumentName_v2`.

- A field or fields for new functionality need to be added to a published interface.

In this scenario you are defining a new interface. This new interface is substantially different from the original interface.

For this scenario, you must create a new interface document. The new interface document is given a new name that expresses the new functionality. The existing interface document remains unchanged and can be deprecated if no longer used.

Use of a version number, instead of assigning a new meaningful name each time the interface changes, allows any number of additions to the interface while not greatly increasing the length of the interface document name. The drawbacks to using a version are:

- Users cannot easily identify the changes to the interface document. However, the documentation for each version should clearly and concisely outline the functionality that the interface provides.
- The version number implies that the new version contains a superset of earlier versions (that is, it contains all the functionality of the previous versions and more). This results in interface documents growing larger and larger, gradually making it more difficult to populate values.

Versioning Flow Services, Adapter Services, and Utility Services

Flow services, adapter services, and utility services are tied to an interface. When changing the interface to flow services, adapter services, and utility services, follow the naming methodology that the interface document uses. Whenever the underlying interface changes for the flow service or the adapter services or the utilities, you must create a new service with a version appended to the original service name. You decide whether to deprecate the original service or to keep it based on the functionality provided by the new service.

This table shows examples of service names that have been versioned:

| Service Type | Example |
|----------------|---|
| Flow Service | map_E_SalesOrder_To_E1_ProcessSalesOrder_v2 |
| AdapterService | getCanonicalCode_v2 |
| Utility | concatenateValue_v2 |

Note. Any services that are needed for the JD Edwards ERP 8.0 system or the JD Edwards OneWorld Xe system that do not support the same interface as the JD Edwards EnterpriseOne system do not follow these versioning guidelines. For this case, the service created for the JD Edwards ERP 8.0 system or the JD Edwards OneWorld Xe system is appended with ERP8 instead of a version. The JD Edwards EnterpriseOne service does not have a version, and is considered the first release of the service.

This table shows examples of service names that have been versioned for ERP 8.0:

| Service Type | Example |
|----------------|---|
| Flow Service | map_E_SalesOrder_To_E1_ProcessSalesOrder_ERP8 |
| AdapterService | getIntegrationOption_ERP8 |
| Utility: | concatenateValue_ERP8 |

Deprecating Artifacts

When deprecating an artifact, update the HTML documentation with a comment that the artifact is being deprecated. Place a deprecated entry in the comment and include text that specifies when the artifact was deprecated; if possible, suggest a replacement artifact.

Product management determines the length of time the deprecated artifact is preserved.

Modifying Delivered Integrations

We recommend that you never modify delivered integrations, including all artifacts within a PSFT_package. This preserves the changes when installing an upgrade. You should create a new folder, copy the integration into this new folder, and then make the modifications to the copy of the integration.

All new components should conform to this naming convention:

Use a three-letter company identifier in uppercase (such as IBM, DSI) for all artifacts that are created, including packages and services. In the examples, the *XXX* represents the three-letter company identifier.

This table lists examples of artifacts that have been copied and named with a three-letter company identifier:

| Artifact | Example |
|----------------|------------------------------------|
| Package | <i>XXX_EnterpriseOne_SCM</i> |
| Folder | <i>XXX_SCM</i> |
| FlowService | <i>XXX_convertGLAcctToPsLedger</i> |
| AdapterService | <i>XXX_GetF0101Z1Record</i> |
| Utility | <i>XXX_getIntegrationOption</i> |

CHAPTER 4

Understanding Development Conventions

This chapter discusses conventions for:

- Coding standards
- Documentation
- Testing

Coding Standards

This section discusses:

- Integration Server development conventions.
- Error handling.
- Transaction and flow service design.
- Invoking flow services with explicit transaction boundaries.
- Transactions and stepping/tracing in Developer.
- Cross-referencing.
- Integration options.

Integration Server Development Conventions

These conventions apply to all integration development done in Integration Server:

- Define all fields in the interface document with a string data type and update properties with schema data type constraints.
- Assign a value set for the XML namespace property for all fields in an interface document.

For all fields, that value is <http://www.peoplesoft.com/>.

See [Chapter 2, “Managing Integration Server Components,” Editing XML Namespaces, page 20](#).

- Never hard code constants into a pipeline mapping.

Include a DeclareGlobals map step to define all constants and variables. Define constants as integration options if the user needs to change the value. The structure should look like this:

```
Globals
  Constants
  Variables
```

- Define all variables as lower and upper case.

Follow Java standards in naming the variables.

- Define all constants as all caps and use an underscore (_) in the name to separate words.
- Set logging levels for each flow service.

Set logging levels on the Audit tab as described in this table:

| Option | Value |
|---------------------|-------------------|
| Enable auditing | Always |
| Log on | Error and success |
| Include in pipeline | On errors only |

- When attaching the input and output documents to the flow, use the reference instead of selecting the input and output doc.
- Implement error handling standard logic.

See [Chapter 4, “Understanding Development Conventions,” Error Handling, page 28](#).

- Comment all custom code.
- Remove all commented code lines and any printlns used for debugging.
- For all branch steps and map steps, provide short meaningful names that describe the action that the component performs in the comment field.

Note. When you add a component to a flow, it is given a default name.

- Use invocations when possible instead of transformers.
- For transaction type interface documents, make separate document types for header and detail.

These are reusable and are used as input/output to the ProcessHeader and ProcessDetail common services for the integration flow.

Error Handling

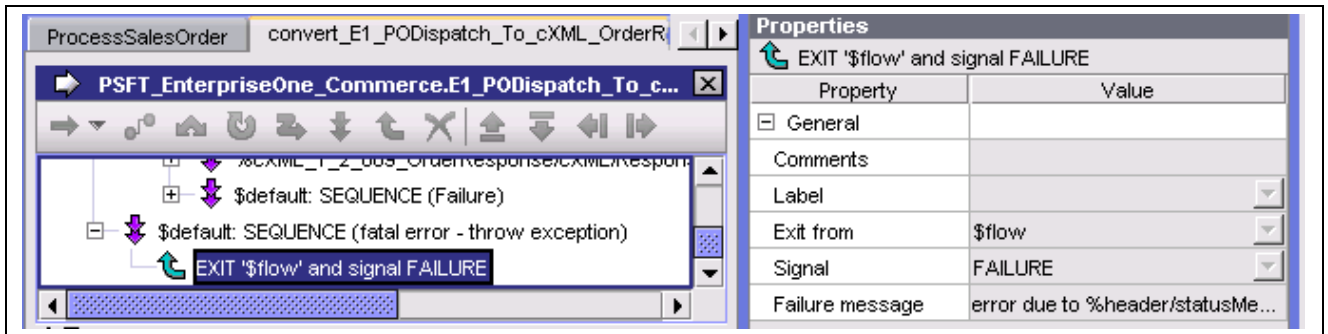
An exception is returned to the calling application for any of these types of errors that occur during execution of the flow:

- System-level errors (for example, Enterprise Server unavailable).
- Application-level errors (for example, business function errors).
- Critical warnings (for example, business function warnings that halt processing).

A flow returns an exception signifying an error via a manual Exit step or an uncaught adapter service error. A flow can return a confirm/show document signifying success of the operation. A flow cannot return both an exception and confirm/show.

Exit Step

To manually throw an exception back to the calling application, use an Exit step, as shown in this example:

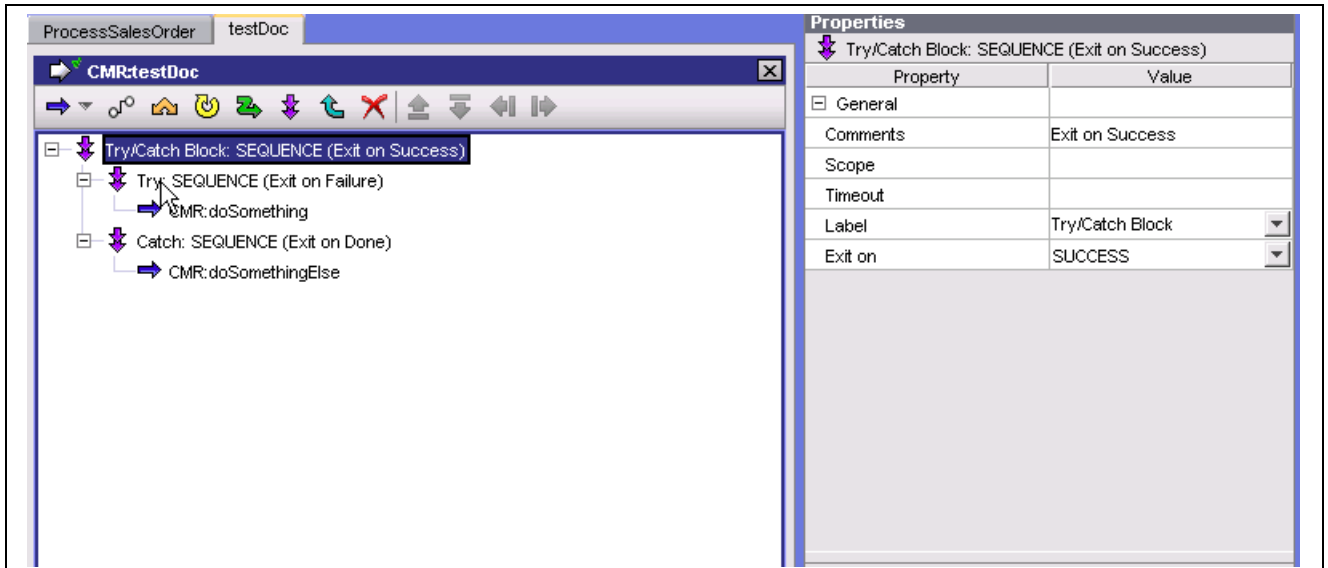


Exit step

To exit completely out of a flow and return an exception to the calling application, set the from field to \$flow, set the signal field to FAILURE, and set the failure-message field set to a literal string (that is, the error message to return) or a variable in the pipeline.

Try/Catch

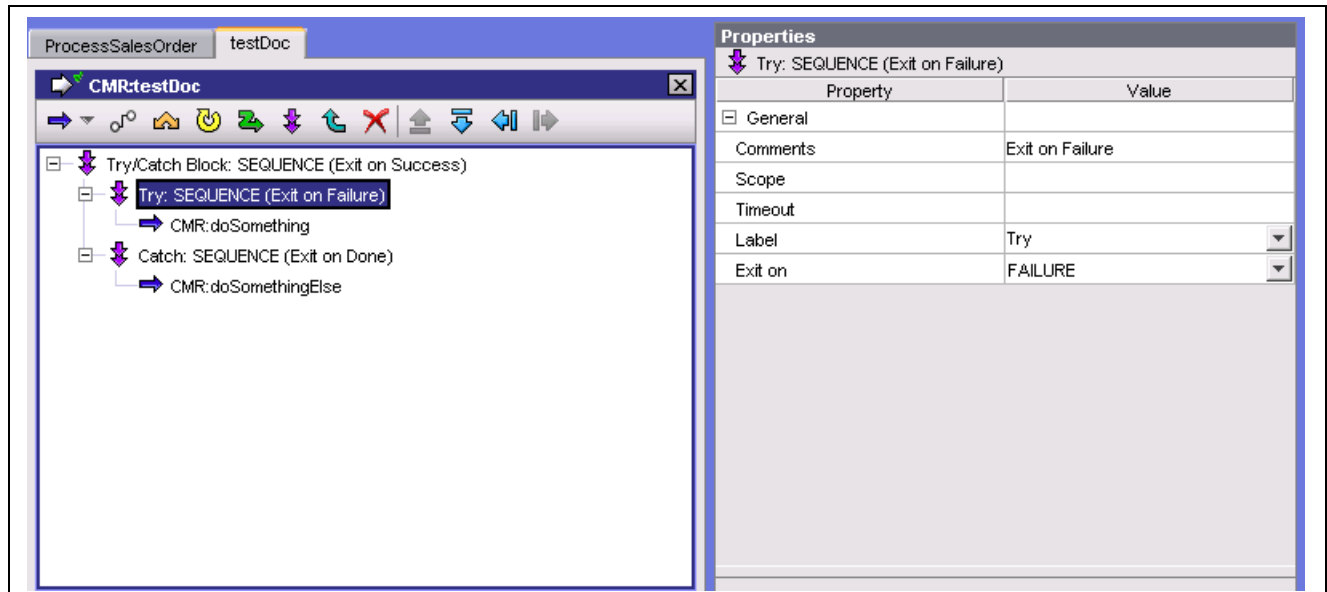
You can simulate a Try/Catch block in a flow using a set of nested Sequence statements. The first sequence step denotes the overall Try/Catch block, as shown in this example:



Try/Catch Block with nested Sequence statements

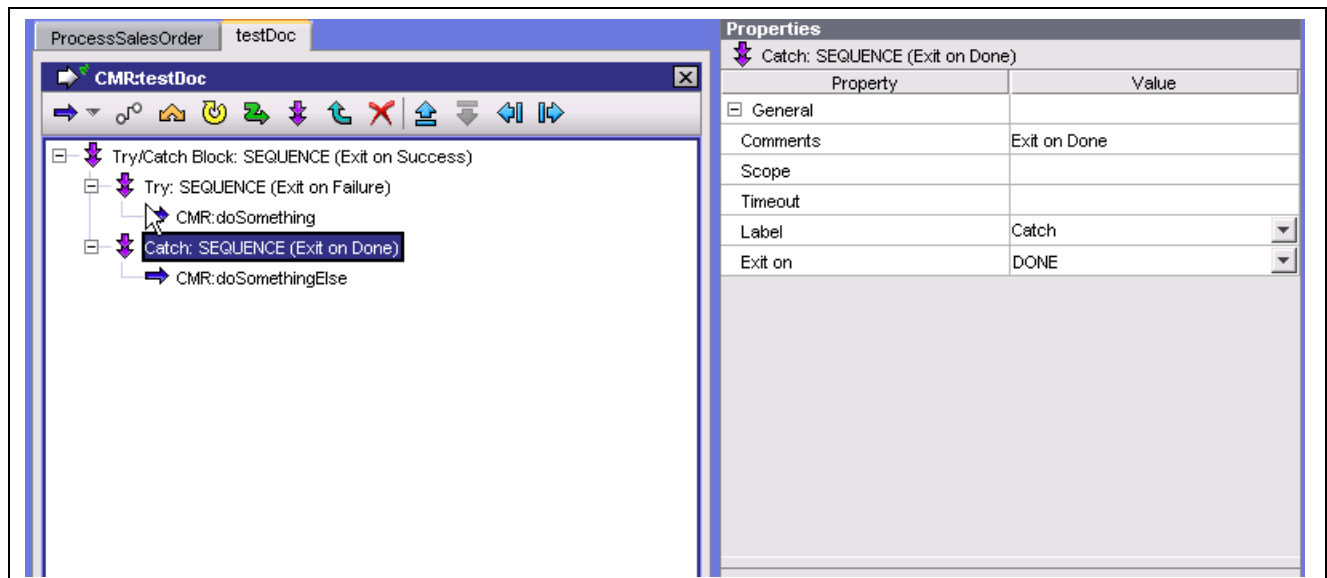
The exit-on field for the overall Try/Catch block should be set to SUCCESS.

The second sequence step is the Try portion of the Try/Catch block, as shown in this example:



Try step

The Exit-on field for the Try portion of the Try/Catch block should be set to FAILURE, as shown in this example:



Catch step

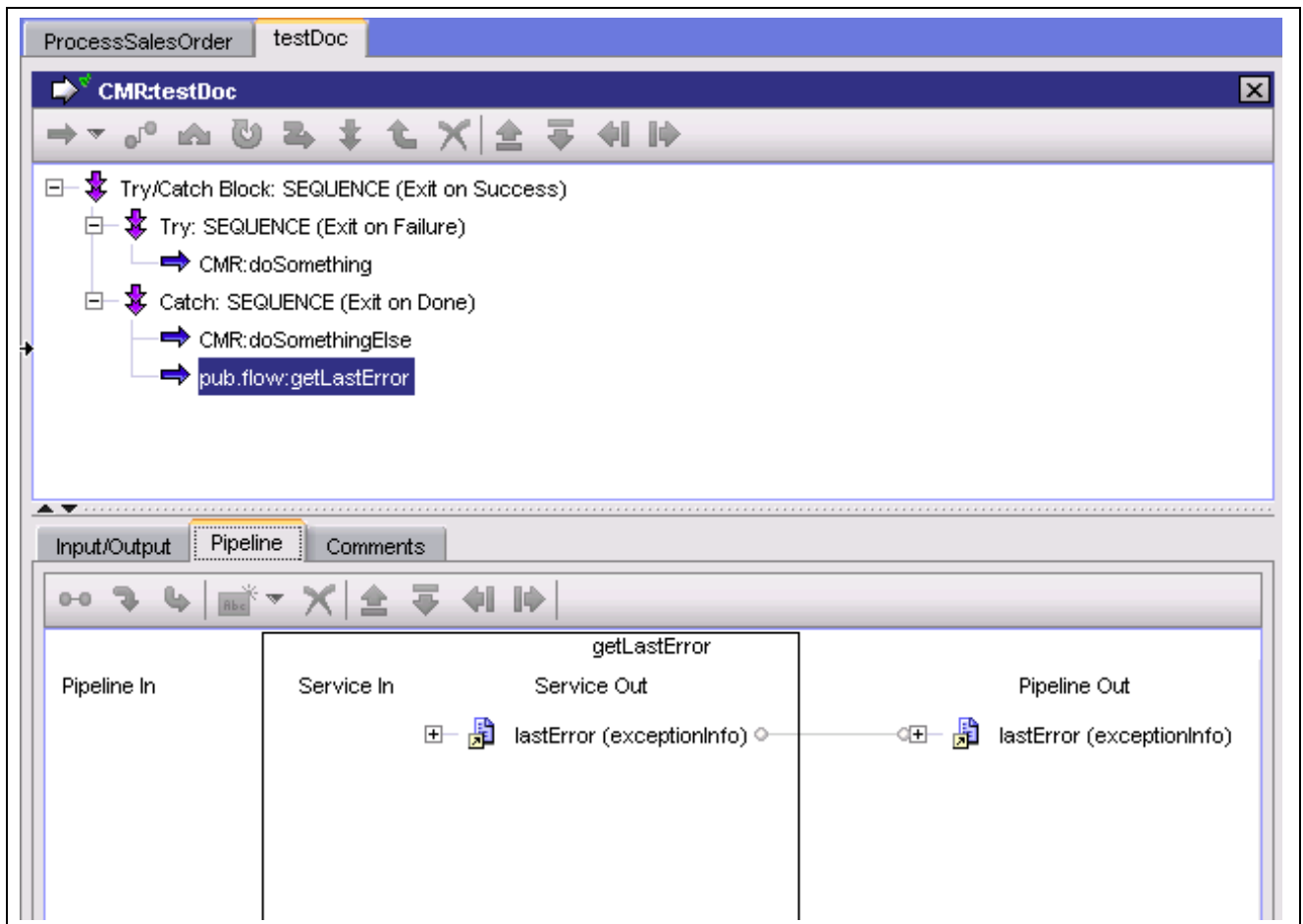
The Exit-on field for the catch portion of the Try/Catch block should be set to DONE.

getLastError service

A built-in service called `getLastError` is provided by `webMethods`. The `getLastError` service retrieves the text of the last exception thrown.

This service is useful in catch blocks. For example, the call to an adapter service is in a Try block. If the adapter service throws an exception, but you don't want to exit the flow immediately, you can call the `getLastError` service in the Catch block, and then retrieve the error text and store it in the pipeline for later use.

The `getLastError` service returns a document of type `exceptionInfo`, as shown in this example:



getLastError service

Child-Record Processing

As a standard, when errors are encountered on a critical child record, the flow is not exited immediately. The error is accumulated, via a pipeline variable, and processing of the remaining child records continues. Only after all critical child records have been processed is the flow exited by throwing an exception back to the calling application. In this case, the accumulated errors are concatenated into a string variable, and this string variable is passed to the failure-message field in the Exit step.

Note. A critical child-record is a child-record that is considered vital to the overall business process. For example, sales order line items are considered critical to the overall success of processing a sales order. A non-critical child-record is a child-record that is not considered vital to the overall business process. For example, phone records for a contact are not considered critical to the overall success of processing a contact. In this case, errors on phone records are accumulated and each phone record is processed. However, the flow does not exit from the flow (using an Exit step) when finished, but sends back a confirm document signifying success of the operation. The confirm document contains the child-record errors as messages.

Warnings

Non-critical warning messages are warnings that are bypassed. Non-critical warning messages received from an adapter service are returned to the calling application in a Confirm/Show document as messages. Non-critical warning messages are not treated as exceptions.

Critical warning messages are warnings that are not bypassed. Critical warning messages received from an adapter service are returned to the calling application as an exception. Critical warning messages are treated as errors.

Transactions and Flow Service Design

Integration Server automatically begins a transaction prior to invoking the first adapter service in a flow. At the end of the flow, all database inserts, updates, and deletes are either committed or rolled back. This includes any database inserts that occurred in the main flow, in any flows that were invoked, and in any adapter services that were invoked. If the flow is successful, the data is committed. If the flow receives an exception, then the data is rolled back.

An example is a flow that creates a new customer. The flow invokes an adapter service to add an address book record. If that was successful, the flow then invokes an adapter service to add a customer record. If the add of the customer record produced an exception, Integration Server rolls back the address book record. If you did not want the address book record to be rolled back, you would need to manually control the transaction boundary.

If you need to control the transaction boundary for the flow, you can explicitly start the transaction and manually commit and rollback. In the previous example, you could explicitly start the transaction by invoking the `pub.art.transaction:startTransaction` before calling the address book adapter service. Within a Try/Catch block, invoke the address book adapter service. In the Try, invoke the `pub.art.transaction:commitTransaction`, and in the Catch, invoke the `pub.art.transaction:rollbackTransaction`. You can then do another `startTransaction` for the customer adapter service and do the same thing with a Try/Catch block for the customer adapter service.

To control the transaction boundary, observe these rules:

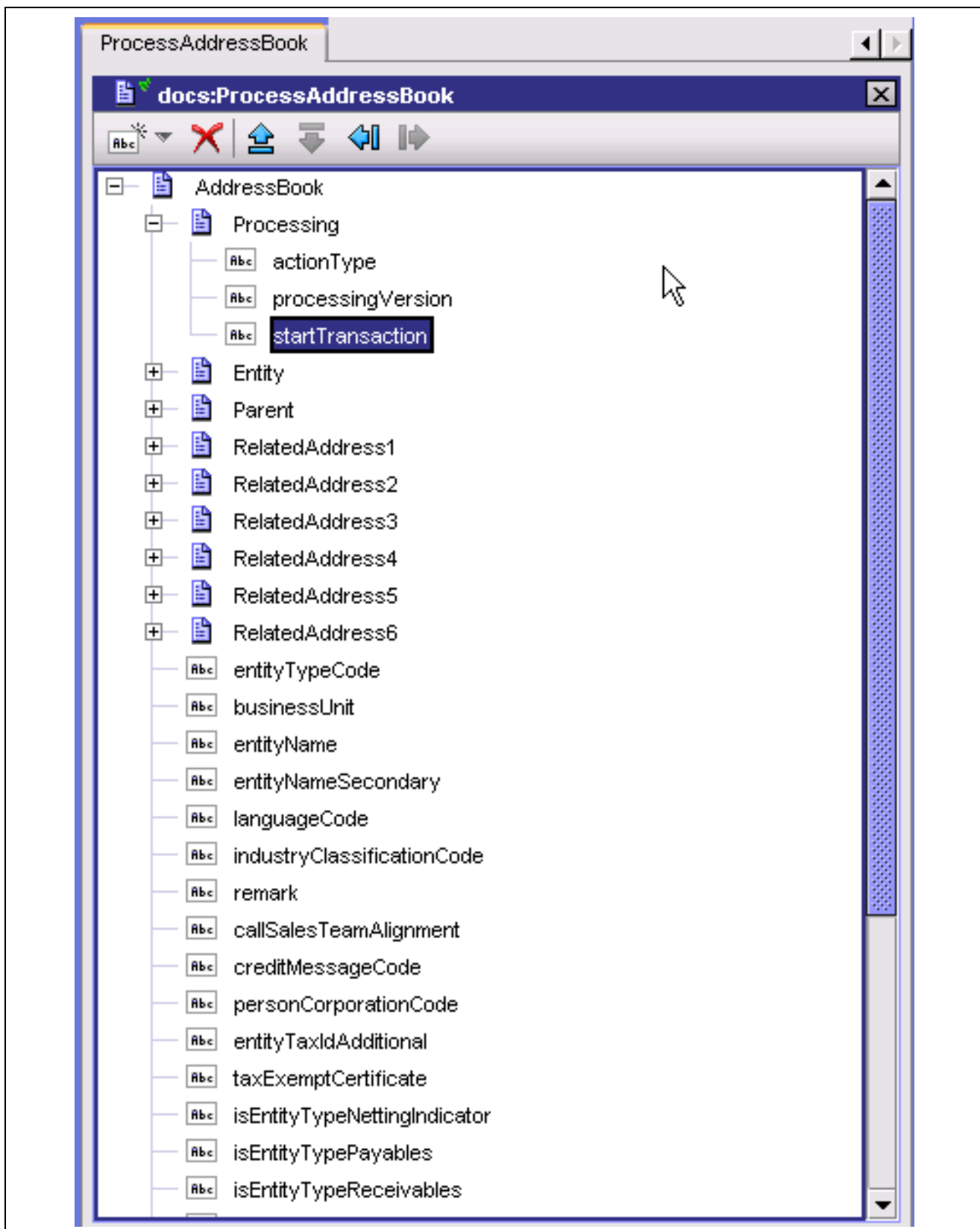
- Include an explicit `startTransaction`, `commitTransaction`, and `rollbackTransaction` invocation.
- Invoke the `commitTransaction` and `rollbackTransaction` within a Try/Catch block.
- Include an indicator in the interface to the flow to determine if the transaction needs to be started.

This rule is optional and can be helpful in debugging and tracing.

- Enclose the `startTransaction`, `commitTransaction`, and `rollbackTransaction` in branch statements based on the indicator in the interface.

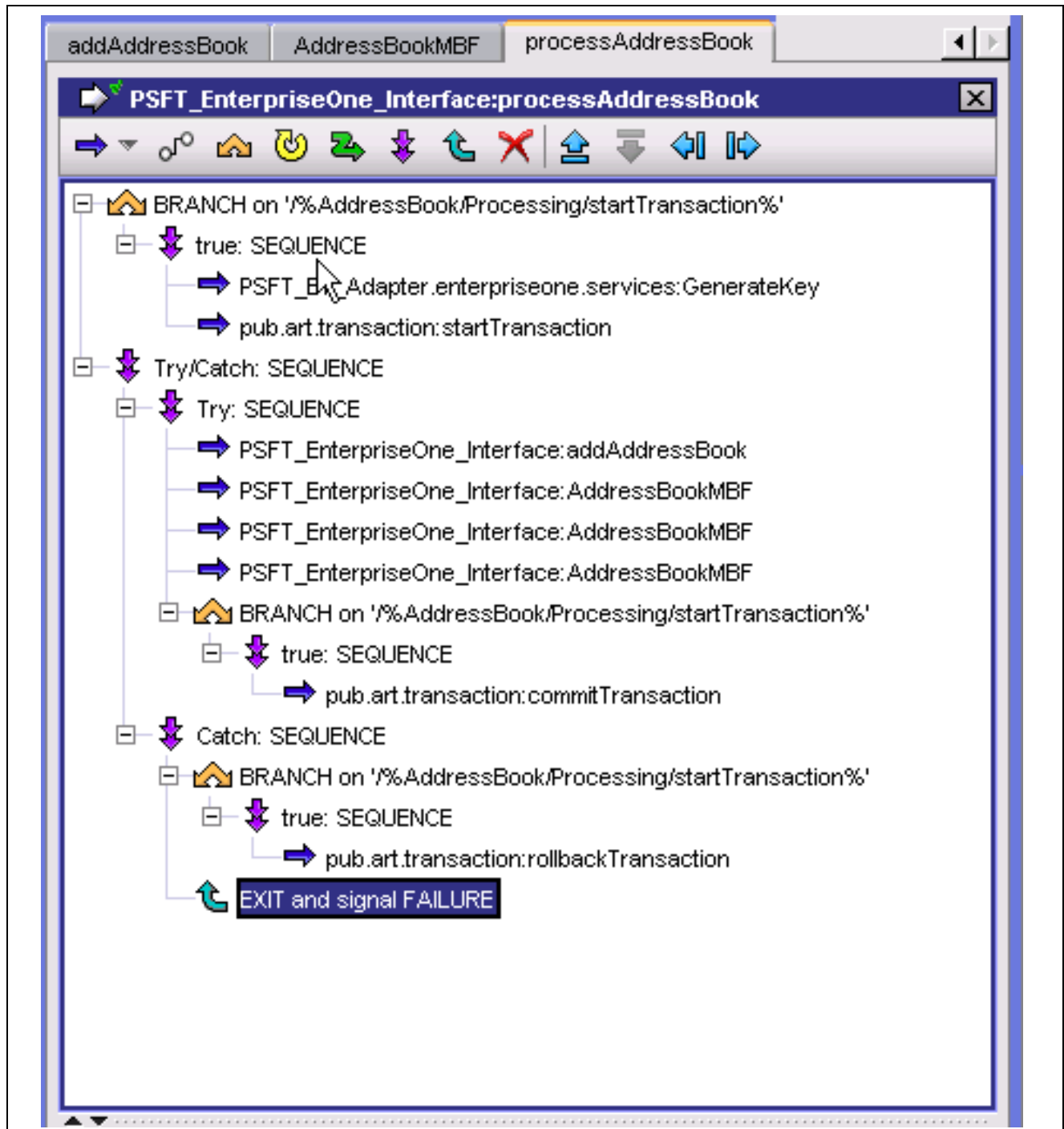
If another flow calls the flow, false is sent in the indicator, and the `startTransaction`, `commitTransaction`, and `rollbackTransaction` are bypassed. Nested transactions are not supported by Integration Server. The calling flow determines the transaction boundary and the flow is included in it.

This is an example of an interface with a `startTransaction`:



startTransaction invocation

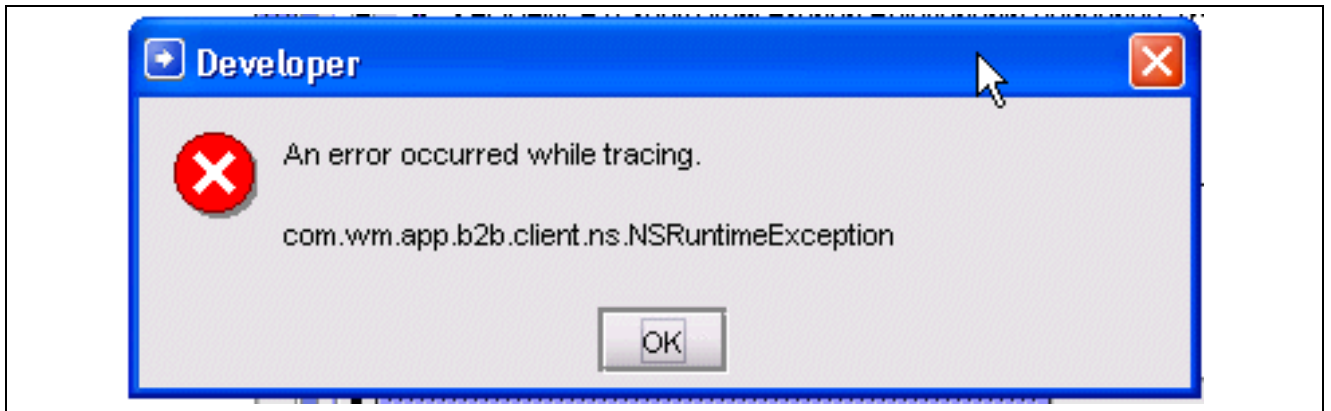
This is an example of branch statements around the startTransaction, commitTransaction, rollbackTransaction, and Try/Catch blocks:



Branch statements

Invoking Flow Services with Explicit Transaction Boundaries

When invoking a flow service that has an explicit transaction boundary for another flow service, you must send false in the indicator in the interface. If you invoke with a true value, you will receive this error when the Integration Server attempts to start another transaction:



Transaction error

Transactions and Stepping/Tracing in Developer

According to the webMethods 6 ADK User's Guide, you cannot step or trace through a flow that has transaction control. If you set your own transaction boundary in the flow when stepping or tracing in Developer, set the interface flag to false. This bypasses the `setTransaction`, `commitTransaction`, and `rollbackTransaction` invocations, and enables you to step/trace the flow.

This screen shows the `startTransaction` field with a value of *false*:

Input for 'processAddressBook'

ProcessAddressBook AddressBook Processing

actionType

processingVersion

startTransaction false

Entity

entityId

entityLongId

entityTaxId

Parent

entityId

entityLongId

entityTaxId

RelatedAddress1

entityId

entityLongId

entityTaxId

RelatedAddress2

entityId

entityLongId

entityTaxId

OK Cancel Load Save Help

startTransaction field

Cross-Referencing

The WSG Developer tool includes a set of common services that enable you to write and retrieve cross-reference values for use in integrations. These common services can be found in the PSFT_XrefAndSoftCoding package.

See [Appendix C, “PSFT_XrefAndSoftCoding Services,”](#) page 45.

You use the WSG Configuration Editor tool to load and maintain the cross-reference values for the disparate systems. A unique canonical ID ties the values together.

See *JD Edwards EnterpriseOne Tools 8.97 Web Services Gateway: Configuration Editor Guide*, “Using the Configuration Editor”.

Integration Options

The WSG Configuration Editor tool allows you create integration options instead of using literals in integrations. Creating integration options allows you to customize the values as needed and also provides the flexibility to modify a value without having to change the integration code directly. Using Configuration Editor, you can configure the integration options to your values. You register all integration options in the Configuration Editor tool and then refer to the integration option key. You use common services to fetch the integrations options within integrations. These services are found in the PSFT_XrefAndSoftCoding package.

See [Appendix C, “PSFT_XrefAndSoftCoding Services,” page 45.](#)

See *JD Edwards EnterpriseOne Tools 8.97 Web Services Gateway: Configuration Editor Guide*, “Using the Configuration Editor”.

Documentation

The XPI Integration Developer tool enables you to create HTML documentation for services created in the tool. For each service, flow, or common service, the integration developer is required to generate the HTML documentation and place it in the package for that service. You must do a Save As for the HTML document and save it in the pub directory for the package. On the Save As, change the type to web archive, single file(*.mht). This is the file system path:

```
x:\PeopleSoft\xpi\IntegrationServer\packages\package name\pub
```

where x is the drive where you installed XPI.

Integration Server Administrator provides the ability to display a home page HTML for each package. When you create a package, an index.html file is placed in the pub directory for the package. You need to replace this empty HTML file with the PeopleSoft template home page HTML file. Update the template with high-level information for the flows contained in the package.

If you add flows to an existing package, you must check out the index.html file for the package from the source control system, add information about the new flow, and check the index.html file back into the source control system.

Make sure you check in the HTML files to the source control system.

See [Appendix A, “Source Control,” page 39.](#)

Testing

This section discusses test script development and maintenance, and provides guidelines for testing.

Test Script Development and Maintenance

Test scripts for the testing facility within the Integration Server are stored in a PSFT_IS_TestScripts project in Clear Case along with the stubs needed for functional testing. After you create and test the script in Integration Server, save the script to the resources directory for this package and check the scripts into the source control system.

See [Appendix A, “Source Control,” page 39.](#)

The name of a script should conform to this format:

```
<Script_FlowServiceName_TestScenario>
```

These are examples of test script names:

- Script_CreditCheck_ModelSuccess
- Script_addSalesOrder_ModelSuccess

- Script_addSalesOrder_ModelFailure

Guidelines for Testing

Observe these guidelines when testing integrations:

- When you create or modify a flow service, the flow service must be accompanied with a minimum of two test scripts: one for a test case of normal operation (no errors) and another for a test case that generates errors (invalid data).
- The test scripts that you develop should be test cases for most code coverage (for example, multiple detail lines).
- If the flow service is complex, additional test scripts might be developed by the software engineer.
- Quality Assurance (QA) is expected to expand on the test scripts provided by the software engineer and create more test scripts.
- For a given integration, the Test Plan should clearly identify what test scripts must be developed by the software engineer and which must be developed by the QA.
- If the Test Plan is not created and approved at the time the software engineer moves the SAR to status 26, the test scripts created by the software engineer should be well documented in the T-Section of the SAR.

Until the test scripts that are to be developed by the software engineer are available, the QA is not expected to start functional testing of the integration.

APPENDIX A

Source Control

This appendix discusses:

- Clear Case structure.
- Views.
- WSG EnterpriseOne Adapter Services package structure.
- Manifest files.
- Testing scripts.

Clear Case Structure

The WSG integration development objects are stored in Clear Case in this structure:

- Project per E1 release (for example, JD Edwards EnterpriseOne 8.11, JD Edwards EnterpriseOne 8.10, JD Edwards EnterpriseOne 8.9).
- Under the project, the folder structure matches the folders under the Integration Server beginning with the package folder name.

All artifacts under this folder are stored in Clear Case. The build process generates the deployment Zip package.

This is an example of the folder structure in Clear Case:

```
PSFT_CRM_DataRetieval
PSFT_CRM_JDBCAdapterServices
PSFT_EnterpriseOne_AdapterServices
ns
  AdapterServices
    Business_Function
    Database
    Notification
    XAPI_Response
    XML_List_Query
  EnterpriseOne_AdapterServices_Listener
pub
PSFT_EnterpriseOne_CRM
PSFT_EnterpriseOne_HCM
PSFT_EnterpriseOne_SRM
PSFT_HCM_DataRetrieval
PSFT_HCM_JDBCAdapterServices
```

Views

All integration services packages are stored in the E1A_PPI tree in Clear Case. Each project has a Development view profile.

WSG EnterpriseOne Adapter Services Package Structure

This is the folder structure of the PSFT_EnterpriseOne_AdapterServices package.

```

PSFT_EnterpriseOne_AdapterServices
  AdapterServices
    BusinessFunction
      [Family]
      [BusinessFunction adapter service]
    Database
      [Family]
      [Database adapter service]
    Notification
      [Family]
      [Notification adapter service]
      [Notification adapter service associated publish document]
    XAPI Response
      [Family]
      [XAPI Response service]
    XML List Query
      [Family]
      [XML List Query service]
    EnterpriseOne_AdapterServices_Listener
      EnterpriseOne_Listener

```

Note. For the PSFT_ERP8_AdapterServices package, the structure is the same except the name EnterpriseOne is replaced with ERP8.

Manifest Files

When you create a package and checked in for the first time, you must also check in a manifest.v3 file. To update the information in the manifest.v3 file, highlight the package in Developer. The information in the tabs on the right-hand side is stored in the manifest.v3 file. Update accordingly and then when checking in, pick up this file from the package directory in the file system.

```
X:\PeopleSoft\xpi\IntegrationServer\Packages\package\manifest.v3
```

Any time this information is changed for the package, make sure you pick up the new file and check it into Clear Case at the root level of the package.

The manifest.rel file is generated when the build creates the package deployment Zip file.

Testing Scripts

Check testing scripts into a folder called resources under the PSFT_IS_TestScripts project in Clear Case. The testing scripts are checked in along with the stubbed out code needed for functional testing.

See [Chapter 4, “Understanding Development Conventions,” Testing, page 37](#).

Save all of the scripts under the resources folder in Clear Case with this folder structure.

```
Resources
  <PackageName>
    <Grouping(optional)>
      <Flow Service Name>
```


APPENDIX B

Family

This appendix lists defined families.

Defined Families

These families are defined:

- Core
- AddressBook
- Business Unit
- Company
- Customer
- Financials
- FixedAssets
- Home Builder
- InstalledBase
- Inventory
- Manufacturing
- Procurement
- Sales
- ServiceAgreement
- ServiceOrder
- Supplier
- UserDefined

APPENDIX C

PSFT_XrefAndSoftCoding Services

This appendix discusses the PSFT_XrefAndSoftCoding services.

PSFT_XRefAndSoftCoding.XRef:createCodeXReference

This service creates an entry in the Codes table.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|--|
| ApplicationID | String | Scope of the data. |
| ObjectType | String | Type of the data object. |
| NativeCode | String | Value of the code in the target enterprise system. |
| CanonicalCode | String | Canonical value of the code in the source enterprise system. |
| LatchClosed | String | The default is Y. Currently not used. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|--|
| Status | String | Returns TRUE if the entry was successfully created; otherwise, FALSE is returned. Contains the error message in case of a SQL exception. |

PSFT_XRefAndSoftCoding.XRef:createKeyXReference

This service creates an entry in the Keys table.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| ApplicationID | String | Scope of the data. |
| ObjectType | String | Type of the data object. |
| NativeKey | String | Value of the key in the target enterprise system. |
| CanonicalKey | String | Canonical value of the key in the source enterprise system. |
| LatchClosed | String | The default is Y. Currently not used. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|--|
| Status | String | Returns TRUE if the entry was successfully created; otherwise, FALSE is returned. Contains the error message in case of a SQL exception. |

PSFT_XRefAndSoftCoding.XRef:deleteCodeXReference

This service deletes the code identified by the ApplicationID, ObjectType, CanonicalCode from the Codes table. The ApplicationID, ObjectType, CanonicalCode uniquely identifies an entry in the Codes table.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|--|
| ApplicationID | String | Scope of the data. |
| ObjectType | String | Type of the data object. |
| CanonicalCode | String | Canonical value of the code in the source enterprise system. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|--|
| Status | String | Returns TRUE if the entry was successfully deleted; otherwise, FALSE is returned. Contains the error message in case of a SQL exception. |

PSFT_XRefAndSoftCoding.XRef:deleteKeyXReference

This service deletes the key identified by the ApplicationID, ObjectType, CanonicalKey from the Keys table. The ApplicationID, ObjectType, CanonicalKey uniquely identifies an entry in the Keys table.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| ApplicationID | String | Scope of the data. |
| ObjectType | String | Type of the data object. |
| CanonicalKey | String | Canonical value of the key in the source enterprise system. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|--|
| Status | String | Returns TRUE if the entry was successfully deleted; otherwise, FALSE is returned. Contains the error message in case of a SQL exception. |

PSFT_XRefAndSoftCoding.XRef:getCanonicalCode

This service returns the CanonicalCode for the ApplicationID, ObjectType, NativeCode supplied.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|--------------------|
| ApplicationID | String | Scope of the data. |

| Variable Name | Type | Description |
|---------------|--------|--|
| ObjectType | String | Type of the data object. |
| NativeCode | String | Value of the code in the target enterprise system. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| CanonicalCode | String | Canonical value of the code in the source enterprise system. |
| Status | String | Returns TRUE if the CanonicalCode was found in the Codes table; otherwise FALSE is returned. Contains the error message in case of a SQL exception. |

PSFT_XRefAndSoftCoding.XRef:getCanonicalKey

This service returns CanonicalKey for the ApplicationID, ObjectType, NativeKey supplied.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| ApplicationID | String | Scope of the data. |
| ObjectType | String | Type of the data object. |
| NativeKey | String | Value of the key in the target enterprise system. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| CanonicalKey | String | Canonical value of the key in the source enterprise system. |
| Status | String | Returns TRUE if the CanonicalKey was found in the Keys table; otherwise FALSE is returned. Contains the error message in case of a SQL exception. |

PSFT_XRefAndSoftCoding.XRef:getCodeLatch

This service returns the code latch for the ApplicationID, ObjectType, CanonicalCode supplied.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|--|
| ApplicationID | String | Scope of the data. |
| ObjectType | String | Type of the data object. |
| CanonicalCode | String | Canonical value of the code in the source enterprise system. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|--|
| CodeLatch | String | Values are Y and N. The default is Y. |
| Status | String | Returns TRUE if the code latch was found in the Codes table; otherwise FALSE is returned. Contains the error message in case of a SQL exception. |

PSFT_XRefAndSoftCoding.XRef:getKeyLatch

This service returns the key latch for the ApplicationID, ObjectType, CanonicalKey supplied.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| ApplicationID | String | Scope of the data. |
| ObjectType | String | Type of the data object. |
| CanonicalKey | String | Canonical value of the key in the source enterprise system. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|--|
| KeyLatch | String | Values are Y and N. The default is Y. |
| Status | String | Returns TRUE if the key latch was found in the Keys table; otherwise FALSE is returned. Contains the error message in case of a SQL exception. |

PSFT_XRefAndSoftCoding.XRef:getNativeCode

This service returns the NativeCode for the ApplicationID, ObjectType, CanonicalCode supplied.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|--|
| ApplicationID | String | Scope of the data. |
| ObjectType | String | Type of the data object. |
| CanonicalCode | String | Canonical value of the code in the source enterprise system. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| NativeCode | String | Value of the code in the target enterprise system. |
| Status | String | Returns TRUE if the native code was found in the Codes table; otherwise FALSE is returned. Contains the error message in case of a SQL exception. |

PSFT_XRefAndSoftCoding.XRef:getNativeKey

This service returns the NativeKey for the ApplicationID, ObjectType, CanonicalKey supplied.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| ApplicationID | String | Scope of the data. |
| ObjectType | String | Type of the data object. |
| CanonicalKey | String | Canonical value of the key in the source enterprise system. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| NativeKey | String | Value of the key in the target enterprise system. |
| Status | String | Returns TRUE if the NativeKey was found in the Keys table; otherwise FALSE is returned. Contains the error message in case of a SQL exception |

PSFT_XRefAndSoftCoding.XRef:setCodeLatch

This service sets the code latch.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|--|
| ApplicationID | String | Scope of the data. |
| ObjectType | String | Type of the data objects |
| CanonicalCode | String | Canonical value of the code in the source enterprise system. |
| LatchClosed | String | Values are Y and N. The default is Y. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|--|
| NativeCode | String | Value of the code in the target enterprise system. |
| Status | String | Returns TRUE if the latch was successfully updated; otherwise FALSE is returned. Contains the error message in case of a SQL exception |

PSFT_XRefAndSoftCoding.XRef:setKeyLatch

This service sets the key latch.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| ApplicationID | String | Scope of the data. |
| ObjectType | String | Type of the data objects. |
| CanonicalKey | String | Canonical value of the key in the source enterprise system. |
| LatchClosed | String | Values are Y and N. The default is Y. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| NativeKey | String | Value of the key in the target enterprise system. |
| Status | String | Returns TRUE if the latch was successfully updated; otherwise FALSE is returned. Contains the error message in case of a SQL exception. |

PSFT_XRefAndSoftCoding.SoftCoding:getIntegrationOption

This service returns the value corresponding to the ID supplied from the Integration Options table.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|--|
| ID | String | Integration key assigned for the values. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| Value | String | Value associated with the integration key supplied. |
| Status | String | Returns TRUE if the value was found in the Integration Options table; otherwise FALSE is returned. Contains the error message in case of a SQL exception. |

PSFT_XRefAndSoftCoding.SoftCoding:getOptionalIntegrationOption

This service returns the value corresponding to the ID supplied from the Integration Options table. If the value retrieved is null, then the default value is returned.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| ID | String | Integration key assigned for the value |
| DefaultValue | String | The default value to use for the output value if the value returned for the ID is null. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| Value | String | Value associated with the integration key supplied. |
| Status | String | Returns TRUE if the value was found in the Integration Options table; otherwise FALSE is returned. Contains the error message in case of a SQL exception. |

PSFT_XRefAndSoftCoding.Utills:concatErrorMessage

This is a utility service that is used by PSFT_XRefAndSoftCoding.XRef services.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|---------------------------------------|
| ErrorMessage | String | Message containing the error details. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|--------------------------|--------|--|
| concatenatedErrorMessage | String | The value is the string error concatenated to the beginning of the input ErrorMessage. |

PSFT_XRefAndSoftCoding.Utls:createKeyDebugMessage

This is a utility service that is used by PSFT_XRefAndSoftCoding.SystemXRef:SystemXPIXRefTarget service.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| ApplicationID | String | Scope of the data. |
| ObjectType | String | Type of the data object. |
| CanonicalKey | String | Canonical value of the key in the source enterprise system. |
| NativeKey | String | Value of the key in the target enterprise system. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|-------------------------------------|
| LogMessage | String | Created for logging debug messages. |

PSFT_XRefAndSoftCoding.Utls:generateCanonicalID

This service generates a canonical ID if one is not supplied by the source enterprise system while creating cross-reference keys.

Input

This service has no inputs.

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| CanonicalID | String | Auto-generated canonical ID. |
| Error | String | Returns an error message if there is an error in generating the canonical ID. |

PSFT_XRefAndSoftCoding.Utills:getBrokerSettings

This service returns the settings of the broker that is connected to the Integration Server.

Input

This service has no inputs.

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|-----------------|--------|---|
| brokerHost | String | Host name : port number where the broker is running. |
| brokerName | String | Name of the broker. |
| clientGroupName | String | Name of the client group to which the broker clients belong. These set of broker clients are the clients running on the Integration Server. |
| useSSL | String | Either true or false depending on whether SSL is used. |
| certfile | String | Location of the certificate file if SSL is used. |
| password | String | Password for using the certificate file. |
| dname | String | |
| isEncrypted | String | True or false. |
| isConnected | String | Yes or no. |
| CLIENTPREFIX | String | Common PREFIX for all broker clients running on the Integration Server. |
| lastError | Object | Last error received while connecting to the broker. |

APPENDIX D

PSFT_Utils Services

This appendix discusses the PSFT_Utils services.

PSFT_Utils.Conversion:booleanToString

This service converts a Boolean value into a String value. The Boolean value input must be the value true or false (any value other than true will return false).

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|---------|----------------|
| in | Boolean | Boolean value. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| out | String | String value representing the Boolean value passed in, for example, "true". |

PSFT_Utils.Conversion:charToString

This service converts a Character value into a String value. The Character value input must be a single character.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|-----------|--|
| in | Character | Input value, for example, character value a. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|--|
| out | String | Output value, for example, string value a. |

PSFT_Utills.Conversion:dateToString

This service converts a Java Date value into a String value according to a specified pattern.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| in | Date | Input value, for example, date value 12/31/2004 0:00:00 MST. |
| pattern | String | String pattern the output should conform to, for example, yyyy-MM-dd. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| out | String | Output value, for example, string value 2004-12-31. |

PSFT_Utills.Conversion:dateToStringWithOffset

This service converts a Java Date value into a String value according to a specified pattern and provides an offset.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| in | Date | Input value, for example, date value 12/31/2004 0:00:00 MST. |
| pattern | String | String pattern the output should conform to, for example, yyyy-MM-dd. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| out | String | Output value, for example, string value 2004-12-31+0. |

PSFT_Utils.Conversion:doubleToLong

This service converts a Double value into a Long value. The Double value input is reduced to a whole number less than 2^{63} , (9,223,372,036,854,775,807) and greater than -2^{63} (-9,223,372,036,854,775,808), with no decimals or scientific notation.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| in | Double | Input value, for example, double value 1.0. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|------|--|
| out | Long | Output value, for example, long value 1. |

PSFT_Utils.Conversion:doubleToString

This service converts a Double value into a String value. The Double value input must be a whole or fractional number less than $1.7E+308$ (with 15 significant digits) and greater than $-1.7E+308$ (with 15 significant digits).

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| in | Double | Input value, for example, double value 1.0. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|--|
| out | String | Output value, for example, string value 1.0. |

PSFT_Utills.Conversion:doubleToStringNoDecimal

This service converts a Double value into a String value, but with no decimal point and decimal value (the Double value input is first converted to a Long; see double ToLong and longToString for details).

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| in | Double | Input value, for example, double value 1.0. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|--|
| out | String | Output value, for example, string value 1. |

PSFT_Utills.Conversion:intToString

This service converts an Integer value into a String value. The Integer value input must be a whole number less than 2^{31} , (2147483647) and greater than -2^{31} (-2147483648), with no decimals or scientific notation.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|---------|--|
| in | Integer | Input value, for example, integer value 1. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|--|
| out | String | Output value, for example, string value "1". |

PSFT_Utils.Conversion:longToString

This service converts a Long value into a String value. The Long value input must be a whole number less than 2^{63} , (9,223,372,036,854,775,807) and greater than -2^{63} (-9,223,372,036,854,775,808), with no decimals or scientific notation.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|------|---|
| in | Long | Input value, for example, long value 1. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|--|
| Long | String | Output value, for example, string value "1". |

PSFT_Utils.Conversion:stringToBoolean

This service converts a String value into a Boolean value. The String value input must be the value true or false (any value other than true will return false).

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| in | String | Input value, for example string value "true". |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|---------|--|
| out | Boolean | Output value, for example, Boolean value true. |

PSFT_Utills.Conversion:stringToChar

This service converts a String value into a Character value. The String value input must be one single character.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| in | String | Input value, for example, string value "a". |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|-----------|---|
| out | Character | Output value, for example, character value a. |

PSFT_Utills.Conversion:stringToDate

This service converts a String value into a Java Date value according to a specified pattern.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|------|--|
| in | in | Input value, for example, string value "2004-12-31". |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|------|---|
| out | Date | Output value, for example, date value "12/31/2004 0:00:00 MST". |

PSFT_Utils.Conversion:stringToDouble

This service converts a String value into a Double value. The String value input must be a whole or fractional number less than 1.7E+308 (with 15 significant digits) and greater than -1.7E+308 (with 15 significant digits).

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| in | String | Input value, for example, string value "1.0". |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|--|
| out | Double | Output value, for example, double value 1.0. |

PSFT_Utils.Conversion:stringToInt

This service converts a String value into an Integer value. The String value input must be a whole number less than 2³¹, (2147483647) and greater than -2³¹ (-2147483648), with no decimals or scientific notation.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| in | String | Input value, for example, string value "1". |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|---------|---|
| out | Integer | Output value, for example, integer value 1. |

PSFT_Utils.Conversion:stringToLong

This service converts a String value into a Long value. The String value input must be a whole number less than 2^{63} , (9,223,372,036,854,775,807) and greater than -2^{63} (-9,223,372,036,854,775,808), with no decimals or scientific notation.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|-------------|---|
| in | String Long | Input value, for example, string value "1". |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|------|--|
| out | Long | Output value, for example, long value 1. |

PSFT_Utils.Error:customizedHandleError

Before reading about this service, please read about the handleError service.

The customizedHandleError service is an example of a service you write yourself that is called from the handleError service if its actionCustomize integration option is set to TRUE. This example service has no functionality, but your service would handle application errors in some way. By default, the customServiceName integration option of the handleError service is set to PSFT_Utills.error.customizedHandleError, which is the name of this example service. If the value of actionCustomize were set to TRUE and value of customServiceName were left to its default, this service would be called. However, you would set the value of customServiceName to the name of your custom service that you would create in your custom package.

PSFT_Utills.Error:getErrorListForE1

This service is used to take in errors and build an array or structure of errors that would be generated from the JD Edwards EnterpriseOne system.

Input: lastError (document)

Output: errorList (document list)

PSFT_Utills.Error:getErrorListForERP8

This service is used to take in errors and build an array or structure of errors that would be generated from JD Edwards ERP8.

Input: lastError (document)

Output: errorList (document list)

PSFT_Utills.Error:handleError

The handleError service will handle errors that may occur in any application flow. It is designed to handle these errors in an expected and consistent way. This service should be called from the catch lock of a try/catch sequence in the calling flow, and prior to its call the service pub.flow:getLastError should be called to provide this service necessary input (see testErrorHandler service).

The handleError service may log the error in a unique log file, it may send an email notifying a user of the error, it may even perform a customized operation. These actions are done depending on the values of specific Integration Options that had been set up prior (see Other Input).

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|----------------------|--------|--|
| error:flowName | String | Flow name/service that experienced the error (provided in the field service, output from service <code>pub.flow:getLastError</code> ; this is a required field), for example, <code>PSFT_Utills.Flow:dynamicServiceInvocation</code> . |
| error:severity | String | Severity of the error (this is an optional field), for example, 2. |
| error:description | String | The error message/description of the error (provided in the field error, output from service <code>pub.flow:getLastError</code> ; this is a required field). |
| error:keys:fieldName | String | Field name of a key relevant to the service that took the error to help with debugging (many keys may be supplied; this is an optional field), for example, <code>orderNumber</code> . |
| error:keys:value | String | Value of a key specified by the field name, relevant to the service that took the error to help with debugging (many keys may be supplied; this is an optional field), for example, 1000. |

Other Input

Other input is provided to this service in the form of XPI Integration Options. Using the XPI Configuration Editor, browse to this location:

Integration Options\Family\Utills\handleError.

These integration options and their default values are defined:

| Option | Default Value | Description |
|-----------------|---------------|---|
| actionLog | FALSE | Change to <i>TRUE</i> to begin logging any application error to a special application log (see <code>logDirectory</code> and <code>logFilename</code> options). |
| actionEmail | FALSE | Change to <i>TRUE</i> to begin sending notification of any application error in an email message (see <code>emailTo</code> , <code>emailFrom</code> , <code>emailSubject</code> , and <code>emailHost</code>). |
| actionCustomize | FALSE | Change to <i>TRUE</i> to override the entire <code>handleError</code> service with some custom service you write yourself (see <code>customServiceName</code>). |

| Option | Default Value | Description |
|-------------------|---|--|
| logDirectory | .\logs | Keep this default value to write logs to the \IntegrationServer\logs directory, or change it to any other directory. |
| logFilename | flowError.log | Keep this default value to write logs to the flowErroryyyMMdd.log filename, or change it to any other filename. |
| emailTo | email@server.com | Change to the proper email address that is to receive notification of application errors in email messages. |
| emailFrom | email@server.com | Change to the proper email address that is to appear to send notification of application errors in email messages. |
| emailSubject | Error Occurred in Flow Service | Keep this default value as the subject title of notification of application errors in email messages, or change to any other subject title. |
| emailHost | mail.server.com | Change to the proper email host server that is to route the notification of application errors in email messages. |
| customServiceName | PSFT_Utills.error.customizedHandleError | Change to the name of your customized service that will override the entire handleError service. Your custom service would exist in your custom package. |

Output

This table describes the output of the service:

The logging of application errors to a special application log (see actionLog) and the notification of application errors in email messages (see actionEmail) is the only output from this service.

PSFT_Utills.Error:testErrorHandler

Before reading about this service, please read about the handleError service.

The testErrorHandler service is an example of how to call the handleError service. This example service has no functionality, but is the model for the applications that call the handleError service. The service demonstrates the importance of calling the handleError service from the catch block of a try/catch sequence, and to precede the call to handleError with a call to pub.flow:getLastError in order to provide the handleError service necessary input. The calling application would be coded as follows:

```
SEQUENCE (exit on Success)
SEQUENCE (exit on Failure) (Try block)
[services go here]
```

```

SEQUENCE (exit on Done) (Catch block)
getLastError
handleError

```

PSFT_Utils.File.Utils:closeFileWriter

This service closes a stream opened for file writing.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|--|
| fileWriter | Object | The Java object representing the stream opened for file writing. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|--|--------|---|
| The Java object representing the stream opened for file writing. | String | Value true (indicating success) or false (indicating no success). |

PSFT_Utils.File.Utils:deleteFile

This service deletes a specified file.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| filename | String | Filename of file, for example, filename.txt. |
| directory | String | Directory path to desired file, for example, C:\Temp. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| successFlag | String | Value true (indicating success) or false (indicating no success). |

PSFT_Utils.File.Utils:doesFileExist

This service renames a specified file to a specified new filename.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| filename | String | Filename of file, for example, filename.txt. |
| directory | String | Directory path to desired file, for example, C:\Temp. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|--|
| exists | String | Value true (indicating file exists) or false (indicating file does not exist). |

PSFT_Utils.File.Utils:fileType

This service indicates whether a specified input is a directory or a file, and whether the directory or file exists.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|--|
| filename | String | Complete path and filename of directory or file, for example, C:\Temp or C:\Temp\filename.txt. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|--|
| fileType | String | Value directory (indicating input is a directory) or file (indicating input is a file). |
| exists | String | Value true (indicating directory or file exists) or false (indicating directory or file does not exist). |
| successFlag | String | Value true (indicating success) or false (indicating no success). |

PSFT_Utils.File.Utils:flushFileWriter

This service flushes (empties) an stream opened for file writing.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|--|
| fileWriter | Object | The Java object representing the stream opened for file writing. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| successFlag | String | Value true (indicating success) or false (indicating no success). |

PSFT_Utils.File.Utils:getFileSeparator

This service returns the file separator appropriate for the operating system.

Input

This service has no inputs.

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|--|
| separator | String | The file separator appropriate for the operating system, for example, / (Unix) or \ (Windows). |

PSFT_Utils.File.Utils:getLineSeparator

This service returns the line (record) separator appropriate for the operating system.

Input

This service has no inputs.

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|--|
| separator | String | The line separator appropriate for the operating system, for example, the carriage return (Unix) or the carriage return line feed (Windows). |

PSFT_Utils.File.Utils:openFileWriter

This service opens a stream for file writing.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| filename | String | Complete path and desired filename of file to write, for example, C:\Temp\filename.txt. |
| overwriteFlag | String | Choice of overwrite or append. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| fileWriter | Object | The Java object representing the stream opened for file writing. |
| successFlag | String | Value true (indicating success) or false (indicating no success). |

PSFT_Utils.File.Utils:renameFile

This service renames a specified file to a specified new filename. If the file with the new file name already exists, then it appends Copy and then appends an index number to the file.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| filename | String | Current filename of file, for example, filename.txt. |
| directory | String | Directory path to desired file, for example, C:\Temp. |
| newFilename | String | Desired filename of file, for example, filename2.txt. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| successFlag | String | Value true (indicating success) or false (indicating no success). |

PSFT_Utils.File.Utils:writeFileWriter

This service writes file contents to a stream opened for file writing.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|--|
| fileWriter | Object | The Java object representing the stream opened for file writing. |
| fileContent | String | Contents of desired file. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| fileWriter | Object | The Java object representing the stream opened for file writing. |
| successFlag | String | Value true (indicating success) or false (indicating no success). |

PSFT_Utills.File:docToFileWithDate

This service writes the value of a document (iData object) to a file, with the current date made part of the filename (for example, fileNameyyyyMMdd.txt).

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|----------|---|
| directory | String | Directory path to desired file, for example, C:\Temp. |
| baseFilename | String | Desired filename of file, for example, filename. |
| extension | String | File extension of desired file, for example, txt. |
| document | Document | Contents of desired file. |
| overwriteFlag | | Choice of overwrite or append. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| successFlag | String | Value true (indicating success) or false (indicating no success). |

PSFT_Utills.File:stringToFileWithDate

This service writes the value of a string to a file, with the current date made part of the filename (for example, fileNameyyyyMMdd.txt).

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|----------|---|
| directory | String | Directory path to desired file, for example, C:\Temp. |
| baseFilename | String | Desired filename of file, for example, filename. |
| extension | String | File extension of desired file, for example, txt. |
| document | Document | Contents of desired file. |
| overwriteFlag | | Choice of overwrite or append. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| successFlag | String | Value true (indicating success) or false (indicating no success). |

PSFT_Utills.File:writeToFile

This service writes the value of a string to a file.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------------------|--------|---|
| filename | String | Complete path and desired filename of file to write, for example, C:\Temp\filename.txt. |
| fileContent | String | Contents of desired file. |
| Contents of desired file. | String | Choice of overwrite or append. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| successFlag | String | Value true (indicating success) or false (indicating no success). |

PSFT_Utills.FlatFiles.FormatServices: formatAndTrimInStrings

This service ensures that the field from a flat file string has been formatted into a value without enclosing quotes and without leading or trailing blanks. This format service is called once for each field of a flat file (see each field definition on the Flat File Structure tab of a flat file schema).

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|--|
| value | String | The value of the flat file string field. |
| direction | String | Set to convertToValues. Only convertToValues is valid. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|----------------|--------|--|
| formattedValue | String | - () The formatted value of the iData value field. |

PSFT_Utills.FlatFiles.FormatServices:formatInStrings

This service ensures that the field from a flat file string has been formatted into a value without enclosing quotes. This format service is called once for each field of a flat file (see each field definition on the Flat File Structure tab of a flat file schema).

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|--|
| value | String | The value of the flat file string field. |
| direction | String | Set to convertToValues. Only convertToValues is valid. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|----------------|--------|---|
| formattedValue | String | The formatted value of the iData value field. |

PSFT_Utills.FlatFiles.FormatServices:leftPadOutNumWithBlanks

This service ensures that the field from an iData value has been formatted into a string. This format service is called once for each field of a flat file (see each field definition on the Flat File Structure tab of a flat file schema).

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|--|
| value | String | The value of the iData value field. |
| direction | String | Set to convertToString. Only convertToString is valid. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|----------------|--------|--|
| formattedValue | String | The formatted value of the flat file string field. |

PSFT_Utills.FlatFiles.FormatServices: trimAndReturnNullIfEmpty

This service ensures that the field from a flat file string has been formatted into a value without leading or trailing blanks, and if an empty string results the field is set to null. This format service is called once for each field of a flat file (see each field definition on the Flat File Structure tab of a flat file schema).

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|--|
| value | String | The value of the flat file string field. |
| direction | String | Set to convertToValues. Only convertToValues is valid. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|----------------|--------|---|
| formattedValue | String | The formatted value of the iData value field. |

PSFT_Utills.FlatFiles.Utills:getNullValue

This service returns a string initialized to a null value (not an empty string value).

Input

This service has no inputs.

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|-------------|
| nullValue | String | A null. |

PSFT_Utills.Flow:dynamicServiceInvocation

This service invokes any service implemented in the Integration Server. It can be useful if the name of the service is not known until run time, based on user input or input from another service. The pipeline existing at the time of invocation is passed into the service being invoked, and the pipeline existing after execution of the service is then returned.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|-------------|--|
| serviceName | serviceName | The fully qualified name of the service to invoke (without the package name), for example, PSFT_Utills.Error: customizedHandleError. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| successFlag | String | Value true (indicating success) or false (indicating no success). |

PSFT_Utills.Format:formatBU

This service formats a string field containing an E1 Business Unit into a string padded on the left with blanks to a total of 12 characters in length.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| inString | String | The JD Edwards EnterpriseOne business unit, for example, M30. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---|--------|---|
| The JD Edwards EnterpriseOne business unit, for example, M30. | String | The formatted JD Edwards EnterpriseOne business unit, for example, M30. |

PSFT_Utills.Format:trimLength

This service ensures that the size of a string field will fit into a field of a specified length. The field is truncated on the right if necessary.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|--------------------------------|
| inString | String | Any value, for example, abcde. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|--|
| businessUnit | String | The same or truncated value, for example, abc. |

PSFT_Utills.Integrations:convert_ActionType_To_AUDIT_ACTN

This service converts a JD Edwards EnterpriseOne action type code into a PeopleSoft Enterprise audit action code.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| ActionType | String | A JD Edwards EnterpriseOne action type code, for example, 1, 2, or 3. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| AUDIT_ACTN | String | A PeopleSoft Enterprise audit action code, for example, A, C, or D. |

PSFT_Utills.Integrations:convert_AUDIT_ACTN_To_ActionType

This service converts a PeopleSoft Enterprise audit action code into a JD Edwards EnterpriseOne Action Type code.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| AUDIT_ACTN | String | A PeopleSoft Enterprise audit action code, for example, A, C, or D. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| ActionType | String | A JD Edwards EnterpriseOne action type code, for example, 1, 2, or 3. |

PSFT_Utills.Math:addInts

This service adds two numeric integer values (expressed as strings) and returns the sum. If either numeric value is passed in as null or an empty String, the value is assumed to be 0.

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|-----------------------------------|
| num1 | String | An integer value, for example, 1. |
| num1 | String | An integer value, for example, 2. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|----------------------------------|--------|--------------------------|
| An integer value, for example, 2 | String | The sum, for example, 3. |

PSFT_Utils.Time:AddTimes

This service adds two time values and returns the sum. For example, start with the current time (retrieved by the GetCurrentTime service) and add 60 seconds to it. The time values are in milliseconds (numeric long values, expressed as strings).

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| num1 | String | A time (long) value, for example, 111111111111. |
| num1 | String | A time (long) value, for example, 222222222222. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|-------------------------------------|
| value | String | The sum, for example, 333333333333. |

PSFT_Utils.Time:CompareTimes

This service compares two time values and returns whether the first time value is greater than, less than, or equal to the second time value. For example, compare a stored time with the current time (retrieved by the GetCurrentTime service). The service returns -1, 0, or 1. The time values are in milliseconds (numeric long values, expressed as strings).

Input

This table describes the input for the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| num1 | String | A time (long) value, for example, 111111111111. |
| num1 | String | A time (long) value, for example, 222222222222. |

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| value | String | -1 if timeValue1 comes before timeValue2. 0 if timeValue1 is the same time as timeValue2. 1 if timeValue1 comes after timeValue2. |

PSFT_Utils.Time:GetCurrentTime

This service returns the current time value. The time value is in milliseconds (numeric long value, expressed as a string).

Input

This service has no inputs.

Output

This table describes the output of the service:

| Variable Name | Type | Description |
|---------------|--------|---|
| value | String | Current time value, for example, 1089654031403. |

Glossary of JD Edwards EnterpriseOne Terms

| | |
|---------------------------------------|---|
| Accessor Methods/Assessors | Java methods to “get” and “set” the elements of a value object or other source file. |
| activity rule | The criteria by which an object progresses from one given point to the next in a flow. |
| add mode | A condition of a form that enables users to input data. |
| Advanced Planning Agent (APAg) | A JD Edwards EnterpriseOne tool that can be used to extract, transform, and load enterprise data. APAg supports access to data sources in the form of relational databases, flat file format, and other data or message encoding, such as XML. |
| alternate currency | <p>A currency that is different from the domestic currency (when dealing with a domestic-only transaction) or the domestic and foreign currency of a transaction.</p> <p>In JD Edwards EnterpriseOne Financial Management, alternate currency processing enables you to enter receipts and payments in a currency other than the one in which they were issued.</p> |
| Application Server | Software that provides the business logic for an application program in a distributed environment. The servers can be Oracle Application Server (OAS) or WebSphere Application Server (WAS). |
| as if processing | A process that enables you to view currency amounts as if they were entered in a currency different from the domestic and foreign currency of the transaction. |
| as of processing | A process that is run as of a specific point in time to summarize transactions up to that date. For example, you can run various JD Edwards EnterpriseOne reports as of a specific date to determine balances and amounts of accounts, units, and so on as of that date. |
| Auto Commit Transaction | A database connection through which all database operations are immediately written to the database. |
| back-to-back process | A process in JD Edwards EnterpriseOne Supply Management that contains the same keys that are used in another process. |
| batch processing | <p>A process of transferring records from a third-party system to JD Edwards EnterpriseOne.</p> <p>In JD Edwards EnterpriseOne Financial Management, batch processing enables you to transfer invoices and vouchers that are entered in a system other than JD Edwards EnterpriseOne to JD Edwards EnterpriseOne Accounts Receivable and JD Edwards EnterpriseOne Accounts Payable, respectively. In addition, you can transfer address book information, including customer and supplier records, to JD Edwards EnterpriseOne.</p> |
| batch server | A server that is designated for running batch processing requests. A batch server typically does not contain a database nor does it run interactive applications. |
| batch-of-one immediate | <p>A transaction method that enables a client application to perform work on a client workstation, then submit the work all at once to a server application for further processing. As a batch process is running on the server, the client application can continue performing other tasks.</p> <p>See also direct connect and store-and-forward.</p> |
| best practices | Non-mandatory guidelines that help the developer make better design decisions. |

| | |
|---|---|
| BPEL | Abbreviation for Business Process Execution Language, a standard web services orchestration language, which enables you to assemble discrete services into an end-to-end process flow. |
| BPEL PM | Abbreviation for Business Process Execution Language Process Manager, a comprehensive infrastructure for creating, deploying, and managing BPEL business processes. |
| Build Configuration File | Configurable settings in a text file that are used by a build program to generate ANT scripts. ANT is a software tool used for automating build processes. These scripts build published business services. |
| build engineer | An actor that is responsible for building, mastering, and packaging artifacts. Some build engineers are responsible for building application artifacts, and some are responsible for building foundation artifacts. |
| Build Program | A WIN32 executable that reads build configuration files and generates an ANT script for building published business services. |
| business analyst | An actor that determines if and why an EnterpriseOne business service needs to be developed. |
| business function | A named set of user-created, reusable business rules and logs that can be called through event rules. Business functions can run a transaction or a subset of a transaction (check inventory, issue work orders, and so on). Business functions also contain the application programming interfaces (APIs) that enable them to be called from a form, a database trigger, or a non-JD Edwards EnterpriseOne application. Business functions can be combined with other business functions, forms, event rules, and other components to make up an application. Business functions can be created through event rules or third-generation languages, such as C. Examples of business functions include Credit Check and Item Availability. |
| business function event rule | See named event rule (NER). |
| business service | EnterpriseOne business logic written in Java. A business service is a collection of one or more artifacts. Unless specified otherwise, a business service implies both a published business service and business service. |
| business service artifacts | Source files, descriptors, and so on that are managed for business service development and are needed for the business service build process. |
| business service class method | A method that accesses resources provided by the business service framework. |
| business service configuration files | Configuration files include, but are not limited to, interop.ini, JDBj.ini, and jdelog.properties. |
| business service cross reference | A key and value data pair used during orchestration. Collectively refers to both the code and the key cross reference in the WSG/XPI based system. |
| business service cross-reference utilities | Utility services installed in a BPEL/ESB environment that are used to access JD Edwards EnterpriseOne orchestration cross-reference data. |
| business service development environment | A framework needed by an integration developer to develop and manage business services. |
| business services development tool | Otherwise known as JDeveloper. |
| business service EnterpriseOne object | A collection of artifacts managed by EnterpriseOne LCM tools. Named and represented within EnterpriseOne LCM similarly to other EnterpriseOne objects like tables, views, forms, and so on. |

| | |
|--|---|
| business service framework | Parts of the business service foundation that are specifically for supporting business service development. |
| business service payload | An object that is passed between an enterprise server and a business services server. The business service payload contains the input to the business service when passed to the business services server. The business service payload contains the results from the business service when passed to the Enterprise Server. In the case of notifications, the return business service payload contains the acknowledgement. |
| business service property | Key value data pairs used to control the behavior or functionality of business services. |
| Business Service Property Admin Tool | An EnterpriseOne application for developers and administrators to manage business service property records. |
| business service property business service group | A classification for business service property at the business service level. This is generally a business service name. A business service level contains one or more business service property groups. Each business service property group may contain zero or more business service property records. |
| business service property categorization | A way to categorize business service properties. These properties are categorized by business service. |
| business service property key | A unique name that identifies the business service property globally in the system. |
| business service property utilities | A utility API used in business service development to access EnterpriseOne business service property data. |
| business service property value | A value for a business service property. |
| business service repository | A source management system, for example ClearCase, where business service artifacts and build files are stored. Or, a physical directory in network. |
| business services server | The physical machine where the business services are located. Business services are run on an application server instance. |
| business services source file or business service class | One type of business service artifact. A text file with the .java file type written to be compiled by a Java compiler. |
| business service value object template | The structural representation of a business service value object used in a C-business function. |
| Business Service Value Object Template Utility | A utility used to create a business service value object template from a business service value object. |
| business services server artifact | The object to be deployed to the business services server. |
| business view | A means for selecting specific columns from one or more JD Edwards EnterpriseOne application tables whose data is used in an application or report. A business view does not select specific rows, nor does it contain any actual data. It is strictly a view through which you can manipulate data. |
| central objects merge | A process that blends a customer's modifications to the objects in a current release with objects in a new release. |
| central server | A server that has been designated to contain the originally installed version of the software (central objects) for deployment to client computers. In a typical JD Edwards EnterpriseOne installation, the software is loaded on to one machine—the central server. Then, copies of the software are pushed out or downloaded to various workstations attached to it. That way, if the software is altered or corrupted through its use on workstations, an original set of objects (central objects) is always available on the central server. |

| | |
|---|--|
| charts | Tables of information in JD Edwards EnterpriseOne that appear on forms in the software. |
| check-in repository | A repository for developers to check in and check out business service artifacts. There are multiple check-in repositories. Each can be used for a different purpose (for example, development, production, testing, and so on). |
| connector | Component-based interoperability model that enables third-party applications and JD Edwards EnterpriseOne to share logic and data. The JD Edwards EnterpriseOne connector architecture includes Java and COM connectors. |
| contra/clearing account | A general ledger account in JD Edwards EnterpriseOne Financial Management that is used by the system to offset (balance) journal entries. For example, you can use a contra/clearing account to balance the entries created by allocations in JD Edwards EnterpriseOne Financial Management. |
| Control Table Workbench | An application that, during the Installation Workbench processing, runs the batch applications for the planned merges that update the data dictionary, user-defined codes, menus, and user override tables. |
| control tables merge | A process that blends a customer's modifications to the control tables with the data that accompanies a new release. |
| correlation data | The data used to tie HTTP responses with requests that consist of business service name and method. |
| cost assignment | The process in JD Edwards EnterpriseOne Advanced Cost Accounting of tracing or allocating resources to activities or cost objects. |
| cost component | In JD Edwards EnterpriseOne Manufacturing, an element of an item's cost (for example, material, labor, or overhead). |
| credentials | A valid set of JD Edwards EnterpriseOne username/password/environment/role, EnterpriseOne session, or EnterpriseOne token. |
| Cross-reference utility services | Utility services installed in a BPEL/ESB environment that access EnterpriseOne cross-reference data. |
| cross segment edit | A logic statement that establishes the relationship between configured item segments. Cross segment edits are used to prevent ordering of configurations that cannot be produced. |
| currency restatement | The process of converting amounts from one currency into another currency, generally for reporting purposes. You can use the currency restatement process, for example, when many currencies must be restated into a single currency for consolidated reporting. |
| cXML | A protocol used to facilitate communication between business documents and procurement applications, and between e-commerce hubs and suppliers. |
| database credentials | A valid database username/password. |
| database server | A server in a local area network that maintains a database and performs searches for client computers. |
| Data Source Workbench | An application that, during the Installation Workbench process, copies all data sources that are defined in the installation plan from the Data Source Master and Table and Data Source Sizing tables in the Planner data source to the system-release number data source. It also updates the Data Source Plan detail record to reflect completion. |
| date pattern | A calendar that represents the beginning date for the fiscal year and the ending date for each period in that year in standard and 52-period accounting. |

| | |
|--|---|
| denominated-in currency | The company currency in which financial reports are based. |
| deployment artifacts | Artifacts that are needed for the deployment process, such as servers, ports, and such. |
| deployment server | A server that is used to install, maintain, and distribute software to one or more enterprise servers and client workstations. |
| detail information | Information that relates to individual lines in JD Edwards EnterpriseOne transactions (for example, voucher pay items and sales order detail lines). |
| direct connect | A transaction method in which a client application communicates interactively and directly with a server application. See also batch-of-one immediate and store-and-forward. |
| Do Not Translate (DNT) | A type of data source that must exist on the iSeries because of BLOB restrictions. |
| dual pricing | The process of providing prices for goods and services in two currencies. |
| duplicate published business services authorization records | Two published business services authorization records with the same user identification information and published business services identification information. |
| embedded application server instance | An OC4J instance started by and running wholly within JDeveloper. |
| edit code | A code that indicates how a specific value for a report or a form should appear or be formatted. The default edit codes that pertain to reporting require particular attention because they account for a substantial amount of information. |
| edit mode | A condition of a form that enables users to change data. |
| edit rule | A method used for formatting and validating user entries against a predefined rule or set of rules. |
| Electronic Data Interchange (EDI) | An interoperability model that enables paperless computer-to-computer exchange of business transactions between JD Edwards EnterpriseOne and third-party systems. Companies that use EDI must have translator software to convert data from the EDI standard format to the formats of their computer systems. |
| embedded event rule | An event rule that is specific to a particular table or application. Examples include form-to-form calls, hiding a field based on a processing option value, and calling a business function. Contrast with the business function event rule. |
| Employee Work Center | A central location for sending and receiving all JD Edwards EnterpriseOne messages (system and user generated), regardless of the originating application or user. Each user has a mailbox that contains workflow and other messages, including Active Messages. |
| enterprise server | A server that contains the database and the logic for JD Edwards EnterpriseOne. |
| Enterprise Service Bus (ESB) | Middleware infrastructure products or technologies based on web services standards that enable a service-oriented architecture using an event-driven and XML-based messaging framework (the bus). |
| EnterpriseOne administrator | An actor responsible for the EnterpriseOne administration system. |
| EnterpriseOne credentials | A user ID, password, environment, and role used to validate a user of EnterpriseOne. |
| EnterpriseOne object | A reusable piece of code that is used to build applications. Object types include tables, forms, business functions, data dictionary items, batch processes, business views, event rules, versions, data structures, and media objects. |

| | |
|---|---|
| EnterpriseOne development client | Historically called “fat client,” a collection of installed EnterpriseOne components required to develop EnterpriseOne artifacts, including the Microsoft Windows client and design tools. |
| EnterpriseOne extension | A JDeveloper component (plug-in) specific to EnterpriseOne. A JDeveloper wizard is a specific example of an extension. |
| EnterpriseOne process | A software process that enables JD Edwards EnterpriseOne clients and servers to handle processing requests and run transactions. A client runs one process, and servers can have multiple instances of a process. JD Edwards EnterpriseOne processes can also be dedicated to specific tasks (for example, workflow messages and data replication) to ensure that critical processes don’t have to wait if the server is particularly busy. |
| EnterpriseOne resource | Any EnterpriseOne table, metadata, business function, dictionary information, or other information restricted to authorized users. |
| Environment Workbench | An application that, during the Installation Workbench process, copies the environment information and Object Configuration Manager tables for each environment from the Planner data source to the system-release number data source. It also updates the Environment Plan detail record to reflect completion. |
| escalation monitor | A batch process that monitors pending requests or activities and restarts or forwards them to the next step or user after they have been inactive for a specified amount of time. |
| event rule | A logic statement that instructs the system to perform one or more operations based on an activity that can occur in a specific application, such as entering a form or exiting a field. |
| explicit transaction | Transaction used by a business service developer to explicitly control the type (auto or manual) and the scope of transaction boundaries within a business service. |
| exposed method or value object | Published business service source files or parts of published business service source files that are part of the published interface. These are part of the contract with the customer. |
| facility | An entity within a business for which you want to track costs. For example, a facility might be a warehouse location, job, project, work center, or branch/plant. A facility is sometimes referred to as a “business unit.” |
| fast path | A command prompt that enables the user to move quickly among menus and applications by using specific commands. |
| file server | A server that stores files to be accessed by other computers on the network. Unlike a disk server, which appears to the user as a remote disk drive, a file server is a sophisticated device that not only stores files, but also manages them and maintains order as network users request files and make changes to these files. |
| final mode | The report processing mode of a processing mode of a program that updates or creates data records. |
| foundation | A framework that must be accessible for execution of business services at runtime. This includes, but is not limited to, the Java Connector and JDBj. |
| FTP server | A server that responds to requests for files via file transfer protocol. |
| header information | Information at the beginning of a table or form. Header information is used to identify or provide control information for the group of records that follows. |
| HTTP Adapter | A generic set of services that are used to do the basic HTTP operations, such as GET, POST, PUT, DELETE, TRACE, HEAD, and OPTIONS with the provided URL. |

| | |
|--|---|
| instantiate | A Java term meaning “to create.” When a class is instantiated, a new instance is created. |
| integration developer | The user of the system who develops, runs, and debugs the EnterpriseOne business services. The integration developer uses the EnterpriseOne business services to develop these components. |
| integration point (IP) | The business logic in previous implementations of EnterpriseOne that exposes a document level interface. This type of logic used to be called XBPs. In EnterpriseOne 8.11, IPs are implemented in Web Services Gateway powered by webMethods. |
| integration server | A server that facilitates interaction between diverse operating systems and applications across internal and external networked computer systems. |
| integrity test | A process used to supplement a company’s internal balancing procedures by locating and reporting balancing problems and data inconsistencies. |
| interface table | See Z table. |
| internal method or value object | Business service source files or parts of business service source files that are not part of the published interface. These could be private or protected methods. These could be value objects not used in published methods. |
| interoperability model | A method for third-party systems to connect to or access JD Edwards EnterpriseOne. |
| in-your-face-error | In JD Edwards EnterpriseOne, a form-level property which, when enabled, causes the text of application errors to appear on the form. |
| IServer service | This internet server service resides on the web server and is used to speed up delivery of the Java class files from the database to the client. |
| jargon | An alternative data dictionary item description that JD Edwards EnterpriseOne appears based on the product code of the current object. |
| Java application server | A component-based server that resides in the middle-tier of a server-centric architecture. This server provides middleware services for security and state maintenance, along with data access and persistence. |
| JDBNET | A database driver that enables heterogeneous servers to access each other’s data. |
| JDEBASE Database Middleware | A JD Edwards EnterpriseOne proprietary database middleware package that provides platform-independent APIs, along with client-to-server access. |
| JDECallObject | An API used by business functions to invoke other business functions. |
| jde.ini | A JD Edwards EnterpriseOne file (or member for iSeries) that provides the runtime settings required for JD Edwards EnterpriseOne initialization. Specific versions of the file or member must reside on every machine running JD Edwards EnterpriseOne. This includes workstations and servers. |
| JDEIPC | Communications programming tools used by server code to regulate access to the same data in multiprocess environments, communicate and coordinate between processes, and create new processes. |
| jde.log | The main diagnostic log file of JD Edwards EnterpriseOne. This file is always located in the root directory on the primary drive and contains status and error messages from the startup and operation of JD Edwards EnterpriseOne. |
| JDENET | A JD Edwards EnterpriseOne proprietary communications middleware package. This package is a peer-to-peer, message-based, socket-based, multiprocess communications middleware solution. It handles client-to-server and server-to-server communications for all JD Edwards EnterpriseOne supported platforms. |
| JDeveloper Project | An artifact that JDeveloper uses to categorize and compile source files. |

| | |
|---|---|
| JDeveloper Workspace | An artifact that JDeveloper uses to organize project files. It contains one or more project files. |
| JMS Queue | A Java Messaging service queue used for point-to-point messaging. |
| listener service | A listener that listens for XML messages over HTTP. |
| local repository | A developer's local development environment that is used to store business service artifacts. |
| local standalone BPEL/ESB server | A standalone BPEL/ESB server that is not installed within an application server. |
| Location Workbench | An application that, during the Installation Workbench process, copies all locations that are defined in the installation plan from the Location Master table in the Planner data source to the system data source. |
| logic server | A server in a distributed network that provides the business logic for an application program. In a typical configuration, pristine objects are replicated on to the logic server from the central server. The logic server, in conjunction with workstations, actually performs the processing required when JD Edwards EnterpriseOne software runs. |
| MailMerge Workbench | An application that merges Microsoft Word 6.0 (or higher) word-processing documents with JD Edwards EnterpriseOne records to automatically print business documents. You can use MailMerge Workbench to print documents, such as form letters about verification of employment. |
| Manual Commit transaction | A database connection where all database operations delay writing to the database until a call to commit is made. |
| master business function (MBF) | An interactive master file that serves as a central location for adding, changing, and updating information in a database. Master business functions pass information between data entry forms and the appropriate tables. These master functions provide a common set of functions that contain all of the necessary default and editing rules for related programs. MBFs contain logic that ensures the integrity of adding, updating, and deleting information from databases. |
| master table | See published table. |
| matching document | A document associated with an original document to complete or change a transaction. For example, in JD Edwards EnterpriseOne Financial Management, a receipt is the matching document of an invoice, and a payment is the matching document of a voucher. |
| media storage object | Files that use one of the following naming conventions that are not organized into table format: Gxxx, xxxGT, or GTxxx. |
| message center | A central location for sending and receiving all JD Edwards EnterpriseOne messages (system and user generated), regardless of the originating application or user. |
| messaging adapter | An interoperability model that enables third-party systems to connect to JD Edwards EnterpriseOne to exchange information through the use of messaging queues. |
| messaging server | A server that handles messages that are sent for use by other programs using a messaging API. Messaging servers typically employ a middleware program to perform their functions. |
| Middle-Tier BPEL/ESB Server | A BPEL/ESB server that is installed within an application server. |
| Monitoring Application | An EnterpriseOne tool provided for an administrator to get statistical information for various EnterpriseOne servers, reset statistics, and set notifications. |

| | |
|---|---|
| named event rule (NER) | Encapsulated, reusable business logic created using event rules, rather than C programming. NERs are also called business function event rules. NERs can be reused in multiple places by multiple programs. This modularity lends itself to streamlining, reusability of code, and less work. |
| <i>nota fiscal</i> | In Brazil, a legal document that must accompany all commercial transactions for tax purposes and that must contain information required by tax regulations. |
| <i>nota fiscal factura</i> | In Brazil, a <i>nota fiscal</i> with invoice information. See also <i>nota fiscal</i> . |
| Object Configuration Manager (OCM) | In JD Edwards EnterpriseOne, the object request broker and control center for the runtime environment. OCM keeps track of the runtime locations for business functions, data, and batch applications. When one of these objects is called, OCM directs access to it using defaults and overrides for a given environment and user. |
| Object Librarian | A repository of all versions, applications, and business functions reusable in building applications. Object Librarian provides check-out and check-in capabilities for developers, and it controls the creation, modification, and use of JD Edwards EnterpriseOne objects. Object Librarian supports multiple environments (such as production and development) and enables objects to be easily moved from one environment to another. |
| Object Librarian merge | A process that blends any modifications to the Object Librarian in a previous release into the Object Librarian in a new release. |
| Open Data Access (ODA) | An interoperability model that enables you to use SQL statements to extract JD Edwards EnterpriseOne data for summarization and report generation. |
| Output Stream Access (OSA) | An interoperability model that enables you to set up an interface for JD Edwards EnterpriseOne to pass data to another software package, such as Microsoft Excel, for processing. |
| package | JD Edwards EnterpriseOne objects are installed to workstations in packages from the deployment server. A package can be compared to a bill of material or kit that indicates the necessary objects for that workstation and where on the deployment server the installation program can find them. It is point-in-time snapshot of the central objects on the deployment server. |
| package build | A software application that facilitates the deployment of software changes and new applications to existing users. Additionally, in JD Edwards EnterpriseOne, a package build can be a compiled version of the software. When you upgrade your version of the ERP software, for example, you are said to take a package build. Consider the following context: “Also, do not transfer business functions into the production path code until you are ready to deploy, because a global build of business functions done during a package build will automatically include the new functions.” The process of creating a package build is often referred to, as it is in this example, simply as “a package build.” |
| package location | The directory structure location for the package and its set of replicated objects. This is usually \\deployment server\release\path_code\package\package name. The subdirectories under this path are where the replicated objects for the package are placed. This is also referred to as where the package is built or stored. |
| Package Workbench | An application that, during the Installation Workbench process, transfers the package information tables from the Planner data source to the system-release number data source. It also updates the Package Plan detail record to reflect completion. |
| Pathcode Directory | The specific portion of the file system on the EnterpriseOne development client where EnterpriseOne development artifacts are stored. |

| | |
|--|---|
| patterns | General repeatable solutions to a commonly occurring problem in software design. For business service development, the focus is on the object relationships and interactions. For orchestrations, the focus is on the integration patterns (for example, synchronous and asynchronous request/response, publish, notify, and receive/reply). |
| planning family | A means of grouping end items whose similarity of design and manufacture facilitates being planned in aggregate. |
| preference profile | The ability to define default values for specified fields for a user-defined hierarchy of items, item groups, customers, and customer groups. |
| print server | The interface between a printer and a network that enables network clients to connect to the printer and send their print jobs to it. A print server can be a computer, separate hardware device, or even hardware that resides inside of the printer itself. |
| pristine environment | A JD Edwards EnterpriseOne environment used to test unaltered objects with JD Edwards EnterpriseOne demonstration data or for training classes. You must have this environment so that you can compare pristine objects that you modify. |
| processing option | A data structure that enables users to supply parameters that regulate the running of a batch program or report. For example, you can use processing options to specify default values for certain fields, to determine how information appears or is printed, to specify date ranges, to supply runtime values that regulate program execution, and so on. |
| production environment | A JD Edwards EnterpriseOne environment in which users operate EnterpriseOne software. |
| production-grade file server | A file server that has been quality assurance tested and commercialized and that is usually provided in conjunction with user support services. |
| Production Published Business Services Web Service | Published business services web service deployed to a production application server. |
| program temporary fix (PTF) | A representation of changes to JD Edwards EnterpriseOne software that your organization receives on magnetic tapes or disks. |
| project | In JD Edwards EnterpriseOne, a virtual container for objects being developed in Object Management Workbench. |
| promotion path | <p>The designated path for advancing objects or projects in a workflow. The following is the normal promotion cycle (path):</p> <p>11>21>26>28>38>01</p> <p>In this path, <i>11</i> equals new project pending review, <i>21</i> equals programming, <i>26</i> equals QA test/review, <i>28</i> equals QA test/review complete, <i>38</i> equals in production, <i>01</i> equals complete. During the normal project promotion cycle, developers check objects out of and into the development path code and then promote them to the prototype path code. The objects are then moved to the productions path code before declaring them complete.</p> |
| proxy server | A server that acts as a barrier between a workstation and the internet so that the enterprise can ensure security, administrative control, and caching service. |
| published business service | EnterpriseOne service level logic and interface. A classification of a published business service indicating the intention to be exposed to external (non-EnterpriseOne) systems. |
| published business service identification information | Information about a published business service used to determine relevant authorization records. Published business services + method name, published business services, or *ALL. |

| | |
|---|--|
| published business service web service | Published business services components packaged as J2EE Web Service (namely, a J2EE EAR file that contains business service classes, business service foundation, configuration files, and web service artifacts). |
| published table | Also called a master table, this is the central copy to be replicated to other machines. Residing on the publisher machine, the F98DRPUB table identifies all of the published tables and their associated publishers in the enterprise. |
| publisher | The server that is responsible for the published table. The F98DRPUB table identifies all of the published tables and their associated publishers in the enterprise. |
| pull replication | One of the JD Edwards EnterpriseOne methods for replicating data to individual workstations. Such machines are set up as pull subscribers using JD Edwards EnterpriseOne data replication tools. The only time that pull subscribers are notified of changes, updates, and deletions is when they request such information. The request is in the form of a message that is sent, usually at startup, from the pull subscriber to the server machine that stores the F98DRPCN table. |
| QBE | An abbreviation for query by example. In JD Edwards EnterpriseOne, the QBE line is the top line on a detail area that is used for filtering data. |
| real-time event | A message triggered from EnterpriseOne application logic that is intended for external systems to consume. |
| refresh | A function used to modify JD Edwards EnterpriseOne software, or subset of it, such as a table or business data, so that it functions at a new release or cumulative update level, such as B73.2 or B73.2.1. |
| replication server | A server that is responsible for replicating central objects to client machines. |
| Rt-Addressing | Unique data identifying a browser session that initiates the business services call request host/port user session. |
| rules | Mandatory guidelines that are not enforced by tooling, but must be followed in order to accomplish the desired results and to meet specified standards. |
| quote order | In JD Edwards Procurement and Subcontract Management, a request from a supplier for item and price information from which you can create a purchase order. In JD Edwards Sales Order Management, item and price information for a customer who has not yet committed to a sales order. |
| secure by default | A security model that assumes that a user does not have permission to execute an object unless there is a specific record indicating such permissions. |
| Secure Socket Layer (SSL) | A security protocol that provides communication privacy. SSL enables client and server applications to communicate in a way that is designed to prevent eavesdropping, tampering, and message forgery. |
| SEI implementation | A Java class that implements the methods that declare in a Service Endpoint Interface (SEI). |
| selection | Found on JD Edwards EnterpriseOne menus, a selection represents functions that you can access from a menu. To make a selection, type the associated number in the Selection field and press Enter. |
| serialize | The process of converting an object or data into a format for storage or transmission across a network connection link with the ability to reconstruct the original data or objects when needed. |
| Server Workbench | An application that, during the Installation Workbench process, copies the server configuration files from the Planner data source to the system-release number |

| | |
|--|--|
| | data source. The application also updates the Server Plan detail record to reflect completion. |
| Service Endpoint Interface (SEI) | A Java interface that declares the methods that a client can invoke on the service. |
| SOA | Abbreviation for Service Oriented Architecture. |
| soft coding | A coding technique that enables an administrator to manipulate site-specific variables that affect the execution of a given process. |
| source repository | A repository for HTTP adapter and listener service development environment artifacts. |
| spot rate | An exchange rate entered at the transaction level. This rate overrides the exchange rate that is set up between two currencies. |
| Specification merge | A merge that comprises three merges: Object Librarian merge, Versions List merge, and Central Objects merge. The merges blend customer modifications with data that accompanies a new release. |
| specification | A complete description of a JD Edwards EnterpriseOne object. Each object has its own specification, or name, which is used to build applications. |
| Specification Table Merge Workbench | An application that, during the Installation Workbench process, runs the batch applications that update the specification tables. |
| SSL Certificate | A special message signed by a certificate authority that contains the name of a user and that user's public key in such a way that anyone can "verify" that the message was signed by no one other than the certification authority and thereby develop trust in the user's public key. |
| store-and-forward | The mode of processing that enables users who are disconnected from a server to enter transactions and then later connect to the server to upload those transactions. |
| subscriber table | Table F98DRSUB, which is stored on the publisher server with the F98DRPUB table and identifies all of the subscriber machines for each published table. |
| superclass | An inheritance concept of the Java language where a class is an instance of something, but is also more specific. "Tree" might be the superclass of "Oak" and "Elm," for example. |
| supplemental data | <p>Any type of information that is not maintained in a master file. Supplemental data is usually additional information about employees, applicants, requisitions, and jobs (such as an employee's job skills, degrees, or foreign languages spoken). You can track virtually any type of information that your organization needs.</p> <p>For example, in addition to the data in the standard master tables (the Address Book Master, Customer Master, and Supplier Master tables), you can maintain other kinds of data in separate, generic databases. These generic databases enable a standard approach to entering and maintaining supplemental data across JD Edwards EnterpriseOne systems.</p> |
| table access management (TAM) | The JD Edwards EnterpriseOne component that handles the storage and retrieval of use-defined data. TAM stores information, such as data dictionary definitions; application and report specifications; event rules; table definitions; business function input parameters and library information; and data structure definitions for running applications, reports, and business functions. |
| Table Conversion Workbench | An interoperability model that enables the exchange of information between JD Edwards EnterpriseOne and third-party systems using non-JD Edwards EnterpriseOne tables. |

| | |
|--|--|
| table conversion | An interoperability model that enables the exchange of information between JD Edwards EnterpriseOne and third-party systems using non-JD Edwards EnterpriseOne tables. |
| table event rules | Logic that is attached to database triggers that runs whenever the action specified by the trigger occurs against the table. Although JD Edwards EnterpriseOne enables event rules to be attached to application events, this functionality is application specific. Table event rules provide embedded logic at the table level. |
| terminal server | A server that enables terminals, microcomputers, and other devices to connect to a network or host computer or to devices attached to that particular computer. |
| three-tier processing | The task of entering, reviewing and approving, and posting batches of transactions in JD Edwards EnterpriseOne. |
| three-way voucher match | In JD Edwards Procurement and Subcontract Management, the process of comparing receipt information to supplier's invoices to create vouchers. In a three-way match, you use the receipt records to create vouchers. |
| transaction processing (TP) monitor | A monitor that controls data transfer between local and remote terminals and the applications that originated them. TP monitors also protect data integrity in the distributed environment and may include programs that validate data and format terminal screens. |
| transaction processing method | A method related to the management of a manual commit transaction boundary (for example, start, commit, rollback, and cancel). |
| transaction set | An electronic business transaction (electronic data interchange standard document) made up of segments. |
| trigger | One of several events specific to data dictionary items. You can attach logic to a data dictionary item that the system processes automatically when the event occurs. |
| triggering event | A specific workflow event that requires special action or has defined consequences or resulting actions. |
| two-way authentication | An authentication mechanism in which both client and server authenticate themselves by providing the SSL certificates to each other. |
| two-way voucher match | In JD Edwards Procurement and Subcontract Management, the process of comparing purchase order detail lines to the suppliers' invoices to create vouchers. You do not record receipt information. |
| user identification information | User ID, role, or *public. |
| User Overrides merge | Adds new user override records into a customer's user override table. |
| value object | A specific type of source file that holds input or output data, much like a data structure passes data. Value objects can be exposed (used in a published business service) or internal, and input or output. They are comprised of simple and complex elements and accessories to those elements. |
| variance | <p>In JD Edwards Capital Asset Management, the difference between revenue generated by a piece of equipment and costs incurred by the equipment.</p> <p>In JD Edwards EnterpriseOne Project Costing and JD Edwards EnterpriseOne Manufacturing, the difference between two methods of costing the same item (for example, the difference between the frozen standard cost and the current cost is an engineering variance). Frozen standard costs come from the Cost Components table, and the current costs are calculated using the current bill of material, routing, and overhead rates.</p> |

| | |
|--|---|
| versioning a published business service | Adding additional functionality/interfaces to the published business services without modifying the existing functionality/interfaces. |
| Version List merge | The Versions List merge preserves any non-XJDE and non-ZJDE version specifications for objects that are valid in the new release, as well as their processing options data. |
| visual assist | Forms that can be invoked from a control via a trigger to assist the user in determining what data belongs in the control. |
| vocabulary override | An alternate description for a data dictionary item that appears on a specific JD Edwards EnterpriseOne form or report. |
| wchar_t | An internal type of a wide character. It is used for writing portable programs for international markets. |
| web application server | A web server that enables web applications to exchange data with the back-end systems and databases used in eBusiness transactions. |
| web server | A server that sends information as requested by a browser, using the TCP/IP set of protocols. A web server can do more than just coordination of requests from browsers; it can do anything a normal server can do, such as house applications or data. Any computer can be turned into a web server by installing server software and connecting the machine to the internet. |
| Web Service Description Language (WSDL) | An XML format for describing network services. |
| Web Service Inspection Language (WSIL) | An XML format for assisting in the inspection of a site for available services and a set of rules for how inspection-related information should be made. |
| web service proxy foundation | Foundation classes for web service proxy that must be included in a business service server artifact for web service consumption on WAS. |
| web service softcoding record | An XML document that contains values that are used to configure a web service proxy. This document identifies the endpoint and conditionally includes security information. |
| web service softcoding template | An XML document that provides the structure for a soft coded record. |
| Where clause | The portion of a database operation that specifies which records the database operation will affect. |
| Windows terminal server | A multiuser server that enables terminals and minimally configured computers to display Windows applications even if they are not capable of running Windows software themselves. All client processing is performed centrally at the Windows terminal server and only display, keystroke, and mouse commands are transmitted over the network to the client terminal device. |
| wizard | A type of JDeveloper extension used to walk the user through a series of steps. |
| workbench | A program that enables users to access a group of related programs from a single entry point. Typically, the programs that you access from a workbench are used to complete a large business process. For example, you use the JD Edwards EnterpriseOne Payroll Cycle Workbench (P07210) to access all of the programs that the system uses to process payroll, print payments, create payroll reports, create journal entries, and update payroll history. Examples of JD Edwards EnterpriseOne workbenches include Service Management Workbench (P90CD020), Line Scheduling Workbench (P3153), Planning Workbench (P13700), Auditor's Workbench (P09E115), and Payroll Cycle Workbench. |
| work day calendar | In JD Edwards EnterpriseOne Manufacturing, a calendar that is used in planning functions that consecutively lists only working days so that component and work order scheduling can be done based on the actual number of work days available. A work |

| | |
|--------------------------------------|---|
| | day calendar is sometimes referred to as planning calendar, manufacturing calendar, or shop floor calendar. |
| workflow | The automation of a business process, in whole or in part, during which documents, information, or tasks are passed from one participant to another for action, according to a set of procedural rules. |
| workgroup server | A server that usually contains subsets of data replicated from a master network server. A workgroup server does not perform application or batch processing. |
| XAPI events | A service that uses system calls to capture JD Edwards EnterpriseOne transactions as they occur and then calls third-party software, end users, and other JD Edwards EnterpriseOne systems that have requested notification when the specified transactions occur to return a response. |
| XML CallObject | An interoperability capability that enables you to call business functions. |
| XML Dispatch | An interoperability capability that provides a single point of entry for all XML documents coming into JD Edwards EnterpriseOne for responses. |
| XML List | An interoperability capability that enables you to request and receive JD Edwards EnterpriseOne database information in chunks. |
| XML Service | An interoperability capability that enables you to request events from one JD Edwards EnterpriseOne system and receive a response from another JD Edwards EnterpriseOne system. |
| XML Transaction | An interoperability capability that enables you to use a predefined transaction type to send information to or request information from JD Edwards EnterpriseOne. XML transaction uses interface table functionality. |
| XML Transaction Service (XTS) | Transforms an XML document that is not in the JD Edwards EnterpriseOne format into an XML document that can be processed by JD Edwards EnterpriseOne. XTS then transforms the response back to the request originator XML format. |
| Z event | A service that uses interface table functionality to capture JD Edwards EnterpriseOne transactions and provide notification to third-party software, end users, and other JD Edwards EnterpriseOne systems that have requested to be notified when certain transactions occur. |
| Z table | A working table where non-JD Edwards EnterpriseOne information can be stored and then processed into JD Edwards EnterpriseOne. Z tables also can be used to retrieve JD Edwards EnterpriseOne data. Z tables are also known as interface tables. |
| Z transaction | Third-party data that is properly formatted in interface tables for updating to the JD Edwards EnterpriseOne database. |

Index

A

- action type code, converting 79
- adapter services
 - naming 11
 - naming packages 6
 - versioning 24
- addInts service 80
- additional documentation x
- AddTimes service 81
- application fundamentals ix
- artifacts, deprecating 24
- audit action code, converting 80

B

- boolean value, converting to string 57
- booleanToString service 57
- broker settings, returning 56
- business units, formatting 78

C

- canonical ID, generating 55
- CanonicalCode, returning 47
- CanonicalKey, returning 48
- character value, converting to string 57
- charToString service 57
- child records, processing 31
- Clear Case
 - structure 39
- closeFileWriter service 68
- code latch
 - returning 49
 - setting 51
- Codes table
 - creating an entry 45
 - deleting entries 46
- comments, submitting xiv
- common fields xiv
- company identifier 25
- CompareTimes service 81
- component scoped interfaces 8
- concatErrorMessage service 54
- contact information xiv
- convert_ActionType_To_AUDIT_ACTN service 79

- convert_AUDIT_ACTN_To_ActionType service 80
- createCodeXReference service 45
- createKeyDebugMessage service 55
- createKeyXReference service 45
- cross-references xiii, 36
- current time, returning 82
- Customer Connection website x
- customizedHandleError service 64

D

- data dictionary
 - naming fields 18
 - naming flat file 17
- date value, converting to string 58
- dateToString service 58
- dateToStringWithOffset service 58
- defined families 43
- deleteCodeXReference service 46
- deleteFile service 68
- deleteKeyXReference service 47
- deployment packages, naming 5
- development conventions 27
- docToFileWithDate service 73
- document lists 7
- document types, naming 17
- documentation
 - printed x
 - related x
 - updates x
- documentation, creating for services 37
- documents 7
- documents, writing contents to a file 73
- doesFileExist service 69
- double value
 - converting to long value 59
 - converting to string value 59
- doubleToLong service 59
- doubleToString service 59
- doubleToStringNoDecimal service 60
- dynamicServiceInvocation service 78

E

- Enterprise to EnterpriseOne integrations, naming flows 19

EnterpriseOne Adapter Services package,
structure 40

ERP 8.0

retrieving errors from 65

versioning guidelines 24

errors

handling 65

retrieving from ERP 8.0 65

retrieving from JD Edwards

EnterpriseOne 65

retrieving text of 30

types of 28

exceptions

retrieving text of 30

returning to calling application 28

exit step 28

F

families, list of defined families 43

fields

described 7

for flat file data dictionaries 18

naming 18

file separator, returning 70

file writing

closing stream 68

emptying stream 70

opening stream 71

writing file contents 72

files

deleting 68

renaming 69, 72

verifying existence 69

fileType service 69

flat file data dictionary

naming 17

flat files 14

flow services

creating from XBPs 21

designing 32

naming 19

versioning 24

flushFileWriter service 70

folders, naming 5

formatAndTrimInStrings service 75

formatBU service 78

formatInStrings service 76

G

generateCanonicalID service 55

getBrokerSettings service 56

getCanonicalCode service 47

getCanonicalKey service 48

getCodeLatch service 49

GetCurrentTime service 82

getErrorListForE1 service 65

getErrorListForERP8 service 65

getIntegrationOption service 53

getKeyLatch service 49

getLastError service 30

getLineSeparator service 71

getNativeCode service 50

getNativeKey service 51

getNullValue service 77

getOptionalIntegrationOption service 53

H

handleError service 65

home page for packages 37

HTML documentation 37

I

iData field, formatting 76

implementation guides

ordering x

integers

adding 80

converting to strings 60

integration flow services, naming 19

integration options

creating 36

retrieving values 53

integrations

modifying 25

testing 38

interface documents

guidelines for creating 7

versioning 23

interfaces

policies 8

supporting past releases 10

intToString service 60

J

java date value, converting to string 58

JD Edwards EnterpriseOne, retrieving
errors from 65

K

- key latch
 - returning 49
 - setting 52
- Keys table
 - creating an entry 45
 - deleting entries 47

L

- leftPadOutNumWithBlanks service 76
- length of names 3
- line separator, returning 71
- long value, converting to string 61
- longToString service 61

M

- manifest.v3 file 40
- master business functions, interfaces for 9

N

- name lengths 3
- naming standards 3
- native code, returning 50
- native key, returning 51
- notes xiii

O

- openFileWriter service 71

P

- packages
 - home page for 37
 - naming 4
 - structure 14
- past releases, supporting 10
- path length 3
- PeopleCode, typographical
 - conventions xii
- PeopleSoft Enterprise to JD Edwards
 - EnterpriseOne integrations, naming
 - flows 19
- policies for interfaces 8
- prerequisites ix
- primary folders, naming 5
- printed documentation x
- PSFT_Utills services 57
- PSFT_XrefAndSoftCoding services 45
- public interfaces 8
- published interfaces 8

R

- realtime events, interfaces for 10
- record separator, returning 71
- related documentation x
- renameFile service 72

S

- schemas
 - folder structure 14
 - naming 17
- services
 - invoking 78
 - naming 12
 - versioning 24
- setCodeLatch service 51
- setKeyLatch service 52
- staging view 40
- standards 27
- Star Team
 - storing test scripts 37
- stepping and tracing 35
- stream
 - closing 68
 - emptying 70
 - opening 71
 - writing file to 72
- string value
 - converting to boolean value 61
 - converting to character value 62
 - converting to double value 63
 - converting to integer 63
 - converting to java date value 62
 - converting to long value 64
- strings
 - checking for empty strings 77
 - formatting 75, 76
 - initializing to null 77
 - truncating 79
 - writing contents to a file with the current
 - date 74
 - writing to a file 74
- stringToBoolean service 61
- stringToChar service 62
- stringToDate service 62
- stringToDouble service 63
- stringToFileWithDate service 74
- stringToInt service 63
- stringToLong service 64
- suggestions, submitting xiv

T

- test scripts
 - checking in 41
 - developing 37
- testErrorHandler service 67
- time values
 - adding 81
 - comparing 81
- tracing and stepping 35
- transaction boundaries 32
- transactions, designing 32
- transformation services, naming 13
- triggers, naming 18
- trimAndReturnNullIfEmpty service 77
- trimLength service 79
- try/catch block 29
- typographical conventions xii

U

- utility services, versioning 24

V

- versioning example 10
- views in Clear Case 40
- visual cues xii

W

- warnings xiii, 31
- writeFileWriter service 72
- writeToFile service 74
- WSDL service document, naming 21

X

- XBPs, converting to flow services 21
- Xe release, versioning guidelines 24
- XML namespaces, editing 20
- XML schema data types 8

Z

- zip file, for deployment 5