

---

# JD Edwards EnterpriseOne Tools

## 8.97 Development Tools: Event Rules 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

<b>About This Documentation Preface .....</b>	<b>vii</b>
JD Edwards EnterpriseOne Application Prerequisites.....	vii
Application Fundamentals.....	vii
Documentation Updates and Printed Documentation.....	viii
Obtaining Documentation Updates.....	viii
Downloading Documentation.....	viii
Additional Resources.....	viii
Typographical Conventions and Visual Cues.....	ix
Typographical Conventions.....	x
Visual Cues.....	x
Country, Region, and Industry Identifiers.....	xi
Currency Codes.....	xii
Comments and Suggestions.....	xii
Common Fields Used in Implementation Guides.....	xii

## Preface

<b>JD Edwards EnterpriseOne Tools: Event Rules Preface.....</b>	<b>xv</b>
JD Edwards EnterpriseOne Tools Fundamentals.....	xv

## Chapter 1

<b>Getting Started with JD Edwards EnterpriseOne Tools Development Tools: Event Rules.....</b>	<b>1</b>
JD Edwards EnterpriseOne Tools Development Tools: Event Rules Overview.....	1
JD Edwards EnterpriseOne Tools Development Tools: Event Rules Implementation.....	1
JD Edwards EnterpriseOne Tools Development Tools: Event Rules Implementation Steps.....	1

## Chapter 2

<b>Understanding Events, Event Rules, and Runtime Processing.....</b>	<b>3</b>
Events.....	3
Event Rules.....	3
Event Rules Fundamentals.....	3
Named Event Rules.....	4

Embedded Event Rules.....	4
Runtime Processing of Event Rules.....	4
Fundamentals of Runtime Processing of Event Rules.....	4
Runtime Data Structures.....	5
Form Flow.....	7

## Chapter 3

<b>Using Event Rules Design.....</b>	<b>23</b>
Understanding Event Rules Design.....	23
Understanding Event Rule Validation.....	25
Understanding If and While Statements.....	25
Understanding ER Consistency.....	25
Understanding ER Variables.....	25
Prerequisites.....	26
Working with Event Rules Design.....	26
Understanding Assignments.....	27
Understanding Event Rules Design Tool Bar Buttons.....	27
Displaying Event Information.....	27
Assigning a Value.....	28
Creating an If or a While Statement.....	28
Creating an ER Variable.....	28
Attaching a System Function to an Event.....	29
Attaching a Business Function to an Event.....	29

## Chapter 4

<b>Using BrowsER.....</b>	<b>31</b>
Understanding BrowsER.....	31
Working with BrowsER.....	31

## Chapter 5

<b>Debugging Event Rules.....</b>	<b>33</b>
Understanding Debugging.....	33
The Debugging Process.....	33
Debugging Strategies.....	34
Debug Logs.....	35
Debugging Event Rules.....	35
Understanding the Event Rules Debugger.....	35

Debugging an Application with the Event Rules Debugger.....40

**Glossary of JD Edwards EnterpriseOne Terms.....43**

**Index .....59**



# 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](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
<b>Bold</b>	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](mailto:PSOFT-INFODEV_US@ORACLE.COM).

---

## Common Fields Used in Implementation Guides

<b>Address Book Number</b>	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.
<b>As If Currency Code</b>	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.
<b>Batch Number</b>	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).
<b>Batch Date</b>	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.
<b>Batch Status</b>	<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.

<b>Branch/Plant</b>	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.
<b>Business Unit</b>	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.
<b>Category Code</b>	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.
<b>Company</b>	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.
<b>Currency Code</b>	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.
<b>Document Company</b>	<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>
<b>Document Number</b>	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.
<b>Document Type</b>	<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>
<b>Effective Date</b>	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: Event Rules Preface

This preface discusses Oracle's JD Edwards EnterpriseOne Development Tools 8.97 Guide: Event Rules.

---

## JD Edwards EnterpriseOne Tools Fundamentals

Additional information describing the setup and design of the JD Edwards EnterpriseOne Tools system resides in companion documentation that applies to many or all JD Edwards EnterpriseOne Tools. These companion guides contain information that applies specifically to JD Edwards EnterpriseOne Tools Event Rules.

- *JD Edwards EnterpriseOne Tools 8.97 System Administration Guide*
- *JD Edwards EnterpriseOne Tools 8.97 Object Management Workbench Guide*

### See Also

*JD Edwards EnterpriseOne Tools 8.97 System Administration Guide*, “Getting Started with JD Edwards EnterpriseOne Tools System Administration”

*JD Edwards EnterpriseOne Tools 8.97 Object Management Workbench Guide*, “Getting Started with JD Edwards EnterpriseOne OMW”



# CHAPTER 1

## Getting Started with JD Edwards EnterpriseOne Tools Development Tools: Event Rules

This chapter discusses:

- Oracle’s JD Edwards EnterpriseOne Tools Development Tools: Event Rules overview.
- JD Edwards EnterpriseOne Tools Development Tools: Event Rules implementation.

---

### JD Edwards EnterpriseOne Tools Development Tools: Event Rules Overview

JD Edwards EnterpriseOne Tools Development Tools: Event Rules are used to create or modify event rules (ER) in JD Edwards EnterpriseOne applications. Event rules are connected to certain runtime events and instruct runtime how to respond to the conditions you choose to define.

---

### JD Edwards EnterpriseOne Tools Development Tools: Event Rules Implementation

This section provides an overview of the steps that are required to implement JD Edwards EnterpriseOne Tools Development Tools: Event Rules.

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 Tools Development Tools: Event Rules Implementation Steps

This table lists the steps for JD Edwards EnterpriseOne Event Rules implementation.

Step	Reference
1. Configure Oracle’s JD Edwards EnterpriseOne Tools Object Management Workbench (OMW).	<i>JD Edwards EnterpriseOne Tools 8.97 Object Management Workbench Guide</i> , “Configuring JD Edwards EnterpriseOne OMW”

Step	Reference
2. Configure OMW user roles and allowed actions.	<i>JD Edwards EnterpriseOne Tools 8.97 Object Management Workbench Guide</i> , “Configuring User Roles and Allowed Actions”
3. Configure OMW functions.	<i>JD Edwards EnterpriseOne Tools 8.97 Object Management Workbench Guide</i> , “Configuring JD Edwards EnterpriseOne OMW Functions”
4. Configure OMW activity rules.	<i>JD Edwards EnterpriseOne Tools 8.97 Object Management Workbench Guide</i> , “Configuring Activity Rules”
5. Configure OMW save locations.	<i>JD Edwards EnterpriseOne Tools 8.97 Object Management Workbench Guide</i> , “Configuring Object Save Locations”
6. Set up default location and printers.	<i>JD Edwards EnterpriseOne Tools 8.97 Development Tools: Report Printing Administration Technologies Guide</i> , “Understanding Report Printing Administration Technologies”

## CHAPTER 2

# Understanding Events, Event Rules, and Runtime Processing

This chapter discusses:

- Events.
- Event rules.
- Runtime processing of event rules.

---

## Events

*Events* are activities that occur on a form, such as entering a form or exiting a field by using TAB. Events can be initiated by the user or the application. A single control might initiate multiple events. The system also initiates some events, such as **Last Grid Record Has Been Read**, when certain actions occur.

---

## Event Rules

This section discusses:

- Event rules
- Named event rules
- Embedded event rules

### Event Rules Fundamentals

*Event rules* (ER) are logic statements that you can create and attach to events. ER is initiated when events occur at runtime. Oracle's JD Edwards EnterpriseOne software supports two kinds of Event Rules: Named Event Rules and Embedded Event Rules.

You can attach multiple event rules to one event. The various kinds of event rules include:

- Conditional statements, such as If/Else/End If.
- While loops.
- Assignments.
- Calls to business functions.
- Form or report interconnections.

- Calls to system functions.
- Table I/O operations.

## Named Event Rules

A *named event rule* (NER) is a series of regular ER statements (such as assignments, business functions, system function calls, and so forth). A NER encapsulates the series of statements into one reusable component. You can call a NER the same way as calling a business function. Business functions implement customized business logic using C language; NERs implement customized business logic using event rule statements.

## Embedded Event Rules

In addition to NERs, the other kind of ER is called *embedded event rules*, or simply *event rules*. Embedded ER is specific to a particular table, interactive application, or batch application. Embedded ER for a table is called *table event rules* or *table triggers*. Embedded ER for an interactive application or batch application is called *application event rules*.

### Application Event Rules

You can add business logic that is specific to a particular application. Interactive applications connect ER using Oracle's JD Edwards EnterpriseOne Form Design Aid (FDA), while batch event rules use Oracle's JD Edwards EnterpriseOne Report Design Aid (RDA).

### Table Event Rules

You can create table-specific event rules, which are ER that you attach to a table using Table Design Event Rules. This logic runs whenever any JD Edwards EnterpriseOne application accesses that table *and* uses that ER. For example, to maintain referential integrity, you might attach ER to a master table that deletes all children when a parent is deleted. Any JD Edwards EnterpriseOne application that deletes information from that table does not need to have embedded parent/child logic, because that logic exists in the table.

---

**Note.** Be aware that this functionality applies *only* to JD Edwards EnterpriseOne applications. Other applications that access the same database table cannot and do not use these ERs.

---

---

## Runtime Processing of Event Rules

This section discusses:

- Runtime processing of event rules.
- Runtime data structures.
- Form flow.

## Fundamentals of Runtime Processing of Event Rules

*Runtime processing* refers to how, at runtime, the system evaluates various events (such as initializing a form, clicking a button, and using TAB to move between fields) and their attached ER. ER is attached to events, which in turn are attached to controls or forms.

FDA provides several different form types, each of which includes predefined fields and features that are specific to the form type. For example, a find/browse form automatically includes a Find menu option or tool bar button with appropriate functions attached to it. When users enter search text in a filter or query-by-example (QBE) field, and then click the Find button on the tool bar, the runtime engine processes logic to fetch a record.

To avoid generating unnecessary ER, you should understand the different field types and associated capabilities that characterize each form type.

## Runtime Data Structures

*Runtime data structures* are structures or blocks of memory that hold data when user is working with an application. You should know what is happening to each form at runtime. You should know what is in a runtime structure at a given event point in the runtime process.

The runtime system dynamically creates runtime data structures. For example, if a form contains hidden controls, the system allocates memory for those controls even though they are not visible on the form. When you pass processing option (PO) values in a form, the system allocates memory to store the PO value.

### Available Objects and Runtime Data Structures

A runtime data structure is made of a variety of objects on a form. An available object is represented by a two-character, alphabetical code that characterizes the source of data and determines how the object data is used in an interactive application at runtime. Available objects make up the runtime data structure for a form.

During runtime processing, the system stores data in memory in an internal data structure. Certain fields of the data structure temporarily store data during runtime. When no longer needed, the data is deleted so that the system can process another record.

In ER, you can access and modify available objects in order to implement business logic. For example, you can assign a value to a QBE field to set query criteria for the form.

This table lists the available objects:

Available Object Code	Description
BC	A column in the business view (BV). BCs for both the form view and the grid view appear in this list. The system fills these columns with values from the database when it performs a fetch. The system writes these values to the database during an add or update.
GC	A column in the grid. The row that the value references depends on which event is accessing the GC. During the fetch cycle, it is usually the selected row. In some circumstances, CG objects also denote a particular physical column in the grid instead of a value. An example is the <b>Set Grid Font</b> system function.
GB	The grid buffer. This buffer is one row of data that is independent of the lines that the system reads from the database and writes to the grid. The GB enables you to manipulate column data for a line that you want to insert or update without affecting the present state of the grid. You access the GB through an available GB object, which appears after the GC objects in the list of available objects in Event Rules Design. Each grid contains only one instance of each GB column.
FC	A control on the form. If the control is a database item, this field corresponds to a BC object. Furthermore, if the control is not a filter, the FC object represents the same value as the BC object, and changing one of these results in changing both.
FI	A value passed through a form interconnection. You access this object either to read values that are passed into the form or to set values to be passed back. These objects correspond to the elements of the form data structure.

Available Object Code	Description
PO	A value passed from a PO. These values are passed into the application when a user launches it. Any form in that application can access them. POs can either be entered by the user, or they can be set up in a particular version of an application.
QC	A cell from the QBE line in the grid. These objects represent the values in any QBE cell on the grid. They include wild cards, but do not include any comparison operators. Likewise, assignments to these objects can include wild cards, but not comparison operators. To set comparisons, you must use a system function.
HC	A hypercontrol item. A hypercontrol item is a menu item or a tool bar item.
VA	ER variables. These objects represent any variables that you set up in ER.
SV	System variables. These objects represent some environment variables that are accessible to ER.
SL	System literals. These objects represent some constant system values that are accessible to ER.
TP	Tab page object.
TK	A column in the table that contains the table ER.
CO	A constant, such as the return code for an error.
TV	Text variables.
RC	Report constants for a batch application.
RV	Report variables (batch application).
IC	An input column (table conversion).
OC	An output column (table conversion).

BC and FC share the same internal structure if an FC is associated with a database item; filter fields are an exception.

## Processing Available Objects

When an available object is changed through ER, these actions occur:

- The object in the internal runtime structure is changed.
- If the object is a form control or grid cell, row, or column, the screen is updated with the new value.

A BV form control shares the same value as the corresponding business view item. (Filter fields are an exception to this rule.) This means that:

- FC data and BC data are always identical.
- Whenever FC data is changed, BC items are changed to the same value.
- Whenever BC values are changed, the FC runtime values also changes to the same values. This change may not immediately reflected to the screen.

- On **Control is Exited** processing, the value entered into the form control is captured in both the BC and FC item for that control.

## Control is Exited Processing

**Control is Exited** processing includes these actions:

- The value in the control is saved to internal runtime structures.
- The **Control is Exited** event is processed.

If the value has changed since the previous time that the control was exited, these steps occur:

- The system processes the **Control Exited/Changed–Inline** event.
- The system processes the **Control Exited/Changed–Async** event.
- The system validates the value using edit rules defined for the DD item.
- The form control data is formatted using format rules defined for the DD item and displayed on the screen.

## Form Flow

Each form type has different properties and event flow. The system provides events for the forms so that you can insert custom logic. These events occur regardless of whether you add event rule logic for that event.

This example represents how values in the runtime structures are stored in memory compared to how they appear on the form. This example uses the find/browse form when it is called directly from a menu. The runtime engine processes events in a certain order. The next sections describe the typical events for the find/browse form and the order in which they are processed. This process flow can vary depending on specific user interaction and the event rule logic that you use.

## Pre-Dialog Is Initialized

These steps occur before the **Dialog is Initialized** event is processed and the form appears:

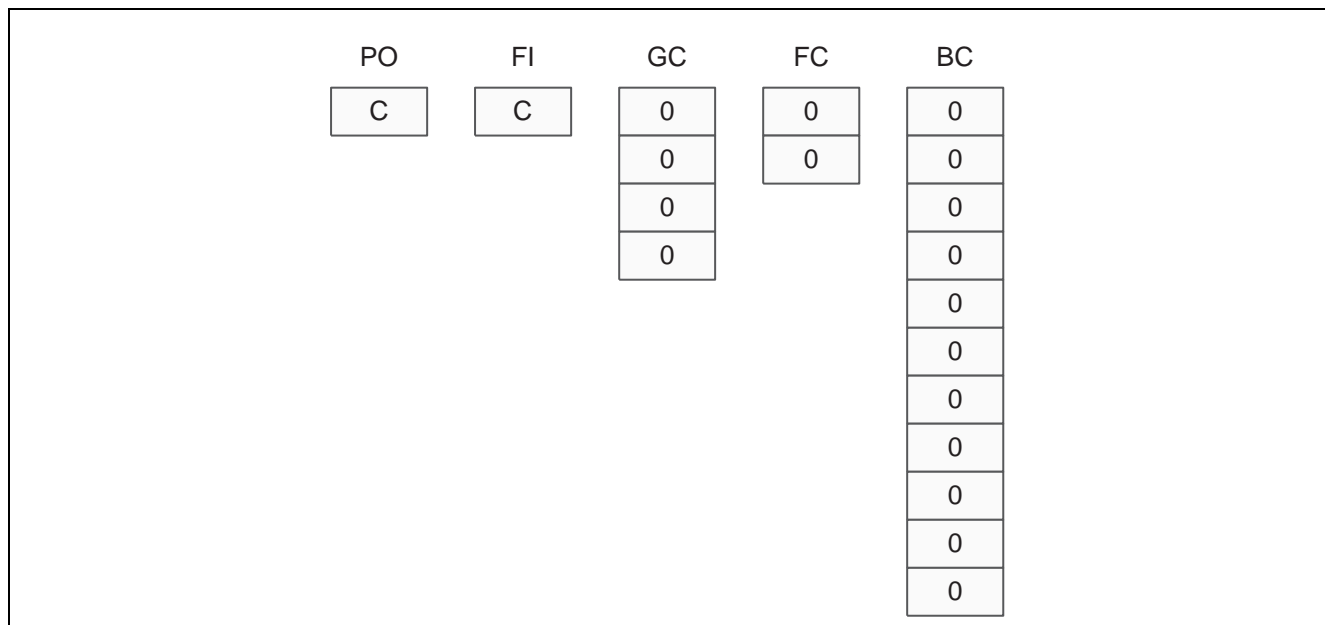
- Initialize runtime structures (or clear memory) as shown:
  - BC = null.
  - FC = null.
  - GC = null.
  - FI = Values passed from a calling form (if any).
  - PO = Values passed from processing options.
- Initialize form controls.
- Initialize error handling.
- Initialize static text.
- Initialize helps.
- Create tool bar.
- Load form interconnect data into corresponding BC columns and filter fields (if any exist).
- Initialize thread handling.

## Dialog Is Initialized

The system processes all event rule logic that is attached to the **Dialog is Initialized** event. When this event starts, the runtime structures contain these values:

- BC = Any FI values passed.
- FC = Any FI values passed.
- GC = null.
- FI = Values passed from a calling form (if any).
- PO = Values passed from POs.

This diagram illustrates the information in the runtime structures just before the system fires **Dialog is Initialized**:



Content of runtime structures before Dialog Is Initialized fires

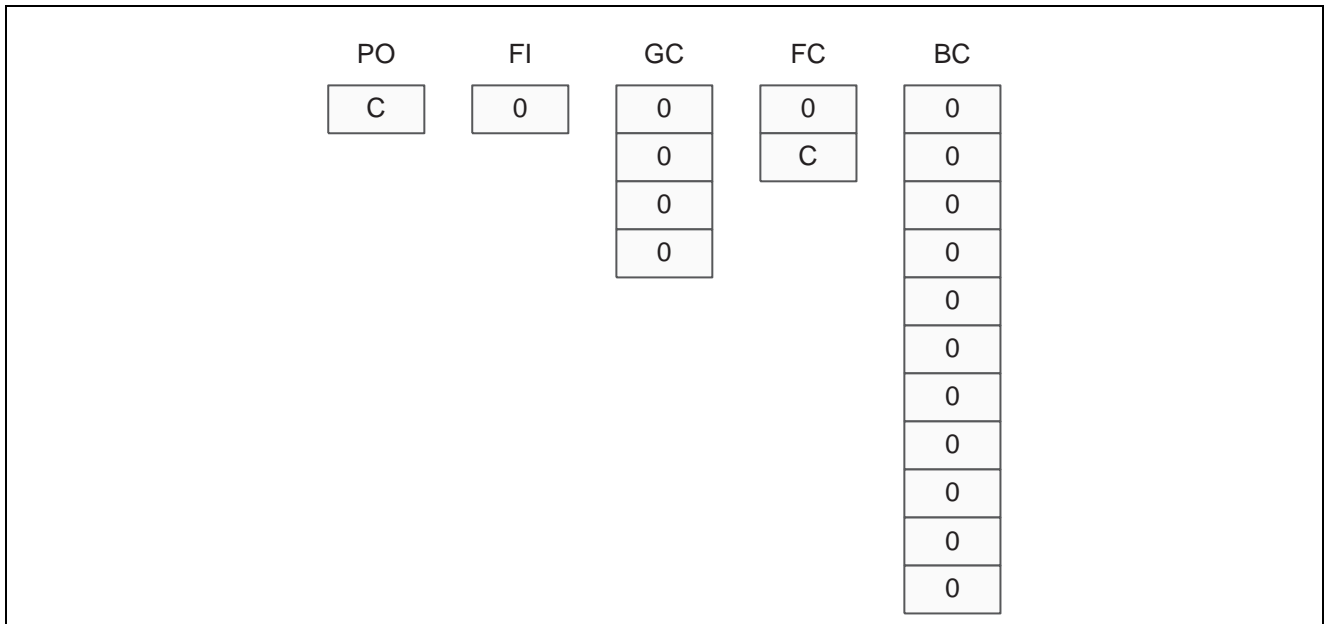
The **Dialog is Initialized** event can be used to initialize the form. After the **Dialog is Initialized** event is finished, runtime starts the **Post Dialog Is Initialized** process.

## Post Dialog Is Initialized

Before the system fires the **Post Dialog is Initialized** event, the runtime structures contain these values:

- BC = null (or values already passed in).
- FC = null (or values already passed in).
- GC = null (or values already passed in).
- FI = Values passed from a calling form (if any).
- PO = Values passed from POs.

This diagram illustrates the information in the runtime structures just before the system fires the **Post Dialog Is Initialized** event:



Content of runtime structures before Post Dialog Is Initialized fires

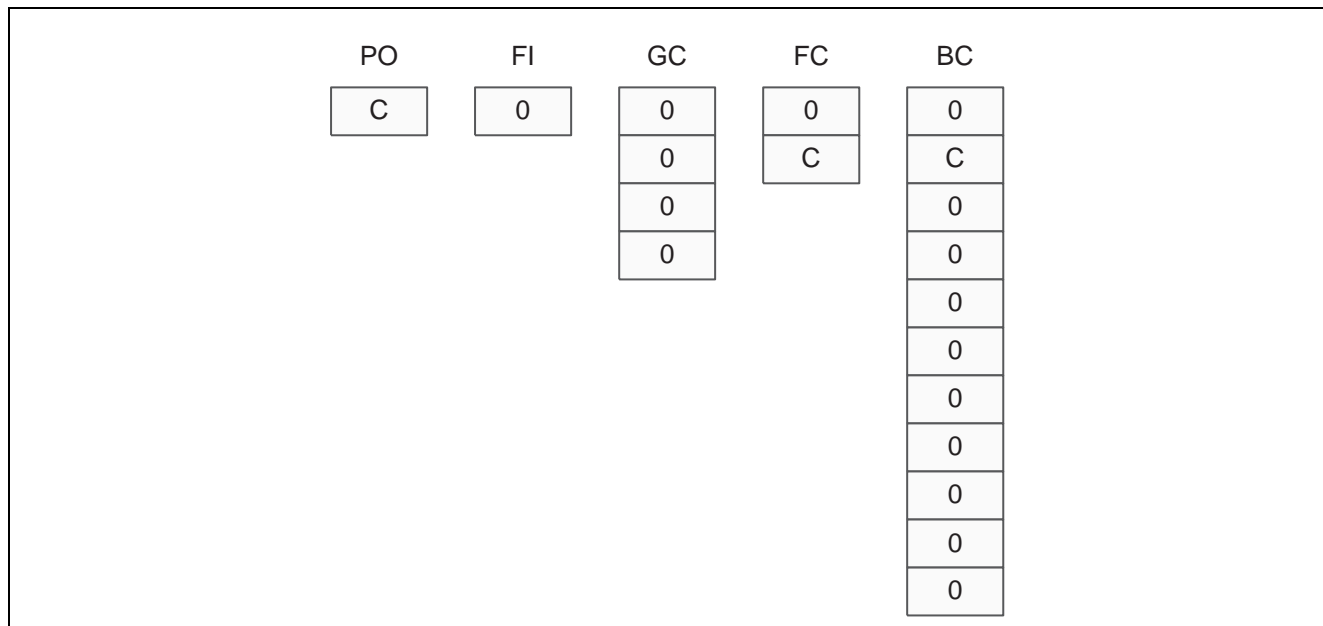
The **Post Dialog is Initialized** event is commonly used to perform these tasks:

- Load filter fields that will be used for the WHERE clause in the SQL SELECT statement.
- Load PO values into filter fields.
- Perform any one-time logic for the form, such as fetching a system date.

### Building SQL SELECT

After the user clicks Find, the system builds a SELECT statement with a WHERE clause. The SQL SELECT statement includes all columns in the BV. The WHERE clause includes any values in the QBE or filter fields. It can also contain values passed through Set Selection and Set Lower Limit system functions. The WHERE clause is then used to get all records that meet the criteria.

This diagram illustrates the information that appears in the runtime structures just before the system builds the SQL statement:



Content of runtime structures before SQL Select statement builds

## Fetching Records

Records are fetched one page at a time (unless page-at-a-time is disabled). The system processes each record fetched one by one and display it in the grid row.

## Page-at-a-Time Processing

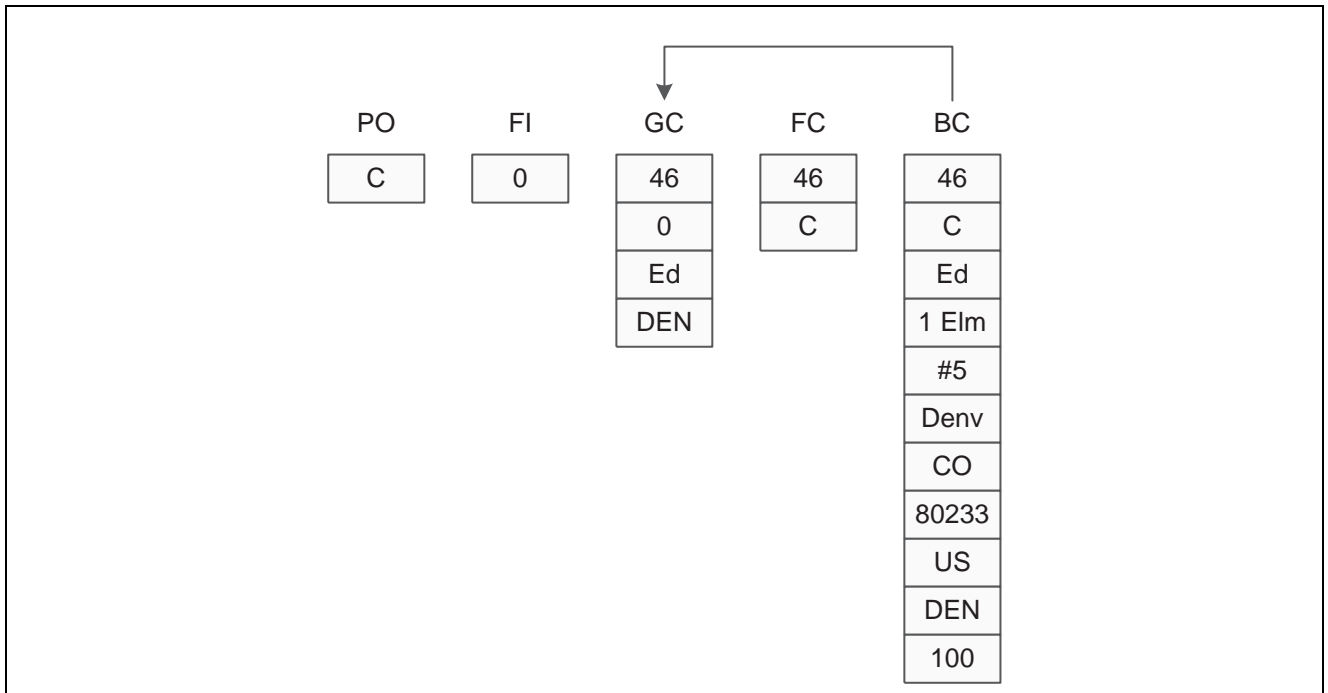
Page-at-a-time processing means that the system fetches only a single page worth of records to display. To see the next page of records, the user clicks the Next button. You can customize the page size for each grid in FDA. A system administrator can also set a global page size for all grids.

Typically, page-at-a-time processing improves performance and scalability. Although it can be disabled, the JD Edwards EnterpriseOne standards state that you should not disable it unless you have a valid business reason to do so.

## BC Assigned Database Values

After the system fetches each record from the database, it copies the database values to the BC items. Values from each marked column in the table appear in the BC runtime structure elements.

This diagram illustrates the information in the runtime structures when the system reads the first record:



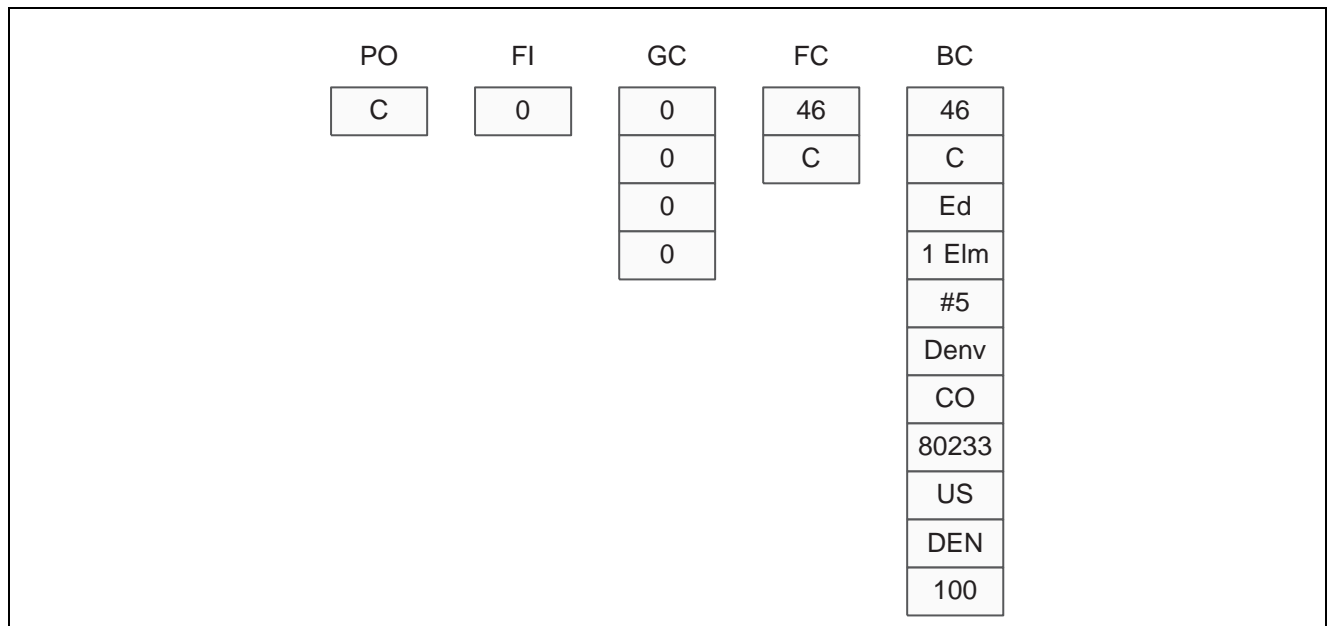
Content of runtime structures when first record read

### Grid Record Is Fetched

The engine then fires the **Grid Record is Fetched** event. At this point, the runtime structures have these values:

- BC = Values from the database (for the first record read).
- FC = Values from the database (if the fields are database fields).
- GC = null.
- FI = Values passed from a calling form (if any).
- PO = Values passed from POs.

This diagram illustrates the information in the runtime structures just before the system fires **Grid Record is Fetched**:



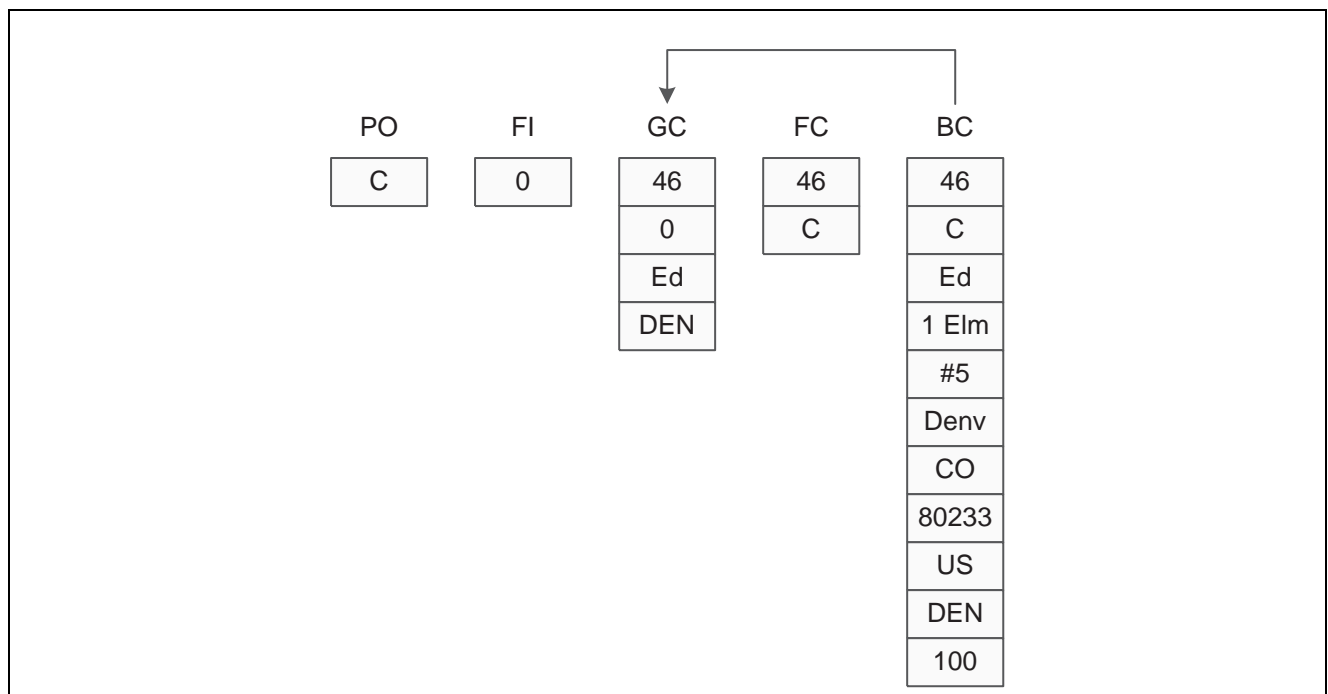
Content of runtime structures before Grid Record is Fetched fires

The **Grid Record is Fetched** event is commonly used to perform these actions:

- Calculate a value for a work field in the grid.
- Suppress a row from being written to the grid.

After the **Grid Rec Is Fetched** event fires, the BC values are copied into the GC runtime structure.

This diagram illustrates the information in the runtime structures when the system reads the first record:



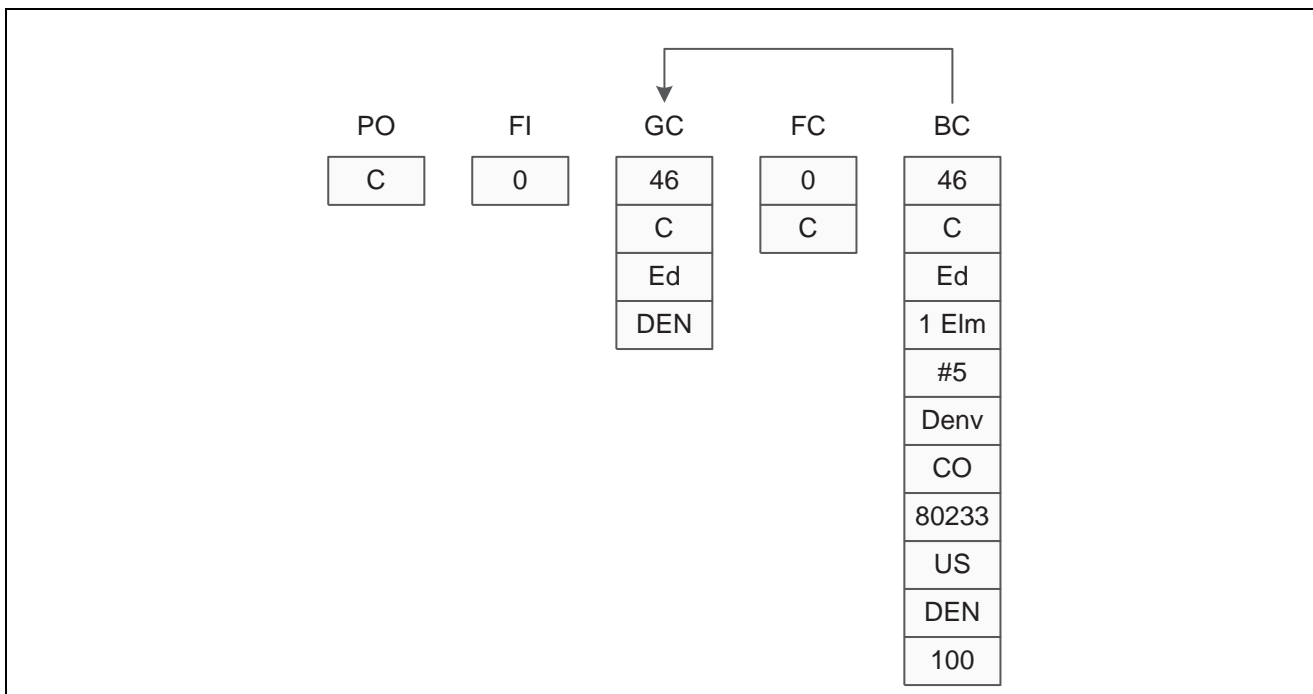
Content of runtime structures when first record is read

## Write Grid Line-Before

The engine then fires the **Write Grid Line-Before** event. At this point, the runtime structures have these values:

- BC = Values from the database (from the record just read).
- FC = Values from the database (if the fields are database fields).
- GC = Values from the database (from the previous read).
- FI = Values passed from a calling form (if any).
- PO = Values passed from POs.

This diagram illustrates the information in the runtime structures just before the system fires **Write Grid Line-Before**:



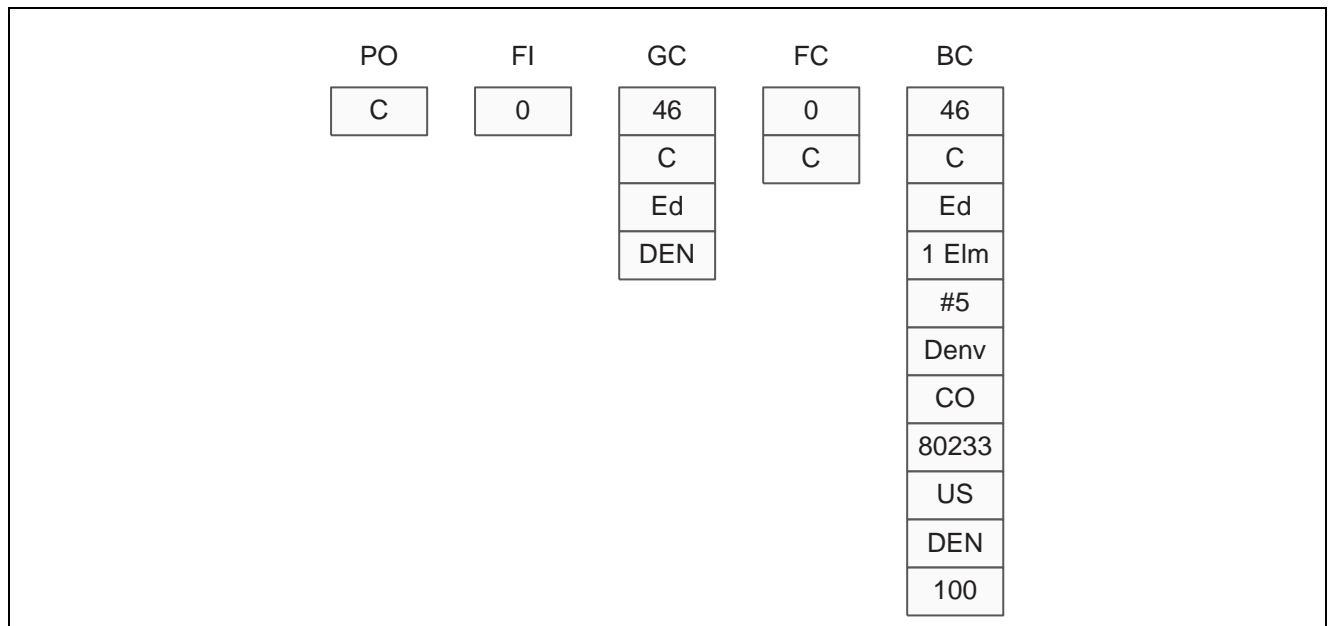
Content of runtime structures before Write Grid Line-Before fires

The **Write Grid Line-Before** event is commonly used to perform these tasks:

- Suppress a grid row from being written.
- Add logic before the user sees a row on the form.
- Change formatting of a grid column.
- Convert any grid value, such as unit of measure.
- Retrieve additional information for the grid row, such as a description, from tables that are not in the BV.

After the system processes **Write Grid Line-Before**, the GC elements, which now include the database values for the first record, are copied to the grid cells on the form.

This diagram illustrates the information that appears in the runtime structures now:



Content of runtime structures after Write Grid Line–Before fires

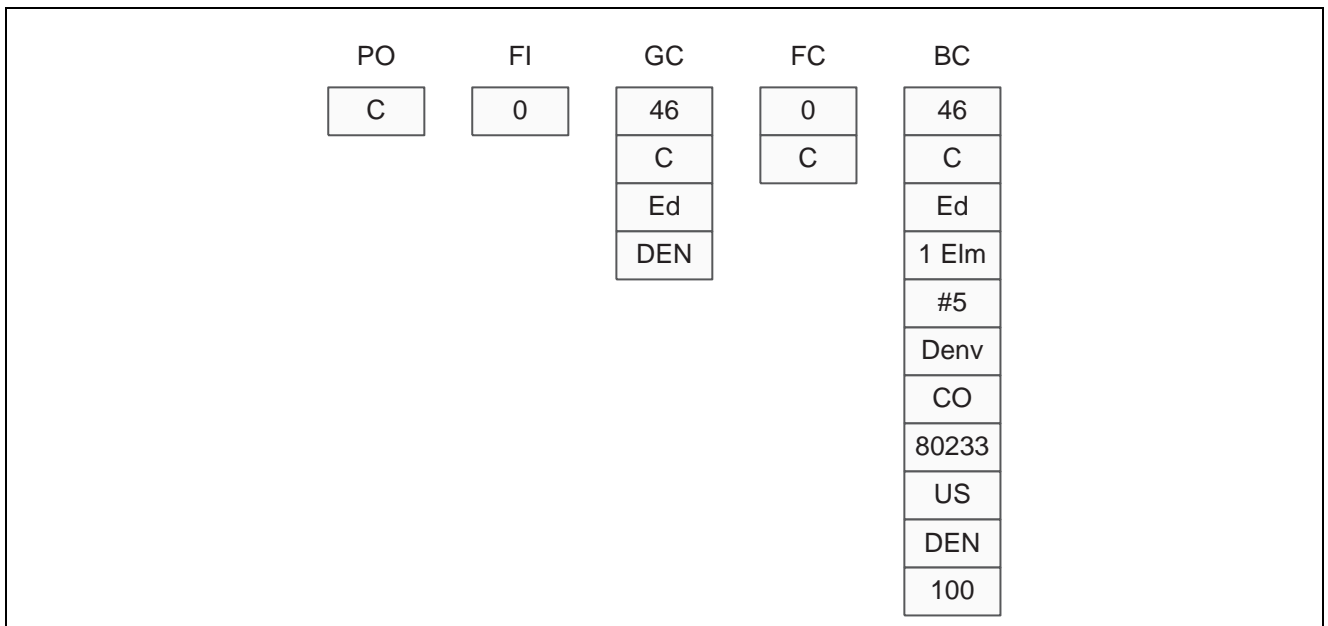
### Write Grid Line–After

The engine then fires the **Write Grid Line–After** event. At this point, the runtime structures have these values:

- BC = Values from the database (from the first record read).
- FC = Values from database (if the field is a database field).
- GC = Values from the database (from the first record read).
- FI = Values passed from a calling form (if any).
- PO = Values passed from POs.

The system displays the current record in the grid cells.

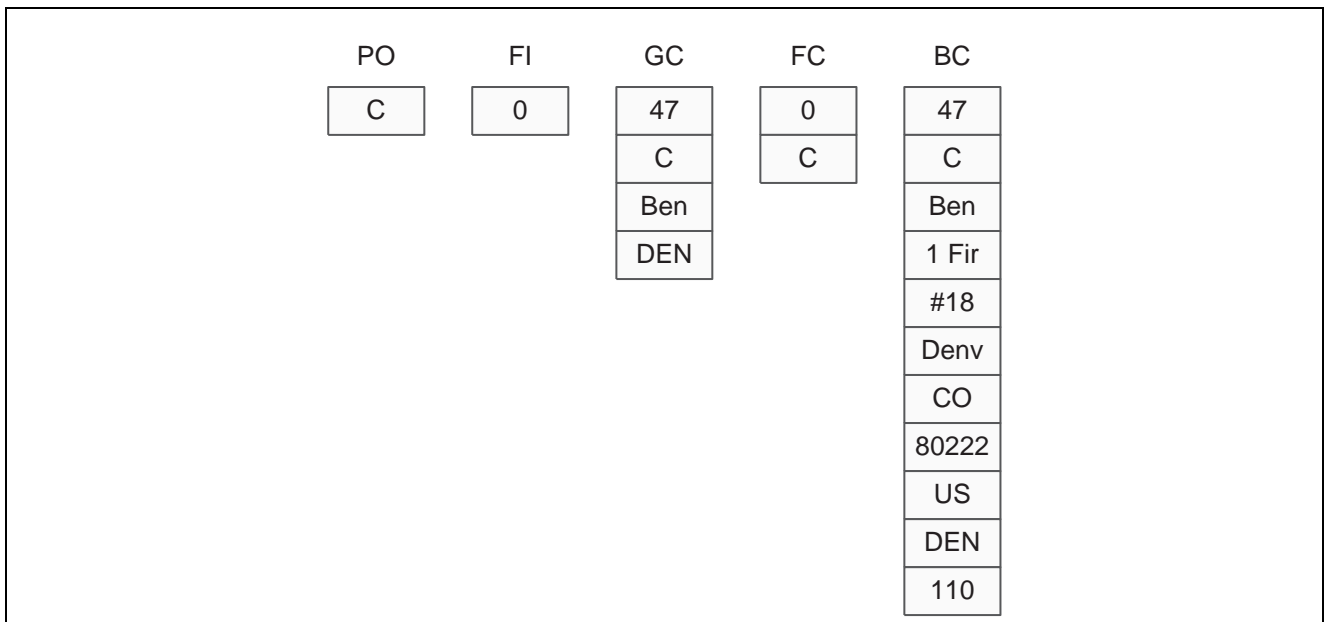
This diagram illustrates the information in the runtime structures just before the system fires **Write Grid Line–After**:



Content of runtime structures before Write Grid Line–After fires

You typically use the **Write Grid Line–After** event to add logic after the user sees a row on the form.

This diagram illustrates the information in the runtime structures after the system processes **Write Grid Line–After**:



Content of runtime structures after Write Grid Line After runs

The system continues to read records from the database and performs the same processing steps. When the system reads the next record, it performs these processing steps:

- Assign BC values from the database.
- Process **Grid Rec is Fetched** ER.
- Assign BC values to GC.
- Process **Write Grid Line–Before** ER.

- Display values in the grid row on the form
- Process **Write Grid Line–After** ER.

This process is repeated until there are no more records fetched.

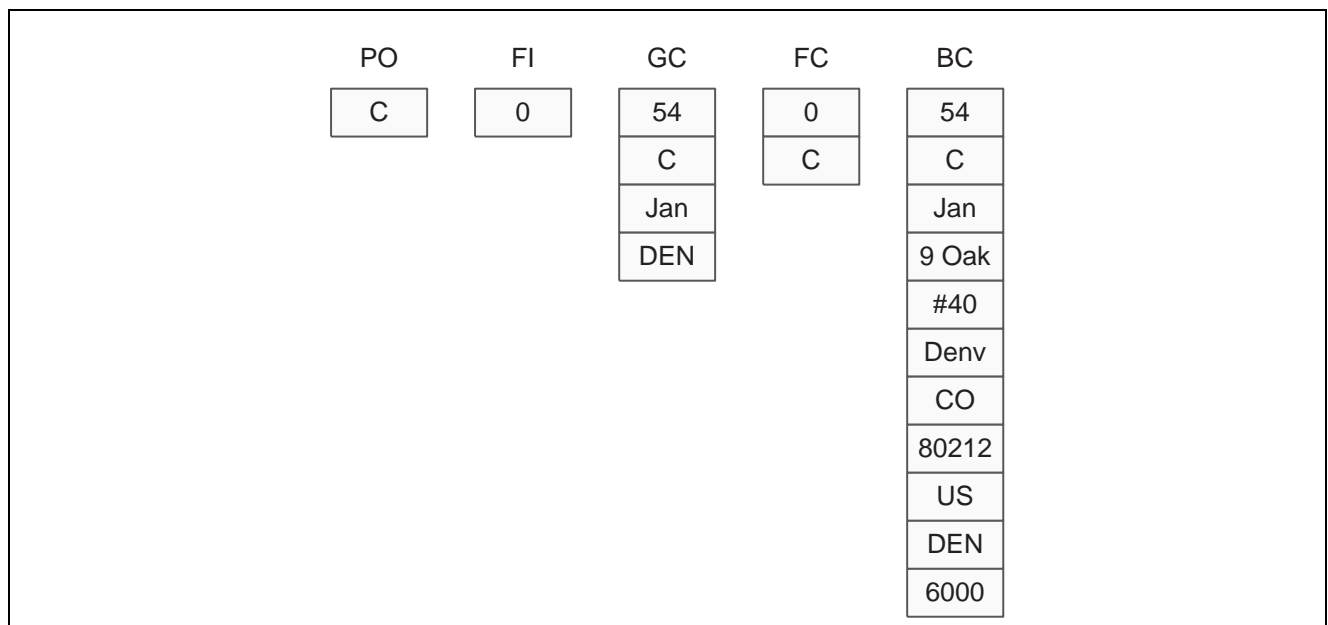
### Last Grid Record Has Been Read

When there are no more records fetched from the database, the engine fires **Last Grid Record Has Been Read** event. At this point, the runtime structures contain these values:

- BC = Values from the database (from the last record read).
- FC = Values from the database (if the field is a database field).
- GC = Values from the database (from the last record read).
- FI = Values passed from a calling form (if any).
- PO = Values passed from POs.

The GC values appear on the last grid row.

This diagram illustrates the information in the runtime structures just before the system runs **Last Grid Record Has Been Read**:



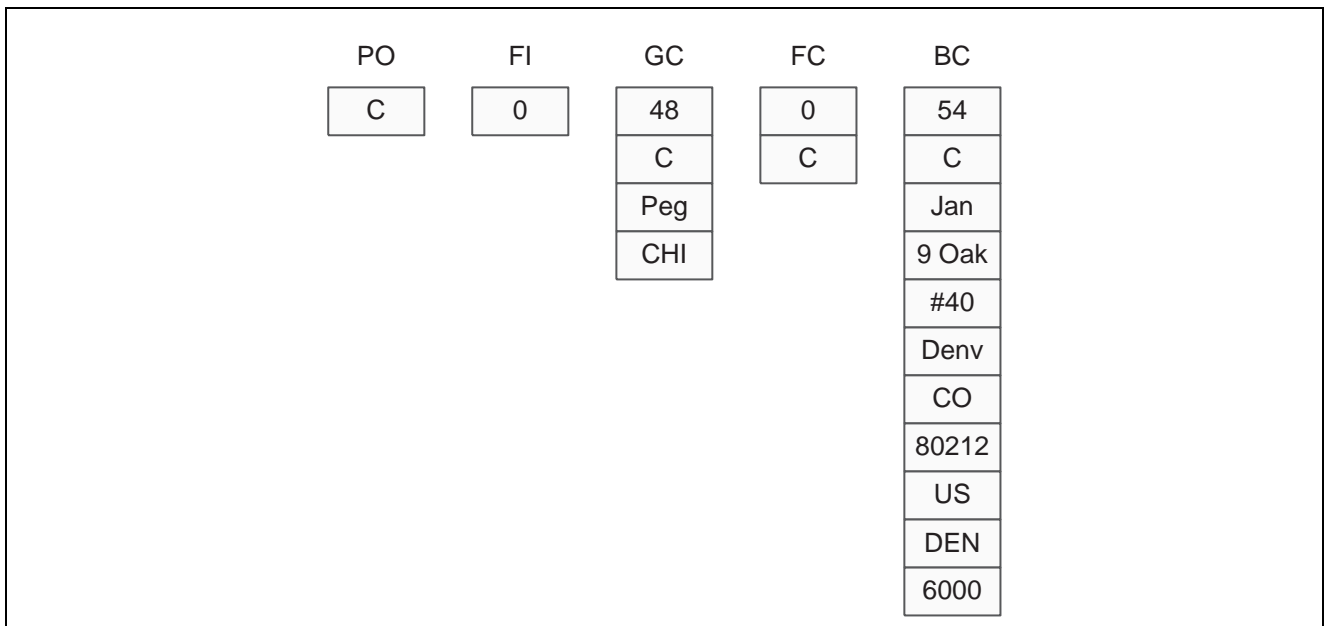
Content of runtime structures before Last Grid Record Has Been Read fires

The **Last Grid Record Has Been Read** event is commonly used to write total lines to the grid and to display totals that are based on grid values.

### Select Button Processing

When a user selects a grid row and clicks the Select button, the BC structure stays the same, however the GC structure reflects values on the row that is being selected.

This diagram illustrates the information in the runtime structures when the user selects a grid row that is other than the last fetched row. Note that the BC and GC structures do not contain the same values:



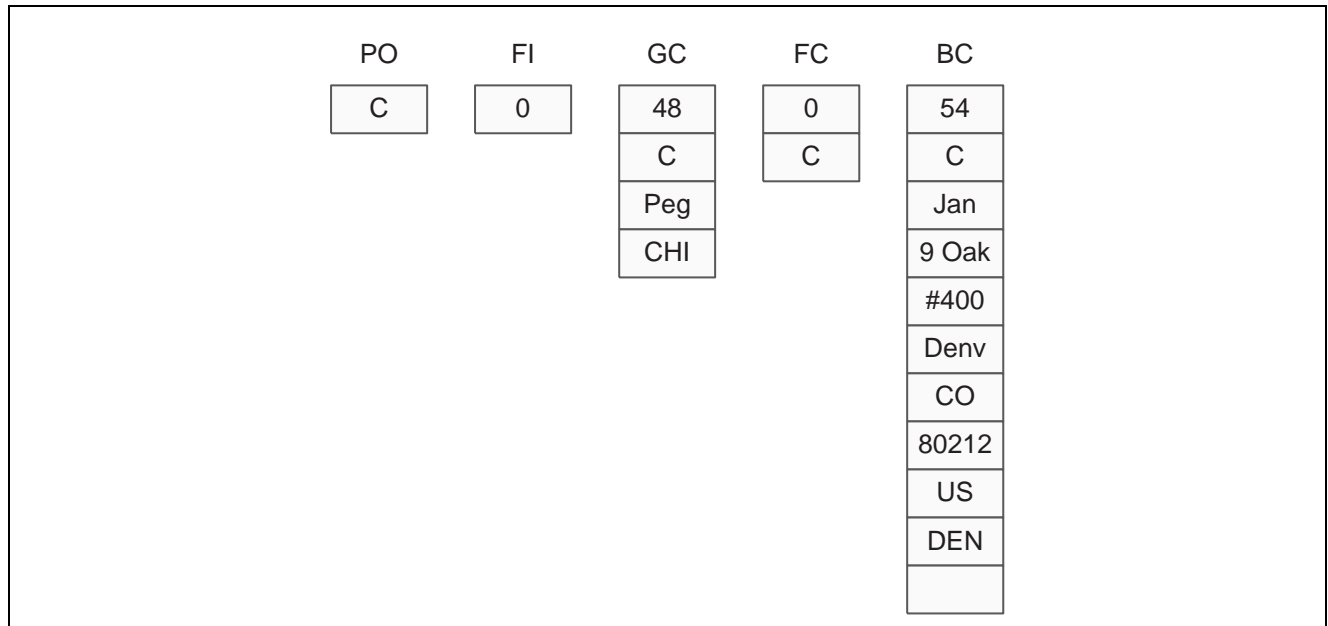
Content of runtime structures when user selects grid row that is other than the last fetched row

### Button Clicked

The engine then fires the **Button Clicked** event for the Select button. At this point, the runtime structures have these values:

- BC = Values from the database (from the last record read).
- FC = Values from the database (if the field is a database field).
- GC = Values from the selected grid row.
- FI = Values passed from a calling form (if any).
- PO = Values passed from processing options.

This diagram illustrates the information in the runtime structures just before the system fires **Button Clicked**:



Content of runtime structures before Button Clicked fires

The **Button Clicked** event is commonly used to connect to another form.

Use Repeat Business Rules for Grid to repeat ER when multiple rows are selected.

### Add Button Processing

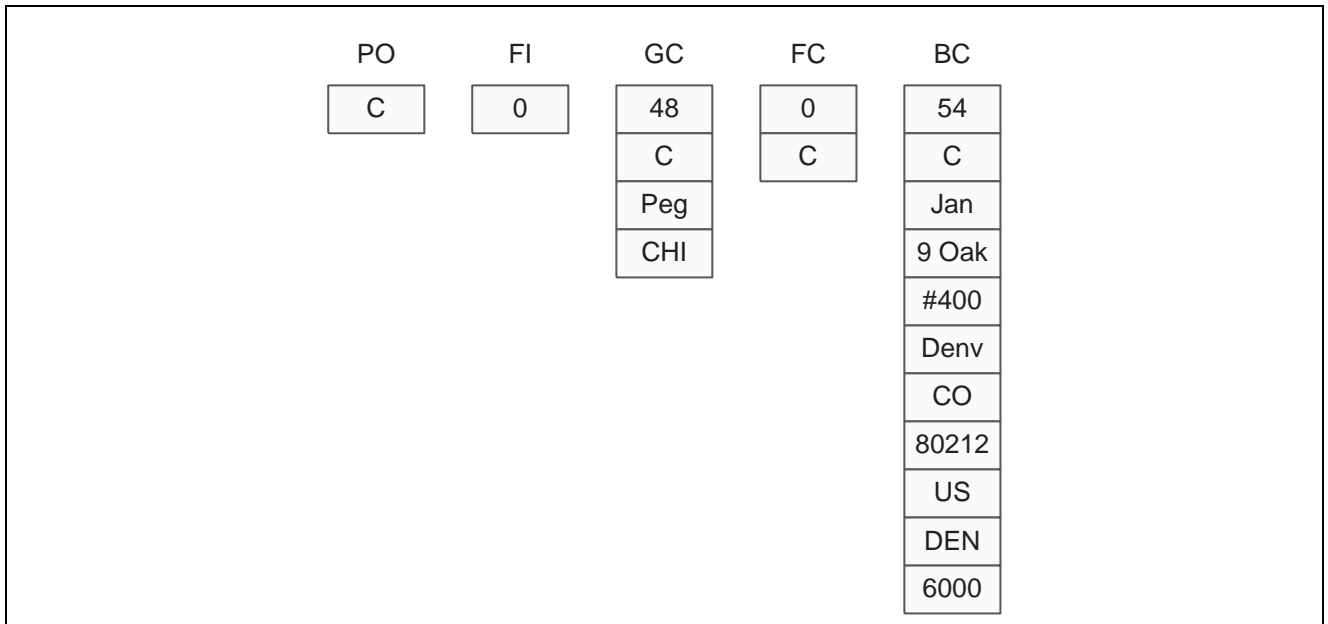
Normally, the user does not select a row before an add action, but if a row is highlighted, the system updates the GC values to reflect the selected row values. The system does not update the database just because the user clicks the row.

The engine pauses for the **Button Clicked** event to be processed. At this point, the runtime structures have these values:

- BC = Values from the database (from the last record read).
- FC = Values from the database (if the field is a database field).
- GC = Values from the database (from the selected row).
- FI = Values passed from a calling form (if any).
- PO = Values passed from POs.

Because this is an add action, the content of GC is irrelevant at this point. BC and GC do not contain the same values.

This diagram illustrates the information that is in the runtime structures just before the system fires **Button Clicked**:



Content of runtime structures before Button Clicked for Add Button fires

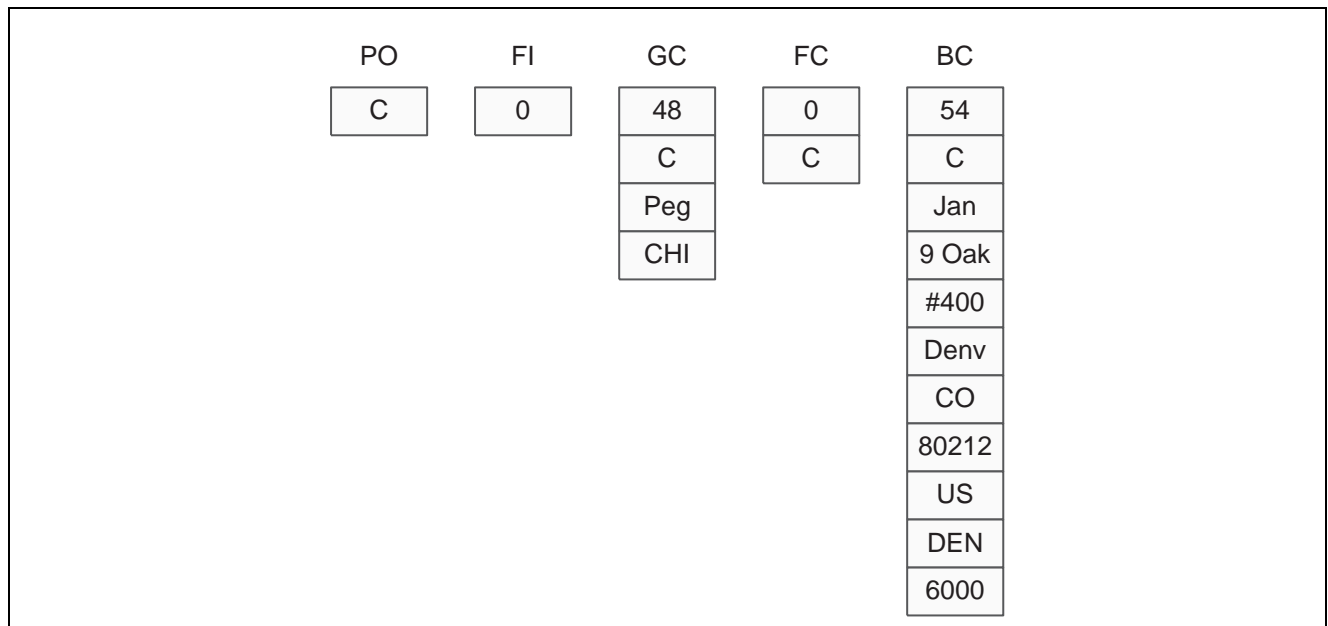
You typically use the **Button Clicked** event for the Add button to interconnect to another form, such as a fix/inspect or headerless detail form on which the system actually performs the add action.

## Delete Button Processing

When the user selects a grid row and clicks the Delete button, the system does not update the database immediately. The engine first fires the **Button Clicked** event for the Delete button. At this point, the runtime structures have these values:

- BC = Values from the database (from the last record read).
- FC = Values from the database (if the field is a database field).
- GC = Values from the database (from the selected row).
- FI = Values passed from a calling form (if any).
- PO = Values passed from POs.

This diagram illustrates the information in the runtime structures just before the system fires **Button Clicked** for the Delete button:

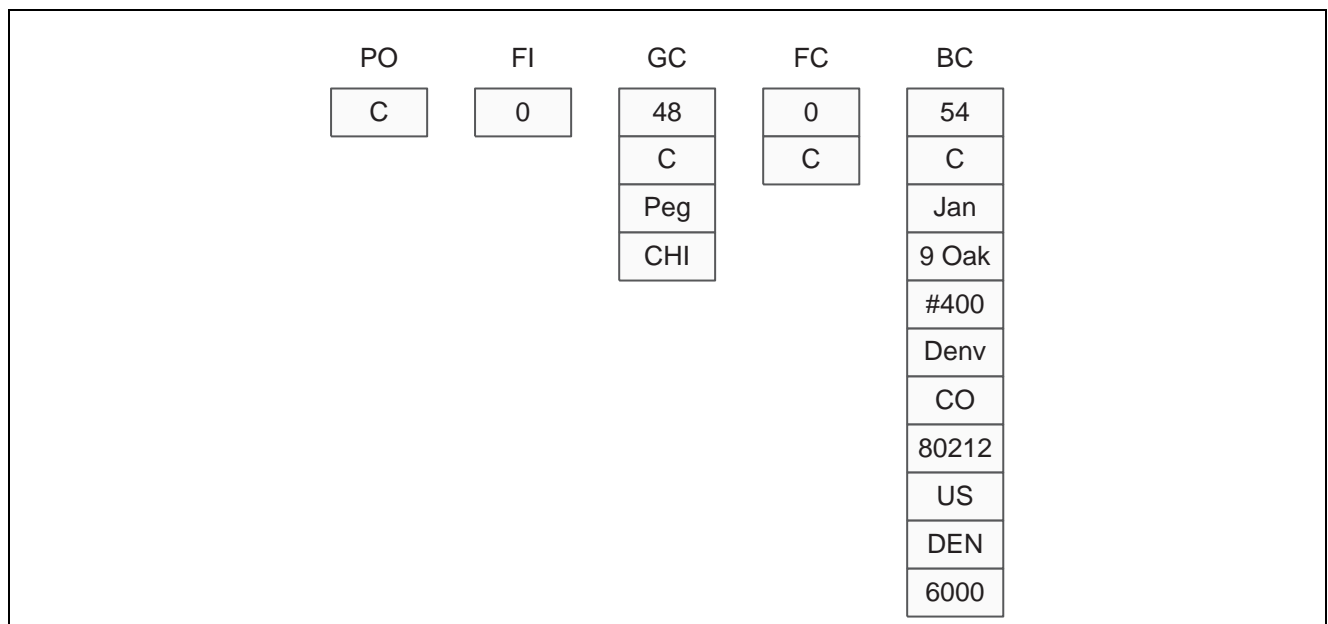


Content of runtime structures before delete Button Clicked fires

Next, the **Delete Grid Rec Verify–Before** event fires.

### Delete Grid Rec Verify–Before

This diagram illustrates how the engine fires the **Delete Grid Rec Verify–Before** event:



Content of runtime structure when Delete Grid Rec Verify–Before fires

Next, the system displays a pop up window for user to confirm the delete. If the delete is confirmed, the **Delete Grid Rec Verify–After** event fires.

### **Delete Grid Rec Verify–After**

In the **Delete Grid Rec Verify–After** event, you might want to perform custom logic to verify that the delete is valid. For example, other tables might contain dependant records that prevent this record from being deleted as long as they exist.

The system processes the logic that is attached to this event after the user clicks the OK button in the Verify confirmation form. If the user clicks the Cancel button in the Verify confirmation form, the logic attached to this event does not occur.

Next, the **Delete Grid Rec From DB–Before** event occurs.

### **Delete Grid Rec From DB–Before**

At this point, the runtime structure FC is blank. The system has not yet deleted the record from the database. You can use the **Suppress Delete** system function in this event to prevent the system from deleting the record. :

After the system processes the **Delete Grid Rec From DB–Before** event, it builds a SQL DELETE statement. Then the system deletes the current record. When user selects multiple records, all selected records are deleted.

### **Delete Grid Rec From DB–After**

After the records are deleted from the database, the system fires the **Delete Grid Rec From DB After** event. You might use this event to call a business function to delete information from related tables that are not in the current BV.

### **All Grid Recs Deleted From DB**

After all selected records are deleted, the engine fires the **All Grid Recs Deleted from DB** event. At this point, FC is blank.



## CHAPTER 3

# Using Event Rules Design

This chapter provides overviews of Oracle's JD Edwards EnterpriseOne Event Rules Design , Event Rule validation, If and While statements, Event Rule consistency, and Event Rule variables, and discusses how to use Event Rules Design to create and modify Event Rules.

---

## Understanding Event Rules Design

You can use JD Edwards EnterpriseOne Event Rules Design to create event rule (ER) logic for forms and controls on a form. For example, you want to pass data for a selected record on a find/browse form to a fix/inspect form to revise that record. You need to create a form interconnection ER and attach it to the Select button option for the **Button Clicked** event.

You can create event rules that perform a large variety of tasks, including:

- Perform a mathematical calculation.
- Pass data from a field on a form to a field on another form.
- Count grid rows that are populated with data.
- Interconnect two forms.
- Hide or display a control using a system function.
- Evaluate If/While and Else conditions.
- Assign a value or an expression to a field.
- Create variables or programmer-defined fields at runtime.
- Perform a batch process upon completion of an interactive application.
- Process table input and output, validate data, and retrieve records.

Before you create an ER, consider which control (form, button, field, grid and so on) you want to add the logic in and what event you want to add the logic for. Answer these questions to determine which event should be used:

- Is the user initializing the form?
- Is the user clicking a button?
- Is the user exiting from a field?
- Is the user changing or exiting from a row?

After you place controls on a form, you can add ERs to any of the event that the control support. Remember that a form is also a control, and you can create logic that the system automatically processes whenever a form event is fired.

You create ERs by clicking the buttons on the tool bar in JD Edwards EnterpriseOne Event Rules Design. Depending on the button that you click, a different work areas appear for creating and manipulating the ER line-by-line. Specific buttons enable you to perform these tasks:

- Attach a business function or system function.
- Create an If/While statement.
- Insert an Else clause in an If statement.
- Assign a value or expression.
- Create a variable.
- Create a form or report interconnection.
- Perform table I/O.
- Find a string in a given ER.
- Add comments in the ER code.
- Print the ER code.

You can cut or copy an ER and paste it in the same event, form, or application or in a different event, form, or application. You can also paste ERs into other applications, such as word processing documents. This feature is useful for documenting the project.

When you paste an ER, the system resolves objects from the source as you paste them. If an object is partially resolved, the system pastes the closest matching object from the destination ER. A comment line appears above the partially-resolved line of event rules and in the status bar to indicate that the object is partially resolved. You can set paste options to display comments before and after a block of pasted ERs. Some objects cannot be resolved in the destination ER. The system disables these lines of ER and displays a comment. For example, an EndIf statement is commented out if its associated If statement is missing.

For criteria statements, the paste operation adds whatever is necessary to maintain a clean, logical structure. For example, if you paste an If statement and no EndIf statement exists, the paste operation adds a matching EndIf statement to make the logic complete.

Use the System Function button to attach predefined system functions to events. For example, you can attach system functions to an event that perform these tasks:

- Hide or display a control.
- Display media objects.

You can attach an existing business function to an event. Business functions include these types of code:

- C code that you generate manually (source language C).
- Named Event Rules (NERs) (source language Event Rules).

You typically use business functions for these purposes:

- Referential integrity, such as deleting secondary records when a master record is deleted, and for editing routines.
- Large and complex calculations that might otherwise overload the runtime engine.

---

## Understanding Event Rule Validation

When you save an application, the system automatically validates all ERs in the application. If errors occur, details on the ER event, and the control and line number being executed at the time of the error are displayed in a popup window. You can also start the validation in FDA by selecting File, Validate Event Rules.

The error log that the system creates is stored in a file, such as `b9\prod\log\p1234.log` (where *prod* is the environment). If no errors exist, the system does not generate a log.

---

## Understanding If and While Statements

If and While statements are conditional instructions for an ER. They evaluate conditions and dictate the flow of logic when the ER is activated.

Use the If/While button to build conditional logic into an event. When you create an If statement, the system inserts an Else clause. However, you can use the Delete button to delete the Else clause and then reinsert it using the Insert Else button. When you delete an If or While statement, the system also deletes the associated Else and Endif or Endwhile clauses, but not the lines inside of those statements.

You can drag and drop statements line-by-line to change their sequence. Resequencing ER can result in improper syntax. When you click the Save or OK button, the system verifies the syntax. If it detects syntax errors, you can either disable the ER and continue or edit the ER to eliminate the errors.

To change a statement, double click the line.

To delete a statement, choose the line of the statement, and then click the Delete button.

---

## Understanding ER Consistency

You can check the consistency of events in JD Edwards EnterpriseOne Event Rule Designer. Events can be either relevant or not relevant. A *relevant* event is one that is valid and that can be executed by the associated control. A *not relevant event* is one that is not executed by the associated control because the properties of that control do not create the conditions that cause the events to execute. You can change a not relevant event to a relevant event by updating the properties of the control.

---

## Understanding ER Variables

An ER variable is a variable that you can use within the ER. You must assign a DD item to an ER variable. The DD item defines the type and default behavior of the variable.

Use ER variables instead of hidden fields. ER variables use fewer system resources at runtime.

After you add an ER variable, you cannot modify it. Instead, you must delete it and create another one.

Each ER variable is available within a scope. The scope of an ER variable determines how you can use it. Different scope options are available for interactive and batch applications. For example, you can:

- Reference a report variable anywhere in the report.
- Reference an ER variable only within the event in which it was created.

After you create an ER variable, it appears in the available objects list in JD Edwards EnterpriseOne Event Rules Designer where you added it. Use the ER variable in ERs just as you would use any available object in the list. If you create an event level variable and do not use it in ERs, FDA automatically deletes it.

The system automatically assigns to each variable one of these prefixes, based on the specified scope:

- frm\_ (Form)
- evt\_ (Event)
- grd\_ (Grid)
- rpt\_ (Report)
- sec\_ (Section)

---

## Prerequisites

Before you complete the tasks in this section:

- Create an application with one or more forms.
- Understand the difference between database items and data dictionary (DD) items.
- Understand the relationship between controls, events, and ER.
- Determine the purpose of each form used in the application.
- Answer these questions:
  - What logic is required?
  - For which control are you creating logic?
  - For which event will the logic occur?
  - Which runtime structures are affected?

---

## Working with Event Rules Design

This section provides overviews of assignments and event rules design tool bar buttons, and discusses how to:

- Display event information.
- Assign a value.
- Create an If or a While statement.
- Create an ER variable.
- Attach a system function to an event.
- Attach a business function to an event.

## Understanding Assignments

Use an assignment to assign a field with a fixed value or a mathematical expression. For example, you can create an assignment that inserts a default value when the user leaves the field. You can also use an assignment to calculate a value.

When you create an expression, calculate only the data items that are of the exact same numerical scale or data type. For example, do not calculate different currencies or decimal figures that represent different decimal values because the result of these calculations might compromise data integrity.

## Understanding Event Rules Design Tool Bar Buttons

The Event Rules Design form displays these tool bar buttons for generating different types of statements:

- **Event Information**  
Displays information about event relevance.
- **Assignment**  
Creates an assignment or a complex expression.
- **Business Function**  
Attaches an existing business function.
- **System Function**  
Attaches an existing JD Edwards EnterpriseOne system function.
- **If/While**  
Creates an If/While conditional statement.
- **Report Interconnect**  
Establishes a connection to a batch application or report.
- **Form Interconnect**  
Establishes a form interconnection.
- **Else**  
Inserts an Else clause, which is valid only within the bounds of If and End If.
- **Variable**  
Creates a programmer-defined field.
- **Table I/O**  
Enables ER support for database access. Performs table input and output, data validations, and record retrieval.

## Displaying Event Information

To display information about the relevance of an event:

1. On Event Rules Design, select an event.
2. Click the Event Information button.

A not relevant event is displayed in dimmed italic text in the Event combo box. A message box appears that displays any information known to the system about why a not relevant event is not relevant.

## Assigning a Value

To assign a value:

1. On Event Rules Design, select an event.
2. Click the Assignment/Expression button.
3. On Assignment, select the To Object that you want to receive the assigned value.
4. Use one of these methods to determine the From/Object Literal value:
  - Select a From Object in the right-hand column to create a simple statement: [left-hand column] = [right-hand column].
  - Type a literal expression (number, text, and so on) in the text entry box to assign a literal statement: [left-hand column] = [literal].
  - Press the  $f(X)$  button to create a complex expression or advanced mathematical function using Expression Manager.

## Creating an If or a While Statement

To create an If or a While statement:

1. On Event Rules Design, select an event in the Event Rules Design window and click the If/While button.  
Each cell in the Criteria Design grid represents a component of the criteria. When you select a cell, a list of valid options appears.
2. Select either the If or While operators.
3. Select a left operand from the list of available data items.  
Right-click to sort the available data items by name or object type. If only one type exists, the sort options are unavailable. The system groups the available data items by a variety of object types.
4. Select a logical operator comparison (is equal to, is less than, and so forth).
5. Select a right operand from the object list.
6. To assign a literal, select <Literal>.
7. To create complex If statements, you can select the And option or the Or option, and continue the logic.

## Creating an ER Variable

To create an ER variable:

1. On Event Rules Design, click the Variables button.  
The Variables form displays different scope options, depending on whether you are working with an interactive application, batch application, or NER.
2. Complete the variable naming field located under the Add button.
3. Click one of the Scope options (Form or Event) depending on the purpose for which you created the variable.
4. If you selected Form Scope and you want to use a grid variable, click the Grid option.
5. Click the DD visual assist to browse for DD items.
6. Select the DD item to which the variable is associated and click the Add button.

The system automatically assigns a prefix to the variable, based on the type of scope that you choose.

## Attaching a System Function to an Event

To attach a system function to an event:

1. On Event Rules Design, select an event.
2. Click the  $f(S)$  button.
3. Select a category in the System Functions box.
4. Select the system function that you want to attach.
5. In the Available Objects list, select objects to pass to the system function.

## Attaching a Business Function to an Event

To attach a business function to an event:

1. On Event Rules Design, select an event.
2. Click the  $f(B)$  button.  
You can view a description (if one exists) for a business function by choosing Attachments from the Row menu.
3. Select a business function and click the Select button.
4. In the Available Objects list, select objects to pass to the business function.
5. To assign a literal to a business function parameter, select *<Literal>* in the Available Objects list.
6. Enter a single value and click the OK button.

Range and List are not valid literals to use with business function parameters.

7. Indicate the direction of data flow between Value and Data Items, and click the OK button.

As you click the direction arrow, it toggles through these four options:

- Data flows from the source to the target (right-pointing arrow).
- Data flows from the target to the source (left-pointing arrow).
- Data flows from the source to the target, and upon exiting the target, data flows back to the source (bi-directional arrow).
- No data flow (slashed circle).

If the direction of the items is hard-coded in the data structure for a business function (such as when the parameters are predetermined to be input, output, or bidirectional), then this predetermined direction appears here. You must complete the required items that appear in red. The status bar indicates the state of the flow to the target.

8. Select the Include in Transaction option to include this business function for transaction processing.  
This option appears on transaction forms only.
9. Select the Asynchronously option to enable asynchronous processing.
10. Click one of these buttons to add notes:

- Business Function Notes
- Structure Notes

- Parameter Notes

## CHAPTER 4

# Using BrowsER

This chapter provides an overview of Oracle's JD Edwards EnterpriseOne BrowsER utility and discusses how to use it.

---

## Understanding BrowsER

You can use JD Edwards EnterpriseOne BrowsER to view event rules (ER) for interactive and batch applications.

JD Edwards EnterpriseOne BrowsER displays the structure of forms within an interactive application, or sections within a batch application. The forms or sections appear in a hierarchical structure, with events and ER for each event. You can use JD Edwards EnterpriseOne BrowsER to search and filter ER, and disable or enable ER.

You can select one of these JD Edwards EnterpriseOne BrowsER options to easily view or search for ER code:

- Expand Tree
- Expand Node
- Show Object IDs
- Hide Objects with no ER
- Filter ER Records

Filter ER Records enables you to show or hide specific ER statements, including:

- Assignments
  - Business functions
  - Criterion
  - Comments
  - Form interconnections
  - Options
  - System functions
- Search ER Records enables you to search for specific ER statements or text within those statements.

---

## Working with BrowsER

To work with JD Edwards EnterpriseOne BrowsER:

1. On Oracle's JD Edwards EnterpriseOne Object Management Workbench (OMW), select an object with ER and then click the Design button.
2. On Interactive Design, click the Design Tools tab, and then click the Browse Event Rules button.  
Alternatively, you can access JD Edwards EnterpriseOne BrowsER directly from within Oracle's JD Edwards EnterpriseOne Form Design Aid (FDA) or Oracle's JD Edwards EnterpriseOne Report Design Aid (RDA) by choosing BrowsER from the View menu.
3. On Browsing, click the plus (+) and minus (–) buttons to expand or collapse the hierarchical view of events for interactive forms or batch report sections.  
Each ER appears beneath the event with which it is associated and beside a control that contains event rule logic. If it does not appear beside a control, then no event rule logic exists on that control.
4. To disable an ER line, select the line and then click Disable button.
5. To enable a disabled line, select the line and then click Enable button.  
You cannot print or modify ER from any BrowsER form.
6. To hide objects with no ER, right-click anywhere on the BrowsER window and select Hide objects with no ER from the popup menu.
7. To start a search or filter, right-click anywhere on the Browsing form and select Search or Filter ER Records from the popup menu.

## CHAPTER 5

# Debugging Event Rules

This chapter provides overviews of debugging, the debugging process, interpreted vs. compiled code, debugging strategies, and the debug log, and discusses how to debug event rules.

---

## Understanding Debugging

Debugging is the method you use to determine the state of your program at any point of execution. Use debugging to help you solve problems and to test and confirm program execution.

Use a debugger to stop program execution so you can see the state of the program at a specific point. This enables you to view the values of input parameters, output parameters, and variables at the specified point. When program execution is stopped, you can review the code line-by-line to check such issues as flow of execution and data integrity.

You use Oracle's JD Edwards EnterpriseOne Event Rules Debugger to debug interactive applications, reports (batch applications), and table conversions.

---

## The Debugging Process

Use the debugging process to determine where problems occur and then fix those problems. Isolate each problem to a particular area, and then examine exactly how the program operates in that area.

If you change your program while you are debugging with the JD Edwards EnterpriseOne Event Rules Debugger, you must:

1. Exit the application.
2. Rebuild debug information.
3. Reset breakpoints.
4. Rerun the application.

Features available in the event rule debugger are listed and described in this table:

Feature	Description
Go	Command that resumes program execution after a breakpoint is reached.
Breakpoint	Command that tell the debugger to stop when a particular line is reached. You can set breakpoints on lines of code where you want to start debugging.

Feature	Description
Delete Breakpoint	Command that removes all breakpoints that you currently have set.
Step, Step Over	Command that executes the current line of code. Step lets you run the program one line at a time. You can use this feature to determine the results of every line of code as it is executed.
Step Into	Command used when the current line of code contains a function call. The debugger will step into the function so that it can be debugged line by line. When the function is complete, the debugger returns to the next line of code after the function call in the calling routine. Step Into can be used to debug a second application that is called from within an application.
Disconnect	Command that disconnects the debugger from the current application. The application continues to run as if the debugger had not been started.

## Debugging Strategies

You can use several strategies to make debugging faster and easier. Begin by observing the nature of the problem.

### Is the Program Ending Unexpectedly?

If the program is ending unexpectedly, the cause is likely an unhandled exception. It is an easy problem to track down if it is happening in the same place: simply set breakpoints at strategic points throughout the code and run the program until you find the problem.

If other objects are missing, termination is more abrupt. Remember to transfer all Media Object (also called Generic Text) objects correctly. If an application has a Row exit to an application that does not exist, an unhandled exception in the program occurs immediately.

Termination of the program is more abrupt and less helpful when other kinds of objects are missing. You must review all of the pieces of your application to verify that they are all present and correctly built. A common error is to overlook media objects. If you cannot enter your program at all, a missing object is most likely the problem.

### Is the Output of the Program Incorrect?

Incorrect program output typically indicates a flaw within the logic of the code. To help find the error:

- Set a breakpoint in the code prior to the point where the bad output is produced.
- Step through the ER line by line, while monitoring the values of relevant ER variables.

At some point, a variable will probably take on an erroneous value that subsequently produces incorrect output.

- If that point occurs before your breakpoint, set another breakpoint earlier in the code and restart the application.
- Continue this process until you find the statement that is causing the wrong value to be assigned to the variable.

## Where Else Could the Problem Be Coming From?

Spend some time thinking about where the source of the problem might be. If you don't know which ER event is causing an error, try to isolate it. For example, you might be able to temporarily disable the ER one event at a time to see if the error still happens. You can try to repeat the processing of a single event by doing unnatural actions in the GUI, like toggling up and down between grid rows to force the execution of the **Row Is Exited** event. There are no predefined debugging strategies that will work in any given situation. Be creative and be persistent, until you narrow down the problem to its source.

---

## Debug Logs

You can output to a file a log of SQL statements and events by changing the line in your jde.ini file under [DEBUG] from `Output = NONE` to `Output = FILE`, as in this sample. This is a useful debugging tool when you have narrowed a problem to a specific issue involving the JDEDB APIs.

```
[DEBUG]
TAMMULTIUSERON=0
Output=FILE
ServerLog=0
LEVEL=BSFN,EVENTS
DebugFile=c:\jdedebug.log
JobFile=c:\jde.log
Frequency=10000
RepTrace=0
```

The jdedebug.log file can become very large when you set the `Output=DEBUG` flag. It is very difficult to work through all the log messages and correlate them to what was happening in the application. One way to aid in this process is to set breakpoints in the ER, and each time the application stops at a breakpoint, make a separate copy of the jdedebug.log file. Label each copy, and when you are done, you will have narrowed down the coverage of each portion of the jdedebug.log file a bit.

---

## Debugging Event Rules

This section provides an overview of the JD Edwards EnterpriseOne Event Rules Debugger and discusses how to:

- Debug an application with the JD Edwards EnterpriseOne Event Rules Debugger.
- Inspect or modify a variable.

## Understanding the Event Rules Debugger

The JD Edwards EnterpriseOne Event Rules Debugger provides the essential debugging tools (such as breakpoints, step commands, and variable inspection) that you need to debug JD Edwards EnterpriseOne interactive and batch applications. You can debug both NERs and table ER. The generated debug information for an application includes NER and table ER information for that application.

Setting up and using JD Edwards EnterpriseOne Event Rules Debugger involves these steps:

1. Launch the ER Debugger.
2. Load into the Debugger the applications to debug.
3. Set any desired breakpoints.
4. Launch the application, report, or table conversion.

Step 4 may be done at any point before, during or after steps 1, 2, or 3.

Step 2 takes a few minutes. The Debugger must read all the specs for the application, and then translate them into a usable format (Debugging Information Archive, or DIA). For large applications, this is slow. Therefore, once you load an application, you don't want to have to reload it again any time soon (unless you modify the application, of course). For this reason, the ER Debugger provides a Deactivate feature. Deactivating an application prevents any debugging from occurring on that application. But when you are ready to debug the application later, you won't have to rebuild the DIA.

Any event on which you want the debugger to stop must have at least one line of ER code. You cannot set a breakpoint on a comment. When you debug an application and encounter a point at which the interactive or batch application fails, you can view the code that the application is currently executing, and inspect the live values of variables used in the code.

By observing specific variables while the program runs, you can isolate where the program begins to fail and what exactly it is doing. For example, if a counter is supposed to increment by 1, but you observe it incrementing by random numbers, you know there is a problem with the number or variable you are adding to the counter.

The JD Edwards EnterpriseOne Event Rules Debugger is a standalone tools program. Its screen interface consists of these main components:

- Object Browse window.
- Object Tree (those applications currently loaded in Debugger).
- Event Rules window (ER listing).
- Breakpoint Manager.
- Variables List and Variable Watch window.

All windows except the Event Rules window are dockable to any side of the main application. You can click and drag a window to dock it. When you close the JD Edwards EnterpriseOne Event Rules Debugger, it saves your docking settings until the next time you run it. Most of these main windows may also be closed by clicking the large toolbar buttons which control them, or by clicking the 'X' in the top border of the window.

## Object Browse Window

The Object Browse window is used to locate applications to load into the debugger. It provides three tab pages, one for interactive applications, one for UBEs, and one for TCs. You can find the object you want to debug in one of these lists, and then select it in order to bring it into the debugger.

Once you are done loading objects into the Debugger, you may want to close, or hide, the Object Browse window, because it takes up screen space that could be more usefully dedicated to one of the other Debugger windows. To hide this window, toggle the Object Browse Window button in the toolbar, or use the View menu.

## Object Tree

The Object Tree view lists applications that have debug information built and are available for debugging. You can navigate through a tree structure to a specific event and open an Event Rules window for that event. If one of the objects is a power form with subforms, the subforms are listed under the power form. Form IDs appear next to the form name.

## Event Rules Window

Each Event Rules window displays the ER for one event. The event name and path, along with an abbreviated description, appear in the title bar for each Event Rules window. The Event Rules window shows the line in the ER that is currently being executed when the runtime engine is stopped on a line break.

The left side of an Event Rule window displays icons that describe the state of a line in the ER. States include breakpoint, disabled, or current line of execution.

You can use the Event Rules window to set and remove breakpoints. You can use any of these methods:

- Double-click the line in ER.
- Right-click a line and select Insert Breakpoint or New Breakpoint from the pop-up menu.
- Select a line and press F9.
- Single click in the margin to the left of the ER text.

You cannot set a breakpoint on a comment line. The breakpoint automatically goes to the first code line after the comment. You cannot set a breakpoint on a data structure member displayed in the ER listing. The breakpoint will automatically go to the function call above it that “owns” the data structure.

The ER window is also useful to show the values of ER variables during a debugging session. The value of a variable in the ER listing can be instantly displayed by moving the mouse cursor over the variable. Its value will appear in a Tool Tip window when the ER Engine is paused (when stopped on a breakpoint or when stepping through code in the debugger).

Variables spelled differently in the ER listing and the variable list will not display.

## Variable Tree and Watch Window

When the program is halted at a breakpoint, you can examine the state of your runtime structures and evaluate ER variables using the Variable Tree and Variable Watch windows.

The Variable Tree and Watch window consists of two panes. The left pane is the Variable Tree pane. It contains a tree structure that lists the variable types as parent nodes and the variables of each variable type as child nodes. The variables displayed are those that are in scope of the currently displayed event in the Event Rule window. The right pane is the Watch pane. It displays variables you select and their most recently known values. Each variable is identified by its category, name, and the Id of the form, if any, to which it belongs. You can add a variable to the Watch pane by double-clicking the desired variable in the Variable Tree, or you can right click on the variable in the Variable Tree and select “Watch Variable” to add it to the Watch list.

You can change the value of variables while you are debugging an application, in order to better understand what effect that might have on subsequent ER execution. This is a powerful feature, which can also mess up your program execution and create unwanted side effects. Use it carefully. To change the value of a variable, the application must be stopped at a line of ER. Double-click the variable in the Watch pane and enter a different value. If the ER engine running the application accepts the new value, the new value appears in the Watch pane. If you enter an inappropriate value (for example, you change a numeric value to an alpha value) the new value is not set and the value is not changed. In any case, the Debugger displays the actual, resulting variable value.

These special values are displayed for variables:

- blank

The value for the variable contains only blanks. This value applies to string and character types only.

- null

The variable has no value, or a null or empty value.

- unknown

The value for the variable could not be obtained from the runtime engine. When adding a variable to the watch list, the initial value is always unknown. The value is also unknown when the applications are not running, or when the variable is out of scope.

---

**Note.** Variable inspection and modification is not available for debugging NERs and table ER.

---

## Breakpoint Manager

You use breakpoints to define where or when to halt the execution of a program. The Breakpoint Manager tracks the breakpoints that are set and the location of those breakpoints in an application. When you set a new breakpoint, the system creates an entry in the Breakpoint Manager. This entry contains the application name, form name, event name, ER line number, and the breakpoint conditions, if any.

Right-click within Breakpoint Manager to perform these operations:

- Delete a breakpoint.
- Delete all breakpoints.
- Display the ER window in which the breakpoint is set.
- Access breakpoint properties.
- Enable or disable a breakpoint.

You can also double-click an entry in Breakpoint Manager to open the Event Rule window in which the breakpoint is set.

## Breakpoint State Indicators

Normal breakpoints are indicated by a red circle.

Breakpoint conditions are indicated by a question mark inside the red circle.

A disabled breakpoint is indicated by a hollow circle in the ER listing.

## Breakpoint with Condition

Formerly, ER breakpoints existed with no apparent properties. Breakpoints now have Condition and Hit Count properties which may be set or examined in the Breakpoint Properties dialog.

To set a breakpoint condition in the form of a logical expression:

1. In the Breakpoint Properties dialog click the Condition button.
2. The Breakpoint Condition dialog will now display. In this window you may enter the condition in the available field or click the Help button. The Help button brings up a dialog that documents the syntax for breakpoint conditions to assist in creating a condition.

---

**Note.** For example one might write a condition using the format of “VariableName = LiteralValue”. The resulting condition might be `PO cSelfServiceMode='1'`.

---

3. After entering a condition, the developer may use the Validate Condition button to validate the expression.

---

**Note.** Checking the Condition checkbox in the Breakpoint Condition dialog enables the Condition. If the Condition checkbox is not checked then the condition is saved for possible later use, but is not effective.

---

## Avoiding Problems with Breakpoint Conditions

To avoid unnecessary problems it is best to:

- Validate breakpoint conditions as you create them.
- Spell variable names exactly as they appear in the variable list, including spaces and special characters where present.

It is possible to set an invalid condition on a breakpoint. There may even be times when you would do this intentionally. Invalid conditions fail when evaluated. It is good practice to utilize the Validate Condition button to validate conditions as you create them.

Misspelled variable names are invalid, of course. The official spelling for all variables is shown in the Debugger's variable list.

Out-of-scope variables are invalid, depending on the context of the breakpoint in which the condition is set.

Any other syntax violations make an expression invalid.

The Debugger does not validate date and number formats. This validation is done at runtime, and may depend on certain User Options.

## Breakpoint with Hit Count Condition

The second Breakpoint property is the Hit Count condition. The Hit Count condition specifies a certain number of times the breakpoint must be reached before the debugger will stop on it.

The Hit Count dialog is used by the developer to specify a Hit Count condition. It also shows the current hit count, that is, the number of times the breakpoint has been reached since the hit count condition was created.

When both an expression Condition and a Hit Count condition are set on a single breakpoint, the debugger stops on the breakpoint if either condition is met.

## Disabled Breakpoint

A breakpoint may be disabled. A disabled breakpoint retains all its conditions and other properties, but the debugger does not stop on a disabled breakpoint.

## Invalid Breakpoints

Invalid breakpoint conditions are logged to `jddebug.log` each time the breakpoint having the invalid condition is reached.

## Search Combo Box

You can use the Search combo box on the tool bar to search for ER text. Enter the text that you want to find in the Search combo box and then press either ENTER or F3. If the system locates the search text in your ER text, it highlights the text. If you press ENTER or F3 again, the next occurrence of your search text is located and highlighted.

The search control accommodates regular expression searches. A regular expression search uses special characters to match text. For example, `^If :` will find every line that starts with If and `If$ :` will find every line that ends with If.

The special characters that you can use for advanced searches are described in this table:

Character	Description
^	The caret (^) indicates the beginning of a line. For example, the expression ^A matches an A only at the beginning of a line.
^	The caret (^) immediately after the left bracket ( [ ) is used to exclude any remaining characters within brackets from matching the target string. For example, the expression [^0-9] indicates that the target character should not be a digit.
\$	The dollar sign (\$) matches the end of a line. For example, the expression abc\$ will match the substring abc only if it is at the end of a line.
	The alternation character ( ) enables the expression on either side of it to match the target string. For example, the expression a b will match a as well as b.
.	The dot (.) matches any character.
*	The asterisk (*) indicates that the character to the left of the asterisk in the expression should match 0 or more times.
+	The plus (+) is similar to the asterisk except that at least one match of the character should occur to the left of the + sign in the expression.
?	The question mark (?) matches the character to its left 0 or 1 times.
()	The parentheses affect the order of pattern evaluation and serve as a tagged expression that you can use to replace a matched substring with another expression.
[ ]	Brackets that enclose a set of characters indicate that any of the enclosed characters can match the target character.

## Debugging an Application with the Event Rules Debugger

To debug an application with the JD Edwards EnterpriseOne Event Rules Debugger:

1. From the Cross Application Development Tools menu (GH902), select Debug Application.
2. Select the object that you want to debug.
3. Select a form (for interactive applications) or section (for batch applications) and an event to view.
4. Select the ER line on which you want to set a breakpoint.
5. Select Debug, Breakpoint.

A red dot appears on the line, indicating the breakpoint.

You can remove the breakpoint by choosing Debug, Breakpoint again. The Breakpoint command is a toggle, and you can also toggle the value using the Breakpoint toolbar button.

6. Run the application.

As your application encounters a breakpoint, the application will pause, and the focus will switch to the Event Rules Debugger.

When execution stops at a breakpoint, you can use the variables view to inspect and modify the values of runtime structures.

7. From the Debug menu, select one of these options:
  - Go
  - Disconnect
  - Step Over
  - Step Into



# Glossary of JD Edwards EnterpriseOne Terms

<b>Accessor Methods/Assessors</b>	Java methods to “get” and “set” the elements of a value object or other source file.
<b>activity rule</b>	The criteria by which an object progresses from one given point to the next in a flow.
<b>add mode</b>	A condition of a form that enables users to input data.
<b>Advanced Planning Agent (APAg)</b>	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.
<b>alternate currency</b>	<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>
<b>Application Server</b>	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).
<b>as if processing</b>	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.
<b>as of processing</b>	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.
<b>Auto Commit Transaction</b>	A database connection through which all database operations are immediately written to the database.
<b>back-to-back process</b>	A process in JD Edwards EnterpriseOne Supply Management that contains the same keys that are used in another process.
<b>batch processing</b>	<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>
<b>batch server</b>	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.
<b>batch-of-one immediate</b>	<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>
<b>best practices</b>	Non-mandatory guidelines that help the developer make better design decisions.

<b>BPEL</b>	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.
<b>BPEL PM</b>	Abbreviation for Business Process Execution Language Process Manager, a comprehensive infrastructure for creating, deploying, and managing BPEL business processes.
<b>Build Configuration File</b>	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.
<b>build engineer</b>	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.
<b>Build Program</b>	A WIN32 executable that reads build configuration files and generates an ANT script for building published business services.
<b>business analyst</b>	An actor that determines if and why an EnterpriseOne business service needs to be developed.
<b>business function</b>	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.
<b>business function event rule</b>	See named event rule (NER).
<b>business service</b>	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.
<b>business service artifacts</b>	Source files, descriptors, and so on that are managed for business service development and are needed for the business service build process.
<b>business service class method</b>	A method that accesses resources provided by the business service framework.
<b>business service configuration files</b>	Configuration files include, but are not limited to, interop.ini, JDBj.ini, and jdelog.properties.
<b>business service cross reference</b>	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.
<b>business service cross-reference utilities</b>	Utility services installed in a BPEL/ESB environment that are used to access JD Edwards EnterpriseOne orchestration cross-reference data.
<b>business service development environment</b>	A framework needed by an integration developer to develop and manage business services.
<b>business services development tool</b>	Otherwise known as JDeveloper.
<b>business service EnterpriseOne object</b>	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.

<b>business service framework</b>	Parts of the business service foundation that are specifically for supporting business service development.
<b>business service payload</b>	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.
<b>business service property</b>	Key value data pairs used to control the behavior or functionality of business services.
<b>Business Service Property Admin Tool</b>	An EnterpriseOne application for developers and administrators to manage business service property records.
<b>business service property business service group</b>	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.
<b>business service property categorization</b>	A way to categorize business service properties. These properties are categorized by business service.
<b>business service property key</b>	A unique name that identifies the business service property globally in the system.
<b>business service property utilities</b>	A utility API used in business service development to access EnterpriseOne business service property data.
<b>business service property value</b>	A value for a business service property.
<b>business service repository</b>	A source management system, for example ClearCase, where business service artifacts and build files are stored. Or, a physical directory in network.
<b>business services server</b>	The physical machine where the business services are located. Business services are run on an application server instance.
<b>business services source file or business service class</b>	One type of business service artifact. A text file with the .java file type written to be compiled by a Java compiler.
<b>business service value object template</b>	The structural representation of a business service value object used in a C-business function.
<b>Business Service Value Object Template Utility</b>	A utility used to create a business service value object template from a business service value object.
<b>business services server artifact</b>	The object to be deployed to the business services server.
<b>business view</b>	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.
<b>central objects merge</b>	A process that blends a customer's modifications to the objects in a current release with objects in a new release.
<b>central server</b>	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.

<b>charts</b>	Tables of information in JD Edwards EnterpriseOne that appear on forms in the software.
<b>check-in repository</b>	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).
<b>connector</b>	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.
<b>contra/clearing account</b>	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.
<b>Control Table Workbench</b>	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.
<b>control tables merge</b>	A process that blends a customer's modifications to the control tables with the data that accompanies a new release.
<b>correlation data</b>	The data used to tie HTTP responses with requests that consist of business service name and method.
<b>cost assignment</b>	The process in JD Edwards EnterpriseOne Advanced Cost Accounting of tracing or allocating resources to activities or cost objects.
<b>cost component</b>	In JD Edwards EnterpriseOne Manufacturing, an element of an item's cost (for example, material, labor, or overhead).
<b>credentials</b>	A valid set of JD Edwards EnterpriseOne username/password/environment/role, EnterpriseOne session, or EnterpriseOne token.
<b>Cross-reference utility services</b>	Utility services installed in a BPEL/ESB environment that access EnterpriseOne cross-reference data.
<b>cross segment edit</b>	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.
<b>currency restatement</b>	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.
<b>cXML</b>	A protocol used to facilitate communication between business documents and procurement applications, and between e-commerce hubs and suppliers.
<b>database credentials</b>	A valid database username/password.
<b>database server</b>	A server in a local area network that maintains a database and performs searches for client computers.
<b>Data Source Workbench</b>	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.
<b>date pattern</b>	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.

<b>denominated-in currency</b>	The company currency in which financial reports are based.
<b>deployment artifacts</b>	Artifacts that are needed for the deployment process, such as servers, ports, and such.
<b>deployment server</b>	A server that is used to install, maintain, and distribute software to one or more enterprise servers and client workstations.
<b>detail information</b>	Information that relates to individual lines in JD Edwards EnterpriseOne transactions (for example, voucher pay items and sales order detail lines).
<b>direct connect</b>	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.
<b>Do Not Translate (DNT)</b>	A type of data source that must exist on the iSeries because of BLOB restrictions.
<b>dual pricing</b>	The process of providing prices for goods and services in two currencies.
<b>duplicate published business services authorization records</b>	Two published business services authorization records with the same user identification information and published business services identification information.
<b>embedded application server instance</b>	An OC4J instance started by and running wholly within JDeveloper.
<b>edit code</b>	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.
<b>edit mode</b>	A condition of a form that enables users to change data.
<b>edit rule</b>	A method used for formatting and validating user entries against a predefined rule or set of rules.
<b>Electronic Data Interchange (EDI)</b>	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.
<b>embedded event rule</b>	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.
<b>Employee Work Center</b>	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.
<b>enterprise server</b>	A server that contains the database and the logic for JD Edwards EnterpriseOne.
<b>Enterprise Service Bus (ESB)</b>	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).
<b>EnterpriseOne administrator</b>	An actor responsible for the EnterpriseOne administration system.
<b>EnterpriseOne credentials</b>	A user ID, password, environment, and role used to validate a user of EnterpriseOne.
<b>EnterpriseOne object</b>	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.

<b>EnterpriseOne development client</b>	Historically called “fat client,” a collection of installed EnterpriseOne components required to develop EnterpriseOne artifacts, including the Microsoft Windows client and design tools.
<b>EnterpriseOne extension</b>	A JDeveloper component (plug-in) specific to EnterpriseOne. A JDeveloper wizard is a specific example of an extension.
<b>EnterpriseOne process</b>	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.
<b>EnterpriseOne resource</b>	Any EnterpriseOne table, metadata, business function, dictionary information, or other information restricted to authorized users.
<b>Environment Workbench</b>	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.
<b>escalation monitor</b>	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.
<b>event rule</b>	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.
<b>explicit transaction</b>	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.
<b>exposed method or value object</b>	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.
<b>facility</b>	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.”
<b>fast path</b>	A command prompt that enables the user to move quickly among menus and applications by using specific commands.
<b>file server</b>	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.
<b>final mode</b>	The report processing mode of a processing mode of a program that updates or creates data records.
<b>foundation</b>	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.
<b>FTP server</b>	A server that responds to requests for files via file transfer protocol.
<b>header information</b>	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.
<b>HTTP Adapter</b>	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.

<b>instantiate</b>	A Java term meaning “to create.” When a class is instantiated, a new instance is created.
<b>integration developer</b>	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.
<b>integration point (IP)</b>	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.
<b>integration server</b>	A server that facilitates interaction between diverse operating systems and applications across internal and external networked computer systems.
<b>integrity test</b>	A process used to supplement a company’s internal balancing procedures by locating and reporting balancing problems and data inconsistencies.
<b>interface table</b>	See Z table.
<b>internal method or value object</b>	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.
<b>interoperability model</b>	A method for third-party systems to connect to or access JD Edwards EnterpriseOne.
<b>in-your-face-error</b>	In JD Edwards EnterpriseOne, a form-level property which, when enabled, causes the text of application errors to appear on the form.
<b>IServer service</b>	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.
<b>jargon</b>	An alternative data dictionary item description that JD Edwards EnterpriseOne appears based on the product code of the current object.
<b>Java application server</b>	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.
<b>JDBNET</b>	A database driver that enables heterogeneous servers to access each other’s data.
<b>JDEBASE Database Middleware</b>	A JD Edwards EnterpriseOne proprietary database middleware package that provides platform-independent APIs, along with client-to-server access.
<b>JDECallObject</b>	An API used by business functions to invoke other business functions.
<b>jde.ini</b>	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.
<b>JDEIPC</b>	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.
<b>jde.log</b>	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.
<b>JDENET</b>	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.
<b>JDeveloper Project</b>	An artifact that JDeveloper uses to categorize and compile source files.

<b>JDeveloper Workspace</b>	An artifact that JDeveloper uses to organize project files. It contains one or more project files.
<b>JMS Queue</b>	A Java Messaging service queue used for point-to-point messaging.
<b>listener service</b>	A listener that listens for XML messages over HTTP.
<b>local repository</b>	A developer's local development environment that is used to store business service artifacts.
<b>local standalone BPEL/ESB server</b>	A standalone BPEL/ESB server that is not installed within an application server.
<b>Location Workbench</b>	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.
<b>logic server</b>	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.
<b>MailMerge Workbench</b>	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.
<b>Manual Commit transaction</b>	A database connection where all database operations delay writing to the database until a call to commit is made.
<b>master business function (MBF)</b>	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.
<b>master table</b>	See published table.
<b>matching document</b>	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.
<b>media storage object</b>	Files that use one of the following naming conventions that are not organized into table format: Gxxx, xxxGT, or GTxxx.
<b>message center</b>	A central location for sending and receiving all JD Edwards EnterpriseOne messages (system and user generated), regardless of the originating application or user.
<b>messaging adapter</b>	An interoperability model that enables third-party systems to connect to JD Edwards EnterpriseOne to exchange information through the use of messaging queues.
<b>messaging server</b>	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.
<b>Middle-Tier BPEL/ESB Server</b>	A BPEL/ESB server that is installed within an application server.
<b>Monitoring Application</b>	An EnterpriseOne tool provided for an administrator to get statistical information for various EnterpriseOne servers, reset statistics, and set notifications.

<b>named event rule (NER)</b>	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.
<b><i>nota fiscal</i></b>	In Brazil, a legal document that must accompany all commercial transactions for tax purposes and that must contain information required by tax regulations.
<b><i>nota fiscal factura</i></b>	In Brazil, a <i>nota fiscal</i> with invoice information. See also <i>nota fiscal</i> .
<b>Object Configuration Manager (OCM)</b>	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.
<b>Object Librarian</b>	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.
<b>Object Librarian merge</b>	A process that blends any modifications to the Object Librarian in a previous release into the Object Librarian in a new release.
<b>Open Data Access (ODA)</b>	An interoperability model that enables you to use SQL statements to extract JD Edwards EnterpriseOne data for summarization and report generation.
<b>Output Stream Access (OSA)</b>	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.
<b>package</b>	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.
<b>package build</b>	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.”
<b>package location</b>	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.
<b>Package Workbench</b>	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.
<b>Pathcode Directory</b>	The specific portion of the file system on the EnterpriseOne development client where EnterpriseOne development artifacts are stored.

<b>patterns</b>	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).
<b>planning family</b>	A means of grouping end items whose similarity of design and manufacture facilitates being planned in aggregate.
<b>preference profile</b>	The ability to define default values for specified fields for a user-defined hierarchy of items, item groups, customers, and customer groups.
<b>print server</b>	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.
<b>pristine environment</b>	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.
<b>processing option</b>	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.
<b>production environment</b>	A JD Edwards EnterpriseOne environment in which users operate EnterpriseOne software.
<b>production-grade file server</b>	A file server that has been quality assurance tested and commercialized and that is usually provided in conjunction with user support services.
<b>Production Published Business Services Web Service</b>	Published business services web service deployed to a production application server.
<b>program temporary fix (PTF)</b>	A representation of changes to JD Edwards EnterpriseOne software that your organization receives on magnetic tapes or disks.
<b>project</b>	In JD Edwards EnterpriseOne, a virtual container for objects being developed in Object Management Workbench.
<b>promotion path</b>	<p>The designated path for advancing objects or projects in a workflow. The following is the normal promotion cycle (path):</p> <p>11&gt;21&gt;26&gt;28&gt;38&gt;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>
<b>proxy server</b>	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.
<b>published business service</b>	EnterpriseOne service level logic and interface. A classification of a published business service indicating the intention to be exposed to external (non-EnterpriseOne) systems.
<b>published business service identification information</b>	Information about a published business service used to determine relevant authorization records. Published business services + method name, published business services, or *ALL.

<b>published business service web service</b>	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).
<b>published table</b>	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.
<b>publisher</b>	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.
<b>pull replication</b>	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.
<b>QBE</b>	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.
<b>real-time event</b>	A message triggered from EnterpriseOne application logic that is intended for external systems to consume.
<b>refresh</b>	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.
<b>replication server</b>	A server that is responsible for replicating central objects to client machines.
<b>Rt-Addressing</b>	Unique data identifying a browser session that initiates the business services call request host/port user session.
<b>rules</b>	Mandatory guidelines that are not enforced by tooling, but must be followed in order to accomplish the desired results and to meet specified standards.
<b>quote order</b>	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.
<b>secure by default</b>	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.
<b>Secure Socket Layer (SSL)</b>	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.
<b>SEI implementation</b>	A Java class that implements the methods that declare in a Service Endpoint Interface (SEI).
<b>selection</b>	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.
<b>serialize</b>	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.
<b>Server Workbench</b>	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.
<b>Service Endpoint Interface (SEI)</b>	A Java interface that declares the methods that a client can invoke on the service.
<b>SOA</b>	Abbreviation for Service Oriented Architecture.
<b>soft coding</b>	A coding technique that enables an administrator to manipulate site-specific variables that affect the execution of a given process.
<b>source repository</b>	A repository for HTTP adapter and listener service development environment artifacts.
<b>spot rate</b>	An exchange rate entered at the transaction level. This rate overrides the exchange rate that is set up between two currencies.
<b>Specification merge</b>	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.
<b>specification</b>	A complete description of a JD Edwards EnterpriseOne object. Each object has its own specification, or name, which is used to build applications.
<b>Specification Table Merge Workbench</b>	An application that, during the Installation Workbench process, runs the batch applications that update the specification tables.
<b>SSL Certificate</b>	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.
<b>store-and-forward</b>	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.
<b>subscriber table</b>	Table F98DRSUB, which is stored on the publisher server with the F98DRPUB table and identifies all of the subscriber machines for each published table.
<b>superclass</b>	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.
<b>supplemental data</b>	<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>
<b>table access management (TAM)</b>	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.
<b>Table Conversion Workbench</b>	An interoperability model that enables the exchange of information between JD Edwards EnterpriseOne and third-party systems using non-JD Edwards EnterpriseOne tables.

<b>table conversion</b>	An interoperability model that enables the exchange of information between JD Edwards EnterpriseOne and third-party systems using non-JD Edwards EnterpriseOne tables.
<b>table event rules</b>	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.
<b>terminal server</b>	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.
<b>three-tier processing</b>	The task of entering, reviewing and approving, and posting batches of transactions in JD Edwards EnterpriseOne.
<b>three-way voucher match</b>	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.
<b>transaction processing (TP) monitor</b>	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.
<b>transaction processing method</b>	A method related to the management of a manual commit transaction boundary (for example, start, commit, rollback, and cancel).
<b>transaction set</b>	An electronic business transaction (electronic data interchange standard document) made up of segments.
<b>trigger</b>	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.
<b>triggering event</b>	A specific workflow event that requires special action or has defined consequences or resulting actions.
<b>two-way authentication</b>	An authentication mechanism in which both client and server authenticate themselves by providing the SSL certificates to each other.
<b>two-way voucher match</b>	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.
<b>user identification information</b>	User ID, role, or *public.
<b>User Overrides merge</b>	Adds new user override records into a customer's user override table.
<b>value object</b>	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.
<b>variance</b>	<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>

<b>versioning a published business service</b>	Adding additional functionality/interfaces to the published business services without modifying the existing functionality/interfaces.
<b>Version List merge</b>	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.
<b>visual assist</b>	Forms that can be invoked from a control via a trigger to assist the user in determining what data belongs in the control.
<b>vocabulary override</b>	An alternate description for a data dictionary item that appears on a specific JD Edwards EnterpriseOne form or report.
<b>wchar_t</b>	An internal type of a wide character. It is used for writing portable programs for international markets.
<b>web application server</b>	A web server that enables web applications to exchange data with the back-end systems and databases used in eBusiness transactions.
<b>web server</b>	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.
<b>Web Service Description Language (WSDL)</b>	An XML format for describing network services.
<b>Web Service Inspection Language (WSIL)</b>	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.
<b>web service proxy foundation</b>	Foundation classes for web service proxy that must be included in a business service server artifact for web service consumption on WAS.
<b>web service softcoding record</b>	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.
<b>web service softcoding template</b>	An XML document that provides the structure for a soft coded record.
<b>Where clause</b>	The portion of a database operation that specifies which records the database operation will affect.
<b>Windows terminal server</b>	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.
<b>wizard</b>	A type of JDeveloper extension used to walk the user through a series of steps.
<b>workbench</b>	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.
<b>work day calendar</b>	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.
<b>workflow</b>	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.
<b>workgroup server</b>	A server that usually contains subsets of data replicated from a master network server. A workgroup server does not perform application or batch processing.
<b>XAPI events</b>	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.
<b>XML CallObject</b>	An interoperability capability that enables you to call business functions.
<b>XML Dispatch</b>	An interoperability capability that provides a single point of entry for all XML documents coming into JD Edwards EnterpriseOne for responses.
<b>XML List</b>	An interoperability capability that enables you to request and receive JD Edwards EnterpriseOne database information in chunks.
<b>XML Service</b>	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.
<b>XML Transaction</b>	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.
<b>XML Transaction Service (XTS)</b>	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.
<b>Z event</b>	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.
<b>Z table</b>	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.
<b>Z transaction</b>	Third-party data that is properly formatted in interface tables for updating to the JD Edwards EnterpriseOne database.



# Index

## A

- Add button 18
- additional documentation viii
- All Grid Recs Deleted from DB event 21
- application event rules 4
- application fundamentals vii
- assignment 27, 28

## B

- BrowsER 31
- business function
  - attaching to an event 29
- Button Clicked event for Add Button 18
- Button Clicked event for Delete Button 19
- Button Clicked event for Select Button 17
- buttons
  - tool bar 27

## C

- comments, submitting xii
- common fields xii
- contact information xii
- Control is Exited event 7
- controls
  - disabling/enabling 8
  - hiding/showing 8, 25
- cross-references xi
- Customer Connection website viii

## D

- data structure 5
- data structure object codes 5
- data structure processing 6
- database triggers 4
- debugger 35
  - breakpoint manager 38
  - event rules window 37
  - object browse window 36
  - search combo box 39
  - step by step instructions 40
  - variable tree and watch window 37
- debugger features 33
- debugging event rules 33
- debugging strategies 34
- Delete button 19

- Delete Grid Rec From DB—After event 21
- Delete Grid Rec From DB—Before event 21
- Delete Grid Rec Verify—After event 21
- Delete Grid Rec Verify—Before event 20
- Dialog is Initialized event 8
- documentation
  - printed viii
  - related viii
  - updates viii

## E

- embedded event rules 4
- event 3
  - All Grid Recs Deleted from DB 21
  - Button Clicked for Add Button 18
  - Button Clicked for Delete Button 19
  - Button Clicked for Select Button 17
  - Control is Exited 7
  - Delete Grid Rec From DB—After 21
  - Delete Grid Rec From DB—Before 21
  - Delete Grid Rec Verify—After 21
  - Delete Grid Rec Verify—Before 20
  - Dialog is Initialized 8
  - Grid Record is Fetched 11
  - Last Grid Record Has Been Read 16
  - Post Dialog is Initialized 8
  - Write Grid Line—After 14
  - Write Grid Line—Before 13
- event information 27
- event rules 3
  - application 4
  - debugging 33
  - design 23, 26
  - embedded 3, 4
  - logic 23
  - named 3, 4
  - table 4
  - validation 25
- event rules runtime data structure 5
- event rules runtime processing 4

## F

- filter fields
  - loading for SQL SELECT 9

