



Menu Design

Use Menu Design (P0082) to create, change, delete, copy, and filter menus and menu selections. Menu Design assists you in the following:

- Managing menus and menu selections
- Managing text overrides
- Defining runtime messages for menu selections
- Indicating the consequences of using particular menu selections
- Copying menu selections

Menu design contains the following topics:

- Understanding menus
- Working with menus
- Working with menu selections
- Working with menu selection revisions



Understanding Menus

A menu is the entry point for running reports and applications. The Menu Master table (F0082) stores the following information, which identifies and characterizes the menu:

- Identifying information (ID and related system code)
- Level of detail
- Menu classification
- Menu Text Override (F0083)

Understanding menus contains the following topics:

- Menu filtering
- Menu design tables

Menu Filtering

OneWorld automatically filters menus based on your user ID so that only menu selections that apply to your job appear on your workstation. This feature allows you to maintain one set of menus (one database) with hundreds of menu selections, but you see only those menus that apply to your job.

Menus are filtered so that the following selections do not appear:

- Menu selections that you or other users do not have authority to access – for example, Employee Information in Human Resources Management.
- Menu selections that are country-code-specific. If your user profile country code matches the country code for that menu, then the selection displays. For example, menu selections that pertain only to Canadian users, such as Canadian tax-related selections, appear only to Canadian users.
- WorldVision menu selections that are not installed on your workstation. For example, if WorldVision (the J.D. Edwards AS/400 product) is not installed on your workstation, that selection does not appear on the menu. You can distinguish a WorldVision menu selection from a OneWorld menu selection by looking at the Job to Execute number. A WorldVision Job to Execute number begins with a J – for example, J3413.

Menu Design Tables

Menu Design stores information in the following tables:

Menu Master (F0082)	Defines all menus but not the selections
Menu Selection (F00821)	Contains the type of selection to be executed, selection consequences, and version information
Menu Text Override (F0083)	Contains menu selection descriptions
Menu Path (F0084)	Contains the menu selection icons

Working with Menus

Menus are the entry point to J.D. Edwards applications and reports. To access an application or report from a menu, the application or report must be attached to a menu selection on the menu.

Working with menus contains the following tasks:

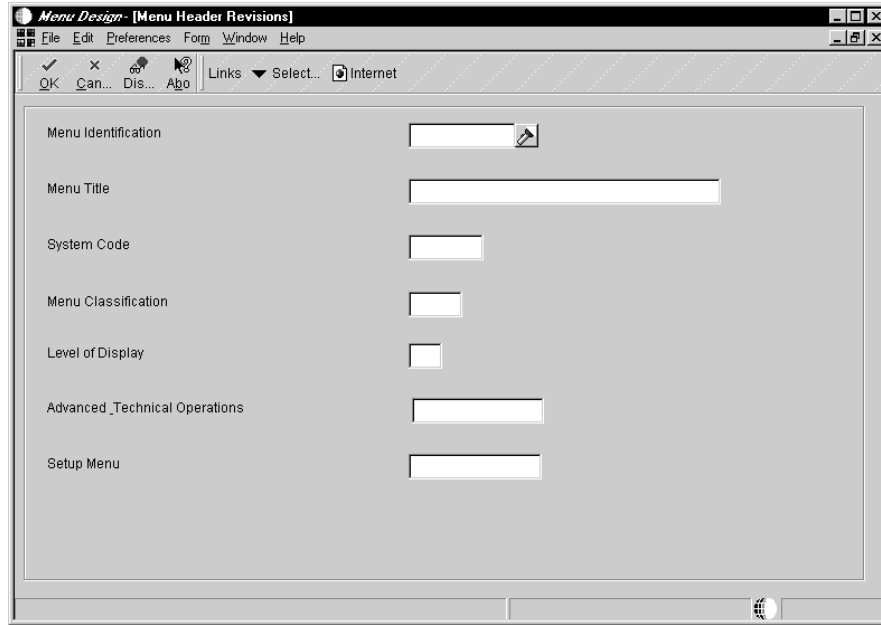
- Defining a new menu
- Reviewing selections for a menu
- Printing a menu report

Defining a New Menu

You can define a menu to include selections that enable you to access the applications and reports that you need from one location.

To define a new menu

1. On System Administration Tools (GH9011), choose Menu Design (P0082).
2. On Work With Menus, click Add.



3. On Menu Header Revisions, complete the following fields:

- Menu Identification
- Menu Title
- System Code
- Menu Classification
- Level of Display
- Advanced & Technical Operations
- Setup Menu

Field	Explanation								
Menu Identification	<p>The menu name, which can include up to nine characters. J.D. Edwards standards are:</p> <ul style="list-style-type: none"> • Menu numbers are preceded with a G prefix. • The two characters following the prefix are the system code. • The next characters further identify the menu. • The 4th character specifies a specific skill level. • The 5th character distinguishes two menus of the same system with the same skill level. <p>For example, the menu identification G0911 specifies the following:</p> <table style="margin-left: 20px;"> <tr> <td>G</td> <td>Prefix</td> </tr> <tr> <td>09</td> <td>System code</td> </tr> <tr> <td>1</td> <td>Display level/skill level</td> </tr> <tr> <td>1</td> <td>First menu</td> </tr> </table>	G	Prefix	09	System code	1	Display level/skill level	1	First menu
G	Prefix								
09	System code								
1	Display level/skill level								
1	First menu								

Field	Explanation
Menu Classification	The menu classification indicates the type of a menu. For example: a JDE Master menu or Company Master menu.
Level of Display	<p>The Level of Display field contains a number or letter identifying the level at which menus and processing options are displayed. The levels of display are as follows:</p> <ul style="list-style-type: none"> A Product Groups (for example, Job Cost, Manufacturing) B Major Products (for example, GL, AP) 1 Basic Operations 2 Intermediate Operations 3 Advanced Operations 4 Computer Operations 5 Programmers 6 Senior Programmers.
Advanced & Technical Operations	<p>For World:</p> <p>The advanced operations key is used to direct the menu selection '27' (Advanced Operations) to the appropriate menu. This menu designation must be preceded with an asterisk (*). For example, the General Accounting Advanced Operations menu would be *A093.</p> <p>..... <i>Form-specific information</i></p> <p>For OneWorld:</p> <p>Each menu may optionally have an Advanced & Technical Operations menu to which it is associated and it is displayed as the last menu selection. This is normally used to categorize more advanced tasks.</p>
Setup Menu	<p>For World:</p> <p>The technical operations control key is used to direct the menu selection '29' (Technical Operations) to the appropriate menu. This menu designation must be preceded with an asterisk (*). For example, the General Accounting Technical Operations Menu would be *A094.</p> <p>..... <i>Form-specific information</i></p> <p>For OneWorld:</p> <p>The menu you enter in this field is associated with the menu item description: Setup Menu. This menu is automatically displayed at the bottom of the menu you specify in the Menu Identification field.</p>

Reviewing Selections for a Menu

You can review the selections included on a specific menu from the Work With Menus form or from the Menu Header Revisions form.

► **To review selections for a menu**

1. On System Administration Tools (GH9011), choose Menu Design (P0082).
2. On Work With Menus, locate and choose a menu that you wish to review.

The Work With Menu Selections form appears, from which you can view and edit selections for a specific menu.

Printing a Menu Report

You can print a report that lists the menus. You can print menus only or menus and menu selections. To print a menu report, choose Menu Print from the Form menu.

Example: Print Menus Report

Menu	Description	SY	LOD	Class				
DEMO	APPLICATIONS		00					
Sel #	Description	Job To	Form	Version	Ctry	Appl	Run Time	SC
		Execute	Name			Ovrd	Message	
0	APPLICATIONS							1
1	Journal Entries	P0911		ZJDE0001				3
2	General Journal Posting	R09801						3
3	Company Numbers and Names	P0010						3
4	Budget vs.Actual Comparison	P09210		ZJDE0001				2
5	Account Ledger Inquiry	P0911L						1
6	Original Budget Update	P14102		ZJDE0001				3
7	Online Consolidations	P09218		ZJDE0001				1
8	Manufacturing Variance Inquiry	P3102		ZJDE0001				1
11	Bill of Material	P3002		ZJDE0001				1
12	Routings	P3003		ZJDE0001				1
13	Forecasting	P3460		ZJDE0001				1
14	Customer Service	P4210		ZJDE0001				1
15	Planners Workbench	P3401		ZJDE0003				1
16	Schedulers Workbench	P31225		ZJDE0001				1

Working with Menu Selections

Work With Menu Selections displays available selections for the selected menu. Use Work With Menu Selections when you are:

- Adding or changing a menu selection
- Adding an application to a menu
- Adding or changing Web addresses on OneWorld Explorer Help
- Creating a Web view subheading on a menu
- Linking menus
- Creating fast path selections

Adding or Changing a Menu Selection

To add a menu selection for an application or report, you must first name the menu selection by assigning a description and unique selection number before you can add a menu selection for an application or report to the menu.

After naming a menu selection, indicate the selection type and define it.

- Selection Type specifies the type of program that is executed for the menu selection. You use OneWorld Application, OneWorld Report, World Vision, or Windows Application to execute a specific application, report, or program.
- The Subheading selection type does not perform an action. You use it to logically group menu selections on the menu. Subheading selections appear on the menu only in Web view.
- You use the Menu selection type to call another menu.

Note: When you delete a menu, you delete the menu and any menu selections available on that menu. The applications called by the menu selections are not deleted, and you can access these applications from other menus.

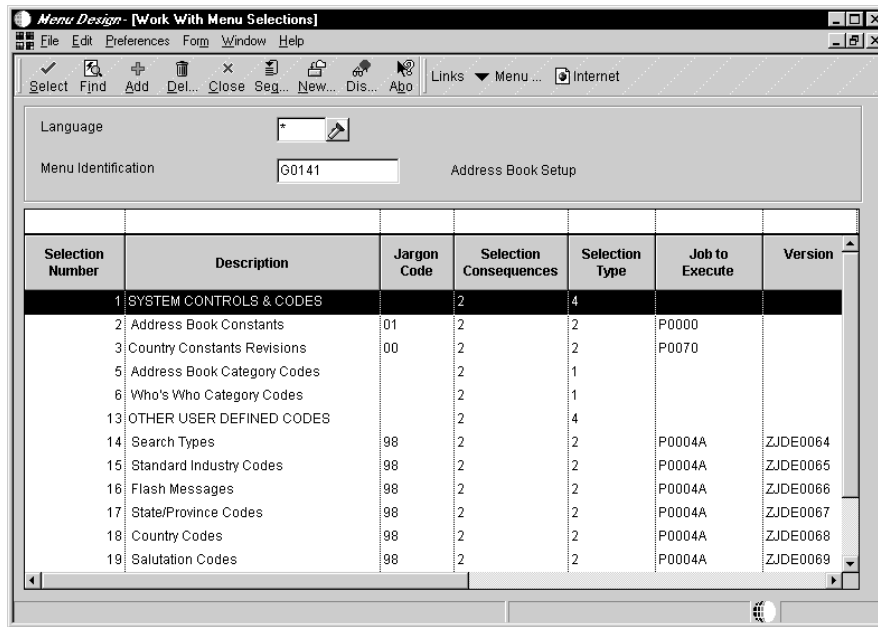
Adding or changing a menu selection consists of the following:

- Naming a menu selection

- Defining the menu selection

► **To name a menu selection**

1. On System Administration Tools (GH9011), choose Menu Design (P0082).
2. On Work With Menus, find and choose the menu that has a selection that you want to name.



3. On Work With Menu Selections, click Add or choose an existing selection to change.

4. On Menu Selection Revisions, complete the following required fields:

- Selection Number
- Selection Description
- Selection Consequences

If you chose an existing selection to change, the Selection Number field is disabled.

5. The following fields are optional; complete them if necessary:

- Jargon
- Country Code

If the base language is a double-byte language, a Search Description field is shown below the Country Code field. Enter the single-byte search description to be used by Menu Word Search. Menu Word Search uses only single-byte search descriptions.

Field	Explanation
Selection Number	Used to determine the order of menu items and allow them to be selected by this number.
Selection Description	Contains menu titles and menu selection descriptions.
Selection Consequences	The selection consequences tell the user what the consequences are in taking that specific menu selection.

Field	Explanation
Jargon	A code used to designate the reporting system number for entering specific text or “jargon”. See User Defined Codes, system code '98', record type 'SY' for a list of valid values.
Country Code	The Menu Country/Region Codes field contains the region code (3 bytes) for all 24 menu selections for each menu record. This region code is used to mask those international selections that are country specific; i.e. 1099 processing in the US and VAT tax processing in Europe.

To define the menu selection

1. On System Administration Tools (GH9011), choose Menu Design (P0082).
2. Find and choose the menu for which you want to define a menu selection.
3. On Work With Menu Selections, choose an existing selection to define.
4. On Menu Selection Revisions, indicate one of the following Selection Types:
 - OneWorld Application
 - OneWorld Report
 - World Vision
 - Windows Application
 - Sub-Heading
 - Menu
5. From the Form menu, choose **Define**.

A form that is specific to the selection type appears.
6. Define options for the selection type indicated.
7. Click OK.

Adding an Application to a Menu

You can add OneWorld applications and reports, WorldVision applications, and Windows applications to a menu. You can also link a menu to another menu.

Complete the following tasks:

- Add a OneWorld application to a menu
- Add a OneWorld report selection to a menu

- Add a WorldVision application to a menu
- Add a Windows application to a menu

► To add a OneWorld application to a menu

You can use this procedure to add a OneWorld application created in Forms Design as a menu selection.

1. On System Administration Tools (GH9011), choose Menu Design (P0082).
2. Find and choose the menu for which you want to add a OneWorld application.
3. On Work With Menu Selections, click Add.
4. On Menu Selection Revisions, complete the following required fields:
 - Selection Number
 - Selection Description
 - Selection Consequences

If you chose an existing selection to change, the Selection Number field is disabled.

5. Click the OneWorld Application option, and from the Form menu, choose Define.

6. On OneWorld Application, complete the following fields:
 - Application

- Form Name

You can use this field to define a specific entry point for the OneWorld application. If you leave this blank, the program's first entry point is used.

- Option Code
- Version
- Application Type

7. From the form menu, choose Application to view and choose from a list of available applications. Likewise, from the Form menu, you can choose Versions to search for available versions.

Field	Explanation														
Option Code	<p>For World, this code specifies the function of a menu selection using the DREAM Writer when F18 is pressed. F18 may be locked out by simply replacing code 1 with 3 or code 2 with 4. This code, in conjunction with the version number and the option key, provide the following functions:</p> <p>Code</p> <table data-bbox="706 1018 1356 1312"> <tr> <td>1</td> <td>version — mandatory; option key — form i.d. F18 displays processing options. Selection = blind DREAM Writer execution.</td> </tr> <tr> <td>2</td> <td>version — blank; option key — form i.d. F18 displays DREAM Writer versions list. Selection = DREAM Writer versions list.</td> </tr> <tr> <td>2</td> <td>version — not blank; option key — form i.d. F18 displays DREAM Writer versions list. Selection = blind execution, batch.</td> </tr> </table> <p>Review the HELP instructions for Menu Information (Menu Locks) (P0090) for a detailed explanation of codes related to job submission and control.</p> <p>For OneWorld, this code specifies whether the user will be prompted for additional information prior to running the application. Available values are:</p> <table data-bbox="706 1522 1128 1648"> <tr> <td>0</td> <td>No processing options</td> </tr> <tr> <td>1</td> <td>Blind execution (no prompt)</td> </tr> <tr> <td>2</td> <td>Prompt for version</td> </tr> <tr> <td>3</td> <td>Prompt for values</td> </tr> </table>	1	version — mandatory; option key — form i.d. F18 displays processing options. Selection = blind DREAM Writer execution.	2	version — blank; option key — form i.d. F18 displays DREAM Writer versions list. Selection = DREAM Writer versions list.	2	version — not blank; option key — form i.d. F18 displays DREAM Writer versions list. Selection = blind execution, batch.	0	No processing options	1	Blind execution (no prompt)	2	Prompt for version	3	Prompt for values
1	version — mandatory; option key — form i.d. F18 displays processing options. Selection = blind DREAM Writer execution.														
2	version — blank; option key — form i.d. F18 displays DREAM Writer versions list. Selection = DREAM Writer versions list.														
2	version — not blank; option key — form i.d. F18 displays DREAM Writer versions list. Selection = blind execution, batch.														
0	No processing options														
1	Blind execution (no prompt)														
2	Prompt for version														
3	Prompt for values														
Version	<p>Version identifies a specific set of data selection and sequencing settings for the application. Versions may be named using any combination of alpha and numeric characters. Versions that begin with 'XJDE' or 'ZJDE' are set up by J.D. Edwards.</p>														

Field	Explanation
Application Type	Complete with a user-defined, alphanumeric value. This field exists in the JDE user profile and within each menu and menu selection record. When security is active, the value of this field in the user profile is compared with the value in the corresponding menu lock. The values must be equal in the user profile and menu lock to access the menu. A blank in this field in the user profile gives the user all authority. A blank in this field in the menu record indicates no security exists on this menu.

To add a OneWorld report selection to a menu

You can add a report created in the OneWorld Report Design tool as a menu selection.

1. On System Administration Tools (GH9011), choose Menu Design (P0082).
2. Find and choose the menu for which you want to add a OneWorld report selection.
3. On Work With Menu Selections, click Add.
4. On Menu Selection Revisions, complete the following required fields:
 - Selection Number
 - Selection Description
 - Selection Consequences

If you chose an existing selection to change, the Selection Number field is disabled.

5. Click the OneWorld Report option, and from the Form menu, choose Define.
6. On OneWorld Report, complete the following fields:
 - Batch Application
 - Version
7. Select the processing options you want users to have:
 - Blind Execution
 - Values
 - Versions
 - Data Selection

Use Form menu options to view and choose from a list of reports and versions.

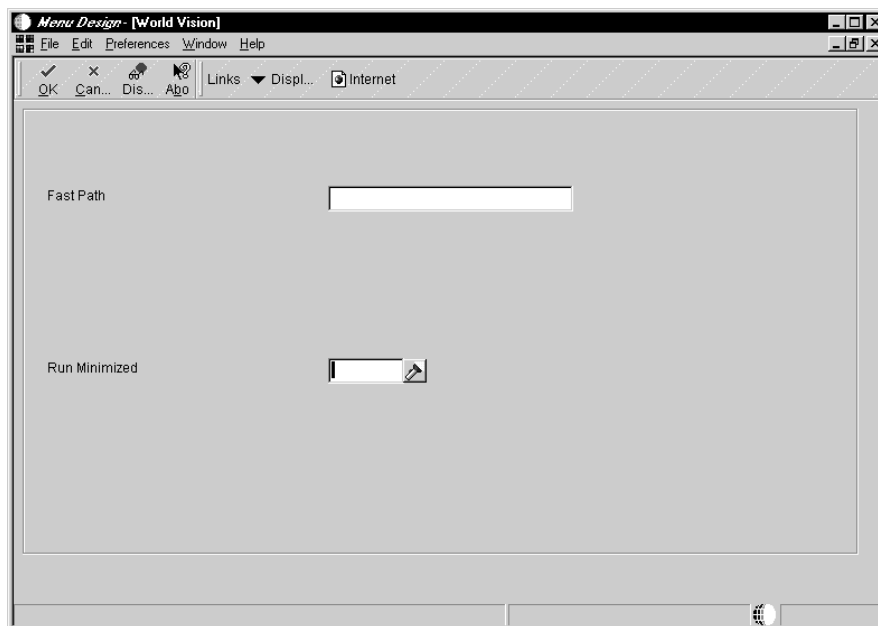
► To add a WorldVision application to a menu

You can add a WorldVision application as a menu selection.

1. On System Administration Tools (GH9011), choose Menu Design (P0082).
2. Find and choose the menu for which you want to add a WorldVision selection.
3. On Work With Menu Selections, click Add.
4. On Menu Selection Revisions, complete the following required fields:
 - Selection Number
 - Selection Description
 - Selection Consequences

If you chose an existing selection to change, the Selection Number field is disabled.

5. Click the WorldVision option, and from the Form menu, choose Define.



6. On WorldVision, complete the following fields:
 - Fast Path
 - Run Minimized

Field	Explanation
Fast Path	<p>The path field contains the path used for client based menus. The path describes where the application is located on your computer or network. A path includes the drive, folders, and subfolders that contain the application to be executed.</p> <p>To specify the path for a World Vision menu selection, the path includes the selection number, slash, menu.</p>
Run Minimized	<p>The Run Minimized flag determines if a Windows application is to be minimized to an icon when you open it.</p>

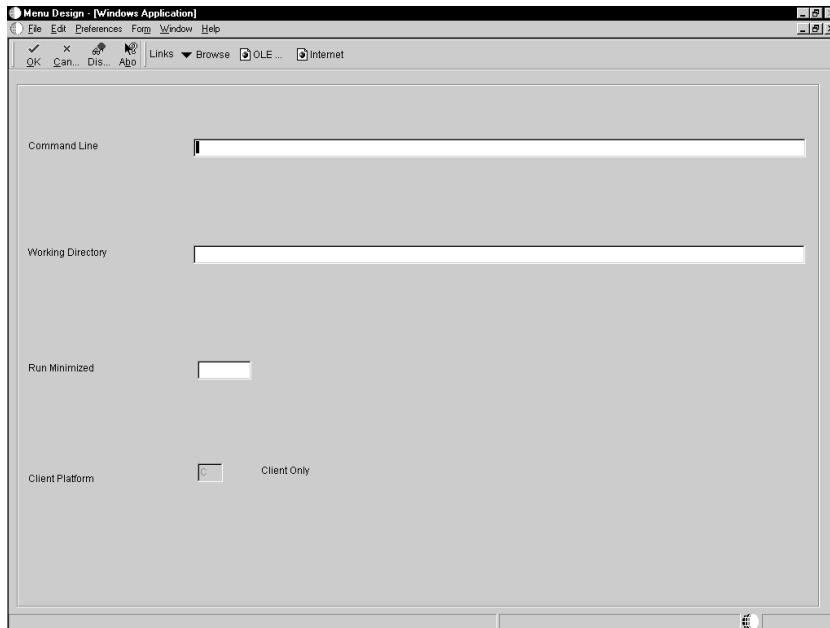
To add a Windows application to a menu

You can add any Windows applications as menu selections. For example, you can add Windows programs such as Calendar, Clock, Note Pad, or Write.

1. On System Administration Tools (GH9011), choose Menu Design (P0082).
2. Find and choose the menu for which you want to add a Windows application.
3. On Work With Menu Selections, click Add.
4. On Menu Selection Revisions, complete the following required fields:
 - Selection Number
 - Selection Description
 - Selection Consequences

If you chose an existing selection to change, the Selection Number field is disabled.

5. Click the Windows Application option, and from the Form menu, choose Define.



6. On Windows Application, complete the following fields, or click the Browse button to search for the Windows application:

- Command Line

Enter the executable of the Windows application that you want to add to the OneWorld menu, such as winword.exe.

- Working Directory

Enter the path on your local machine where the Windows application resides.

- Run Minimized

Field	Explanation
Command Line	<p>The path field contains the path used for client based menus. The path describes where the application is located on your computer or network. A path includes the drive, folders, and subfolders that contain the application to be executed.</p> <p>To specify the path for a World Vision menu selection, the path includes the selection number, slash, menu.</p>

Field	Explanation
Working Directory	<p>The path field contains the path used for client based menus. The path describes where the application is located on your computer or network. A path includes the drive, folders, and subfolders that contain the application to be executed.</p> <p>To specify the path for a World Vision menu selection, the path includes the selection number, slash, menu.</p>

Adding or Changing Web Addresses on OneWorld Explorer Help

You can add Web addresses or change some of the addresses that appear on the Help menu of OneWorld Explorer. From the Help menu on OneWorld Explorer, there is an option called “J.D. Edwards on the Web.” From this option, a list of Web addresses appear, and there is a line that separates the addresses. The addresses above this line, which include J.D. Edwards Home Page and Contact Us, are hard-coded into OneWorld, which means you cannot change them. You can, however, change the Web addresses below the line or add your own Web addresses to the list.

► To add or change Web addresses on OneWorld Explorer Help

1. On System Administration Tools (GH9011), choose Menu Design (P0082).
2. On Work With Menus, in the Menu ID query-by-example field, type HELP, then click Find.

The Web Access menu appears.

3. Choose the Web Access menu and click Select.
4. On Work With Menu Selections, click Add to add a new Web address, or choose a row and click Select to change an existing Web address.
5. On Menu Selection Revisions, complete the following required fields:
 - Selection Number
 - Selection Description
 - Selection Consequences

If you chose an existing selection to change, the Selection Number field is disabled.

6. Click the Windows Application option, and from the Form menu, choose Define.

7. On Windows Application, complete the following fields:

- Command Line

Enter the Web address that you want to add to the Help menu, such as <http://www.jdedwards.com>.

- Working Directory

Because you are entering a Web address, you do not need to complete this field.

- Run Minimized

Creating a Web View Subheading on a Menu

Use Web subheadings to logically group menu selections on the menu. Subheadings appear on the menu in Web view only and do not perform an action.

To create a Web view subheading selection

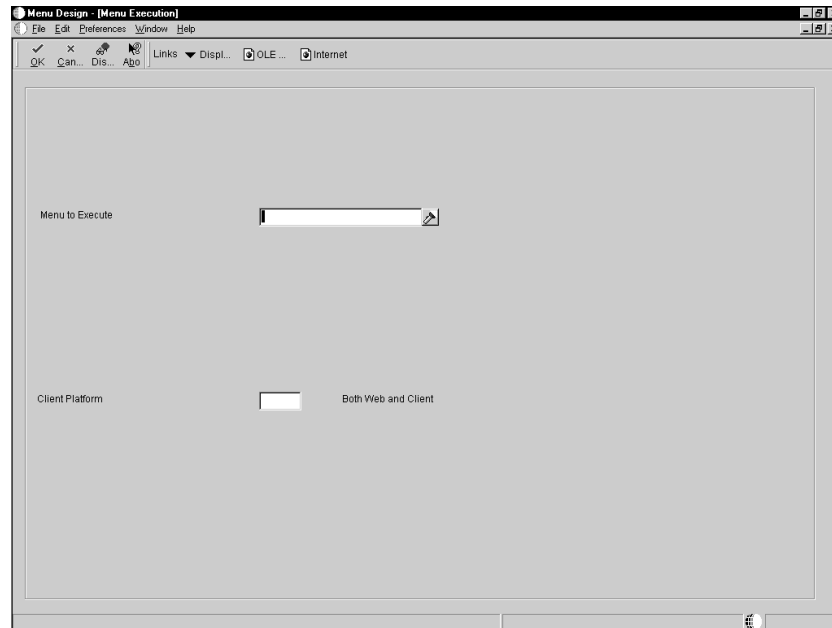
1. On System Administration Tools (GH9011), choose Menu Design (P0082).
2. Find and choose the menu for which you want to create a subheading.
3. On Work With Menu Selections, click Add.
4. On Menu Selection Revisions, complete the following field:
 - Selection Number
5. Click the Web View Sub-Heading option.
6. Click OK to complete the Web view subheading assignment.

Linking Menus

You can add a menu selection that displays another menu.

To link another menu to a menu

1. On System Administration Tools (GH9011), Choose Menu Design (P0082).
2. Find and choose the menu for which you want to add a selection.
3. On Work With Menu Selections, click Add.
4. On Menu Selection Revisions, click the Menu option.
5. From the Form menu, choose Define.



6. On Menu Execution, complete the following field or click the visual assist to search for menus:
- Menu to Execute
 - Client Platform

Field	Explanation
Menu to Execute	The specific menu to be executed as a selection on a menu.

Creating Fast Path Selections

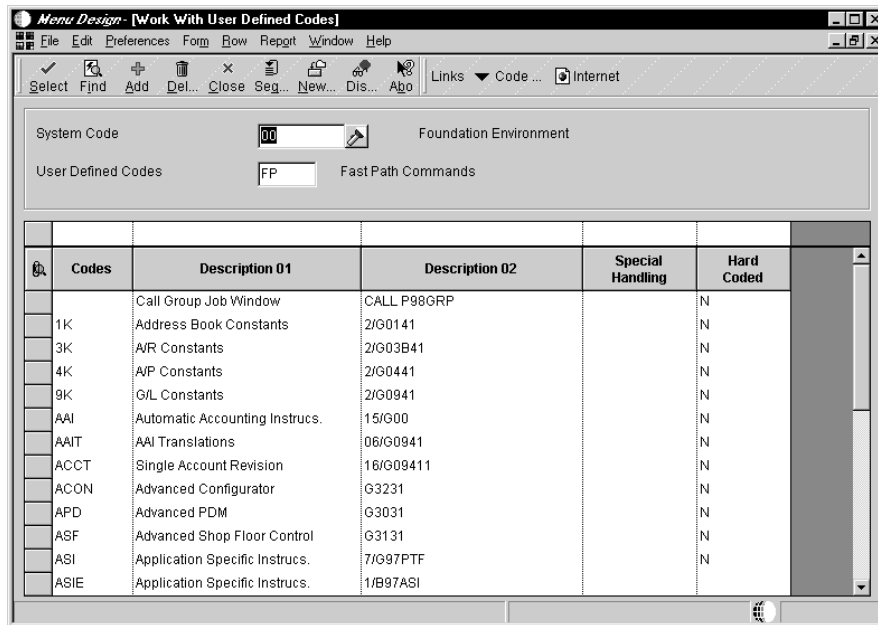
You can quickly move among menus and applications by using fast path commands. A fast path command is:

- An abbreviation that is either shipped with J.D. Edwards demo data or which you define to suit your business environment. For example, the fast path OL takes you to the application Object Librarian so that you can work with OneWorld objects.
- A combination of menu selection and menu number. For example 2/G01 (menu selection number 2 on menu number G01) takes you to Work With Addresses in Address Book. As you become more familiar with OneWorld menu abbreviations, you might find fast path a quicker way to navigate to an application.

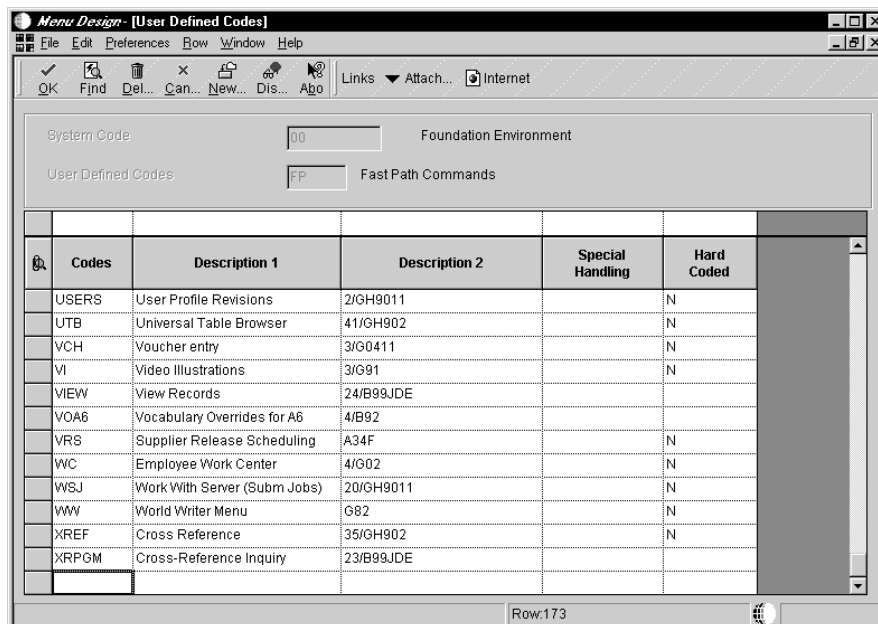
You can set up your own a fast path abbreviations to access frequently used applications.

► **To create a fast path selection**

1. On System Administration Tools (GH9011), choose Menu Design (P0082).
2. On Work With Menus, find and choose the menu that has a selection for which you want to create a fast path.
3. On Work With Menu Selections, from the Form menu, choose Fast Path Revs.



4. On Work With User Defined Codes, click Add.



5. On User Defined Codes, click inside the grid, then press the Ctrl and End keys to display the bottom of the grid.
6. To add a user defined code for a new fast path, complete the following required fields in the last row of the grid:
 - Codes
 - Description 1
 - Description 2

You enter the abbreviation for the fast path in the Code field. Enter the description of the abbreviation, such as the name of the menu selection, in the Description 01 field. Enter the selection number and menu number in the Description 02 field.

To determine the selection number for the fast path you created (for example, selection number 2 on menu G01), use Work With Menu Selections. Do not count the menu selections in OneWorld Explorer because the menu might be filtered.

Working with Menu Selection Revisions

You can revise your existing menu selections to change menu text for languages, change menu selection text, or renumber a menu selection. This chapter describes the following tasks:

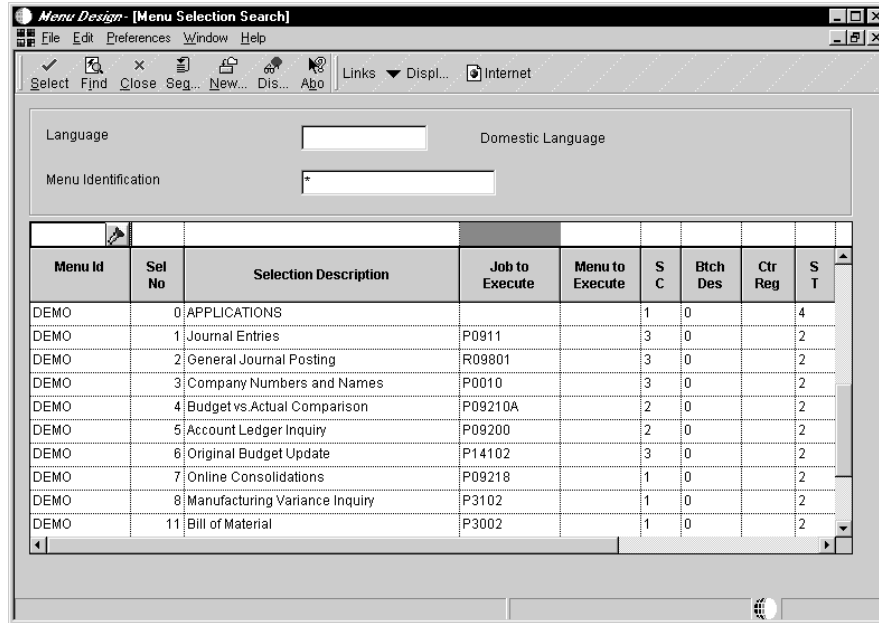
- Copying a menu selection
- Changing menu text for languages
- Changing menu selection text
- Renumbering a menu selection

Copying a Menu Selection

You can copy an existing menu selection and attach it to another menu.

To copy a menu selection

1. On System Administration Tools (GH9011), choose Menu Design (P0082).
2. On Work With Menus, find and choose the menu that has a selection you want to copy.
3. On Work With Menu Selections, click Add.
4. On Menu Selection Revisions, enter the new selection number and choose Copy from the Form menu.



5. On Menu Selection Search, choose an existing menu selection.

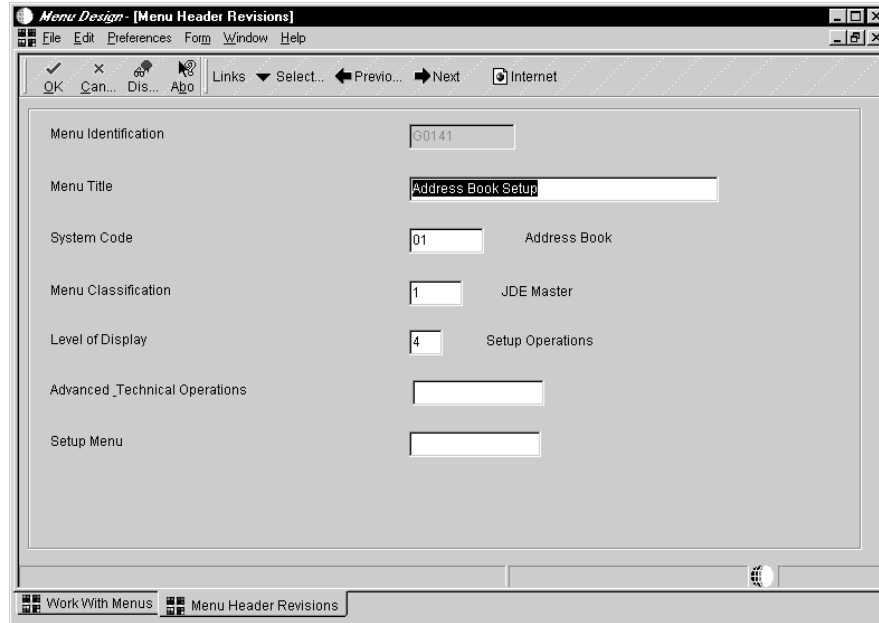
The selection description, consequences, and type are inserted into the newly added menu selection.

Changing Menu Text for Languages

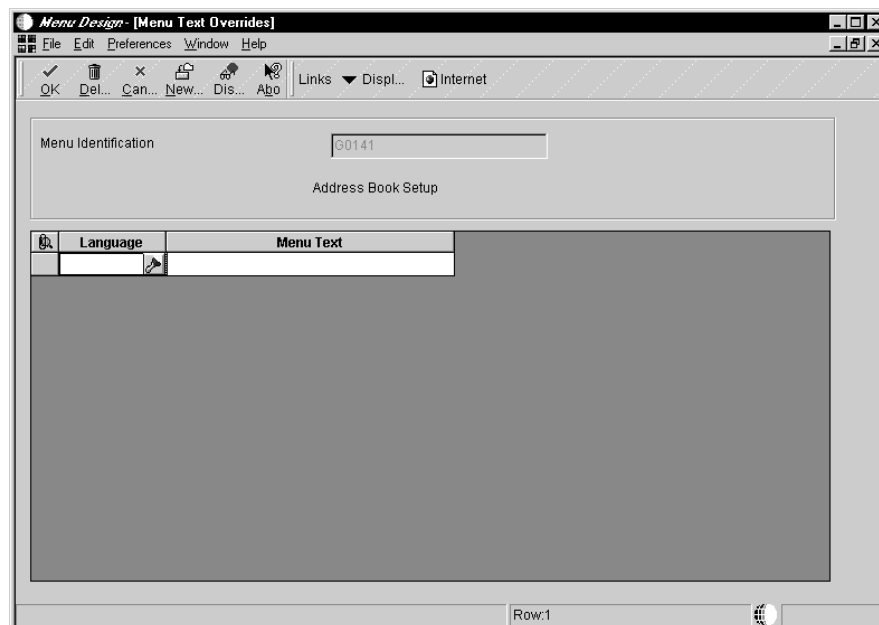
You use Title Overrides to change the language and description of a menu selection.

► To change menu text for languages

1. On System Administration Tools (GH9011), choose Menu Design (P0082).
2. On Work With Menus, find and choose the menu for which you want to change menu text.
3. On Work With Menus, choose Header from the Row menu selection.



4. On Menu Header Revisions, from the Form menu, choose Title Overrides.



5. On Menu Text Overrides, enter the desired language code (such as S for Spanish) and text description (such as the Spanish description of the menu selection) in the following fields and click OK.
 - Language
 - Menu Text

Field	Explanation
Language	A user defined code (system 01/type LP) that specifies a language to use in forms and printed reports. Before any translations can become effective, a language code must exist at either the system level or in your user preferences.
Menu Text	Contains menu titles and menu selection descriptions.

Changing Menu Selection Text

You use Text Overrides to change a menu selection's text.

To change menu selection text

1. On System Administration Tools (GH9011), choose Menu Design (P0082).
2. On Work With Menus, find and choose the menu for which you want to change menu's selection text.
3. On Work With Menu Selections, choose the menu selection you want to change and click Select.
4. On Menu Selection Revisions, choose Text Overrides from the Form menu.
5. On Menu Text Overrides, complete the following fields and click OK:
 - Language
 - Menu Text

Changes in menu text are not displayed until the changed menu is closed and reopened. You can also choose Refresh from the View menu to update the menu text.

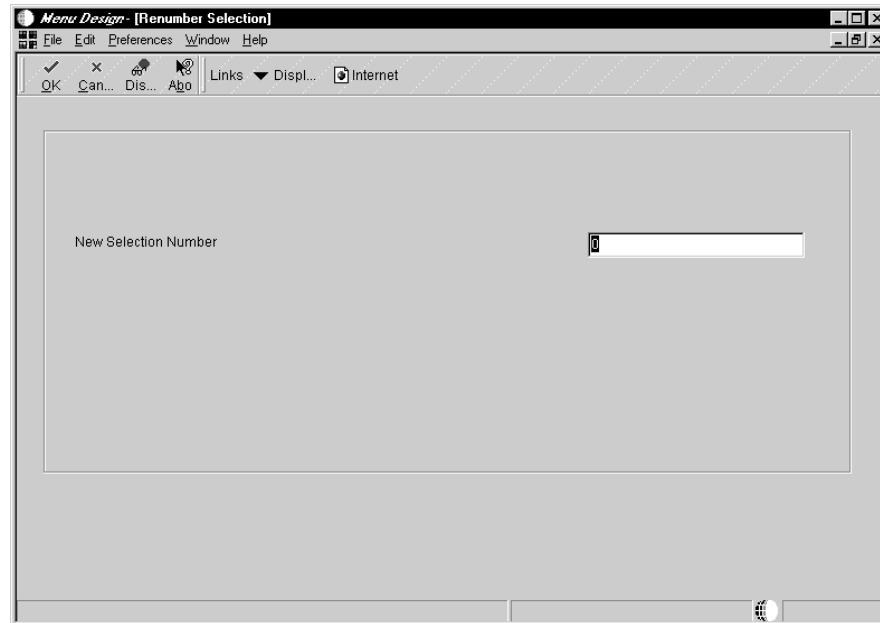
Renumbering a Menu Selection

You can renumber menu selections from both Work With Menu Selections and Menu Selection Revisions. You can edit each selection to change the sequence of selections on a menu. You cannot rearrange menu selections by clicking and dragging them.

To renumber a menu selection

1. On System Administration Tools (GH9011), choose Menu Design (P0082).

2. On Work With Menus, find and choose the menu for which you want to change the menu's selection number.
3. On Work With Menu Selections, choose the menu selection that you want to change and click Select.
4. On Menu Selection Revisions, from the Form menu, choose Renumber.



5. Complete the following field, and click OK.
 - New Selection Number

Field	Explanation
New Selection Number	Used to determine the order of menu items and allow them to be selected by this number.



Tips of the Day

Tips of the day are a sets of short, informational text that appear in a special form each time the user launches an application or accesses a form. Tips of the day appear sequentially, so the user can browse through the tips. When the user closes the tip form, the system records where in the tip sequence the user is and displays the next tip when the user launches the object again.

J.D. Edwards provides tips of the day with many OneWorld applications. You can change these tips sets or create your own. Tips of the day contains the following topic:

- Working with tips of the day



Working with Tips of the Day

In OneWorld, tips of the day are the glossary texts of data dictionary items. You create one data dictionary item for each tip. Since you can translate data dictionary glossaries, tips of the day can appear in different languages.

You can associate tips with an application, a form, or an application version. The tips appear in the order you specify, and you can override a user's option to turn off the tip of the day feature for the tip set.

After you have associated tips with an object, you can rearrange the tip order, add new tips, or delete existing ones from the tip set.

Working with Tips of the Day contains the following tasks:

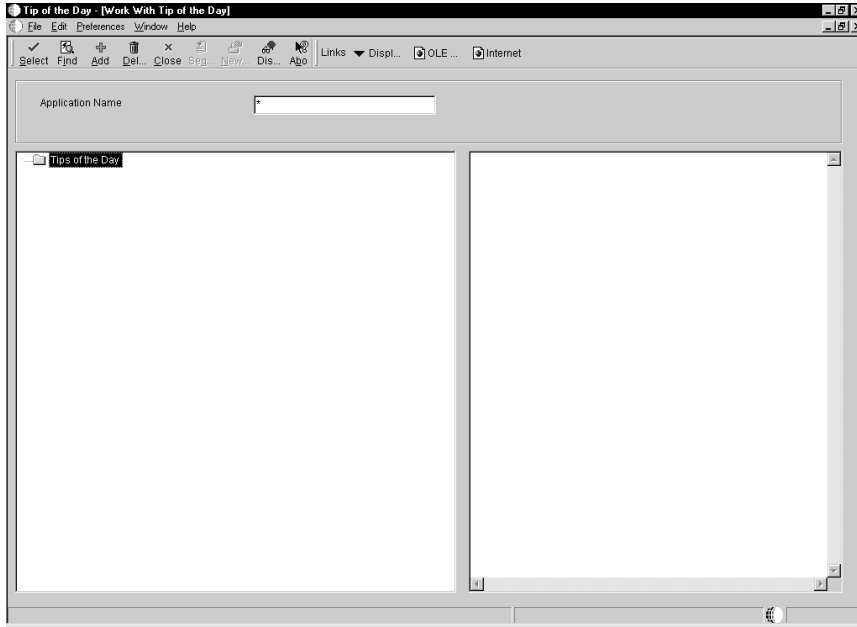
- Working with the Tip of the Day utility
- Adding tips of the day to an object

Before You Begin

- Create a data dictionary item with glossary text for each tip of the day. See *Creating a Data Item* for detailed instructions about creating data dictionary items.

▶ **To work with the Tip of the Day utility**

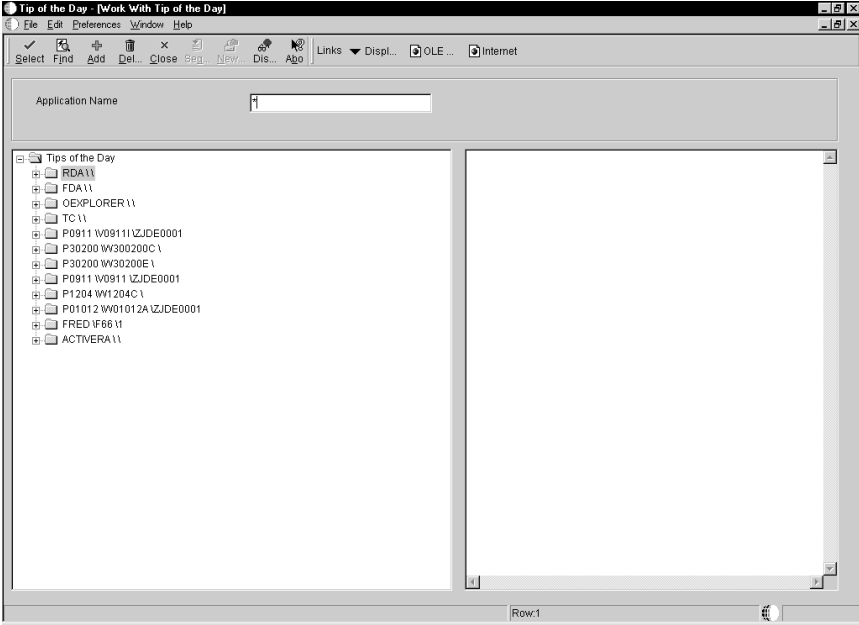
1. From the System Administration Tools menu (GH9011), double-click Tip of the Day.



2. On Work With Tip of the Day, click Find to see the objects to which tips have been added.

Enter an application name in the Application Name field to limit your search.

A tree-style file structure appears on the left side of the form. Each object has its own folder. Tips can be associated with applications, forms, or application versions. Folder names include the application, form, and application version, in that order. For example, RDA\\ indicates the RDA application alone. P0911\V0911I\ZJDE0001 indicates form V0911I in the ZJDE0001 version of application P0911.

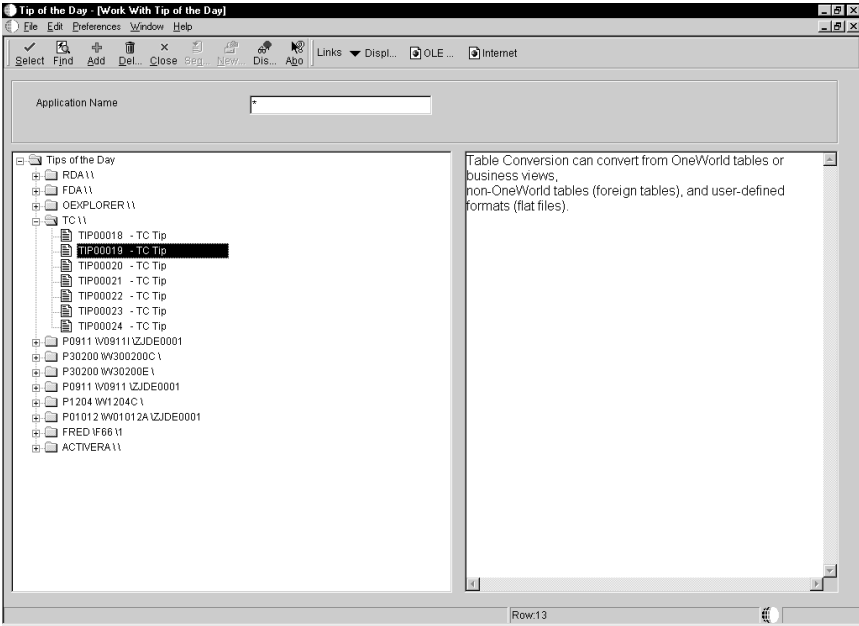


- 3. To see the specific tips associated with an object, expand the folder for the object.

The data dictionary items associated with the object as tips appear in the file structure.

- 4. To view the text for a tip, click the tip.

The glossary text associated with the data dictionary item appears on the right side of the form.



5. To change the order of the tips or to add or delete tips for an object that already has tips associated with it, double-click one of the tips under the object.

The Tips of the Day Revisions form appears.

To add tips of the day to an object

Note: Use this task to add tips to an application, form, or application version that does not already have tips associated with it. To add tips to an existing tip set, double-click one of the existing tips on the Work With Tip of the Day form to access the Tips of the Day Revisions form.

1. On Work With Tip of the Day, click Add.
2. On Tips of the Day Revisions, complete the following fields:
 - Application Name
 - Form Name
 - Version
 - Description
 - Force tip to all users
3. In the detail area, add a data dictionary item for each tip you want to associate with the application. Complete the following columns for each row:
 - Tip Sequence
 - Data Item

Note: You cannot add a data dictionary item to the tip set if it does not have glossary text.

4. When finished adding data dictionary items to the application, click OK.

Field	Explanation
Application Name	Tip of the day application.
Form Name	The unique name assigned to a form.

Field	Explanation
Version	A user-defined set of specifications that control how applications and reports run. You use versions to group and save a set of user-defined processing option values and data selection and sequencing options. Interactive versions are associated with applications (usually as a menu selection). Batch versions are associated with batch jobs or reports. To run a batch process, you must choose a version.
Description	Describes the tip of the day application.
Force tip to all users	Allows the tips for an application to be forced on.
Tip Sequence	The tip of the day sequence.
Data Item	<p>For World, the RPG data name. This data field has been set up as a 10-byte field for future use. Currently, it is restricted to 4 bytes so that, when preceded by a 2-byte table prefix, the RPG data name will not exceed 6 bytes.</p> <p>Within the Data Dictionary, all data items are referenced by this 4-byte data name. As they are used in database tables, a 2-character prefix is added to create unique data names in each table specification (DDS). If you are adding an error message, this field must be left blank. The system assigns the error message number using next numbers. The name appears on a successful add. You should assign error message numbers greater than 5000. Special characters are not allowed as part of the data item name, with the exception of #, @, \$.</p> <p>You can create protected data names by using \$xxx and @xxx, where you define xxx.</p> <p>For OneWorld, a code that identifies and defines a unit of information. It is an 8-character, alphabetical code that does not allow blanks or special characters such as: % & , . +.</p> <p>Create new data items using system codes 55-59.</p> <p>The alias cannot be changed.</p>

Messaging



Messaging

Use J.D. Edwards messaging facilities to get pertinent information to the end user in the most effective and user-friendly manner. When you design an application to use messaging, you must evaluate what information is necessary for a user to accomplish a task. You can deliver a message in real time, where the message is displayed on the status bar. The method you use to provide information to the user depends on the situation:

- Use an interactive error message if there is an error during the entry of a record.
- Use an informational message sent to the Workflow Center if information needs to be conveyed and action requested.
- Use an alert message if information is urgent and needs immediate attention.
- Use a batch error message if errors are detected while a report is running.

There are three parts to creating system-generated messages:

- The message itself.

Do you require a simple message or a text substitution message? Are all the text substitution pieces available?

- The logic.

Has certain criteria been met so that a message should be sent? This will usually be event rule logic.

- Is this an action message?

Does the message require action by the users? Are all the required parameters available at the time the message is to be sent?

Message Types

There are two main types of messages:

- Error and warning messages
- Information messages



Information messages can also be action messages that enable the user to connect from a current form to another form that will allow them to correct an error, or to evaluate information and then take action. To use action messaging you must be sure the parameters for the connecting form are available at runtime.

Within these message types you can use simple messages or text substitution messages. Text substitution messages allow you to use variable text substitution. Substitution values are inserted into the message for the appropriate variable at runtime. This gives the user a customized message unique to every instance of the message.

There are two types of text substitution messages:

- Error messages (glossary group E)
- Workflow messages (glossary group Y)

They are both created in the same manner.

Error Messages

Error messages are stored in the data dictionary. The data dictionary design tools can be found on menu GH951.

Workflow Messages

When designing an application to use messaging, you must evaluate what information needs to be used to decide whether a message needs to be created.

For example, if a voucher is created for \$20,000 for capitalized equipment and the accounting department decides that the controller must be notified of any vouchers entered for greater than \$10,000, the information needed to create a message from voucher entry would be “who is the controller,” and “what are the keys to voucher entry.” At this point, you need to determine what type of J.D. Edwards message would be appropriate and call that type of message within Event Rules. See *Enterprise Workflow Management* for more information about creating complete workflow processes using workflow messages.

Level Messages

Level messages are used to categorize error messages into the proper level break within a report. They are like a container for messages. Level messages can be thought of as titles, such as “Here are your batch errors” or “Here are the document errors,” and so on. Level messages are used to separate every logical grouping of error messages. Messages that begin with “LM” are level messages. Level messages that belong to glossary group “Y,” indicate that they are workflow messages.

Information Messages

Messages that are not level messages (“LM”) but are in glossary group “Y” are considered informational messages (these may also include action messages). These messages supply pertinent information to the user and usually require action be taken.

This section describes the following:

- Understanding error handling
- Working with messages
- Working with the Send Message system function

Understanding Error Handling

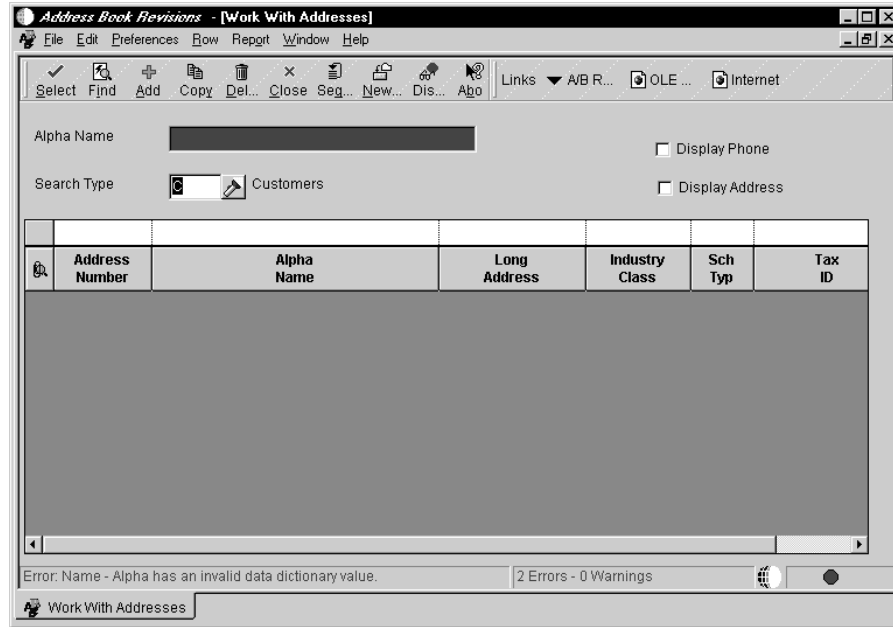
To understand error handling you should become familiar with:

- The event-driven model
- Classifications of error settings
- Resetting errors
- Multilevel error messaging

Event-Driven Model

Interactive messages display whenever a specific event occurs. This means that errors clear, set and display based on certain events. When an error occurs on an event, the event then runs again to clear the error.

For example, suppose the user types an incorrect value in a field. An error is set when the *Control is Exited* event is run (if you have logic on this event or a high level trigger is causing validation of the field). The user receives notification of the error through a visual cue (the field in error turns red). In addition, the status bar displays information about the error. To find out details about why the error occurred and how to correct it, the user can press the F8 function key. You can use the F7 function key to walk through the errors on a form if there are two or more errors.



The only way for the user to correct the error is to go into the field again, type in the correct value and leave the field. This causes the same *Control is Exited* event to be run again, which clears the error and sets it again in case an incorrect value is entered again. If the value typed in this time is correct, no error is set.

When is an Error Set?

When a user enters data in a form field or grid cell and tabs out, the runtime engine edits the value that was entered. If the edit fails, (such as a value is not found in the User Defined Code table, or an invalid date has been entered), an error is issued with the appropriate error code and the field is highlighted. The error message and message count are displayed on the status bar.

How Does the Application Know Which Field to Highlight?

When an error is issued, the handle is on the field. If the error is within a grid, then the offending grid cell coordinates (row and column) are identified. OneWorld stores this information in an *Error_Ctrl_Key* structure, along with the application form handle. At the time the error is displayed, the color of the control is changed and the error information is displayed from the error link list. If you use the system function *Set Control Error* or *Set Grid Cell Error*, you can specify the field or cell to highlight.

How Is an Event ID Used for Error Setting?

OneWorld applications are event-driven. Each keystroke or mouse click is considered an event. When a user enters a value in a control and tabs out of the field, the *Leave Control* event or *Leave Column* event validates that field. All

errors are set using the event ID in the `Error_Event_Key` structure. This event ID is used to clear the errors that were set on this event when the event is reprocessed. For example, when the user enters an invalid address book number and tabs out of the field, an error is displayed. This error is set with the Control Handle, Event ID and Grid Cell Coordinate (if any). To clear this error, the user must enter a valid address book number and tab out again. Validation is done and the error is cleared if no more errors are found on this control.

Error Setting

Errors are set:

- Automatically
- Manually

Automatic Error Setting

The runtime engine validates a field in the following instances:

- The item is a data dictionary item, for which an edit rule trigger has been defined in the data dictionary. In this case, the field for the data item is validated across all OneWorld applications. Data dictionary triggers can also be assigned or redefined using overrides within the Forms Design tool. However, in this case the trigger for the data item is specific to the application where the override was defined.
- Validation is performed when the *Control is Exited* event, *Column is Exited* event, or OK Button Processing occurs. You can use validation to check for things such as invalid Data or invalid Dates.

Manual Error Setting

You can use either a system function or a business function API to set other error messages.

System Function

You can use the following system functions for setting an error message:

- Set Edit Control Error
- Set Grid Cell Error

You can use Set Edit Control Error to set an error on a Form Control field through event rules. The following parameters must be passed in as system function arguments:

- Control (for example, the field that is in error)

- Error Code (a literal in this case means the error message, created through 3/GH951 or a variable)

You can use Set Grid Cell Error to set an error on a Grid Control field through event rules. You pass in the following parameters as system function arguments:

- Grid (does not receive any value)
- Row Number
- Column Number
- Error Code

System functions cannot use data items with text substitution.

Business Function API

You can set an error message using a business function. This is used for manual error setting.

Use one of the following business function APIs to set an error message without text substitution:

- `jdeErrorSet (lpBhvrCom, lpErrInfo, idItem, lpzError)` sets errors through business functions.
- `jdeSetGBRError (lpBhvrCom, lpVoid, (ID)0, "Error#")`

Following is an example of an error message without text substitution:

```
{
    jdeSetGBRError(lpBhvrCom,lpVoid,IDERRCRetainedEarningsLedger_3,"4524");
    idReturn = ER_ERROR;
}
```

The return code successful message has no influence on what happens next. It is used for information.

Use one of the following business function APIs to set an error message with text substitution:

- `jdeErrorSet (lpBhvrCom, lpvoid, idItem, lpzError, lpDs)` sets errors through business functions using substitution text. It always gets called
- `jdeSetGBRErrorSubText`

`JdeSetGBRError` and `jdeSetGBRErrorSubText` work the same except that `jdeSetGBRErrorSubText` has an additional parameter in which you pass the address of the data structure that holds the information you want substituted in.

Following is an example of the setup for a text substitution error message:

```

/*****
*****
* Declare structures
*****
*****/
DSDE0018    dsDE0018;
.
.
/*****
*****
* Main Processing
*****
*****/
    strncpy(dsDE0018.szAccountID, (const char
*) (lpDS->szAccountID),
        sizeof(dsDE0018.szAccountID));
.
.
.
if (idReturn != ER_SUCCESS)
{
    jdeSetGBRErrorSubText(lpBhvrCom, lpVoid,
IDERRszAccountID_1, "044E", &dsDE0018);
JDB_CloseTable(hRequestF0901);
JDB_FreeBhvr(hUser);
return ER_ERROR;
}

```

In this example, DSDE0018 is the data structure for the error messages and is declared in jdeapper.h. Jdeapper.h is where all of the data structures are declared for text substituted error messages used by J.D. Edwards. Non-J.D. Edwards substituted error messages should be defined in their own global declaration header file or in the function's header file that is using the structure. The line IDERRszAccountID_1 in this example contains the values to substitute.

A business function may call a validation routine (jdeddValidation), which when called may set an error if the field in question is invalid.

To use text substitution error messages in C business functions, you create an instance of a data structure in the global header file for every custom business function. If the error data structure exists, it should exist in jdeapperr.h

Resetting Errors

The system will reset errors on the OK button or on the Find button. This occurs on both manual and automatic errors.

Resetting Errors on the OK Button

The system resets errors when the OK button is clicked on a Header Detail form or a Headerless Detail form. The following actions occur:

1. Setup `Error_Event_Key` structure with event *Button Clicked*, the Control Handle is the OK button
2. Clear event errors on OK *Button Clicked*
3. Clear error due to the following events:
 - *Button Clicked Processing Done*
 - *Add Record to DB - Before*
 - *Add Record to DB - After*
 - *Update Record to DB - Before*
 - *Update Record to DB - After*
 - *All Grid Recs Added to DB*
 - *All Grid Recs Updated to DB*
 - *Delete Record from DB - Before*
 - *Delete Record from DB - After*
 - *All Grid Recs Deleted from DB*
 - *Delete Grid Rec Verify - Before*
 - *Delete Grid Rec Verify - After*
 - *Row is Exited*
4. Stop processing if error still exists
5. Perform event rule on *Button Clicked*
6. `ValidateAllData` - this includes form controls
7. Stop processing if error occurred
8. Delete any rows that were removed from the grid
9. Perform event rule for Add or Update to Database Before or After
10. Perform grid changes (validate any unprocessed grid row)
11. Stop processing if error occurred
12. Perform event rule on *After Button Clicked*
13. Stop processing if error occurred
14. Otherwise, clear screen and exit

Do not try to validate and set errors on any grid control field for the *Button Clicked* event on the OK button. The *Button Clicked* event will not process all the grid rows.

Resetting Errors on the Find Button

The system resets errors on the Find button. It performs the following steps:

1. Setup `Error_Event_Key` structure with event *Button Clicked*

2. Clear event errors on OK/Select *Button Clicked*
3. Clear errors due to the following events:
 - *Button Clicked*
 - *Last grid record has been read.*
 - *Form Record is Fetched*
 - *Grid Record is Fetched*
 - *Confirm Delete - Before*
 - *Confirm Delete - After*
4. Perform event rule on *Button Click*

Multilevel Error Messaging for C Business Functions

When a business function calls another business function and the error is issued from the second business function, you must provide a mapping key array. Otherwise, errors will be set incorrectly.

When you use `jdeCallObject` (the standard API for calling other business functions from within a business function) for the second business function call, the sixth parameter is for error mapping. Each business function has its own header and definition. You need to know why you are calling the second level business function. For example, suppose you are calling a second level business function to validate the company number. You must have the company number ID in the first business function header so you can find out the company number ID in the second business function.

► To create multi-level error messaging

1. Open the called business function's header file and determine the maximum number of possible mapping fields.

Calling Function's Header File	Called Function's Header File
<code>#define IDERRszComputerid_1 1L</code>	<code>#define IDERRcHeader_1 1L</code>
<code>#define IDERRszCompany_2 2L</code>	<code>#define IDERRcEvent_2 2L</code>
<code>#define IDERRszDate_3 3L</code>	<code>#define IDERRDetail_3 3L</code>
<code>#define IDERRszValue_4 4L</code>	<code>#define IDERRCompany_4 4L</code>

```
#define IDEERszPoint_5 5L
```

```
#define IDERRDateOpen_5 5L
```

2. Create an Error map section in the calling business function's C file.

In the following example of an active error message, you need to map one field (IDERRCompany_4).

```

/*****
* Business Function: B1234
*
*
* Parameters:
*     LPBHVRCOM
*     LPVOID
*     LPDS1234
*****/

/*****
* Error Mapping Section
*
* Map to function "ValidateCompanyName"
* #define N1234 1
*****/

```

The number “1234” is the data structure number of the called business function. The number “1” is the number of mapped fields.

Before each #define statement there should be a comment that refers to the business function name that this number will be used for.

In the variable section create a “cm_xxx” map array for each function that needs to return errors.

For example:

```

/*****
* Variable declarations
*****/

CALLMAP    cm_1234[N1234] = { { IDERRCompany_2, IDERRCompany_4 } };

```

The array is mapped from IDERRCompany_2 to IDERRCompany_4

The function call to jdeCallObject is:

```
idReturn = jdeCallObjdect("ValidateCompanyNumber",
ValidateCompanyNumber, lpBhvrCom,lpVoid, &ds1234,
cm_1234, N1234, (char *)NULL, (char *)NULL, (int)0);
```

The business function sets the error on the ID field.

Working with Error Messages

OneWorld displays messages automatically based on certain events. For example, you can display an error message whenever an invalid value has been entered into a field.

OneWorld uses the data dictionary glossary to display messages. The message is contained within the glossary portion of a data item. By leveraging the existing framework of the data dictionary, you do not have to create a messaging system from scratch.

This chapter describes the following:

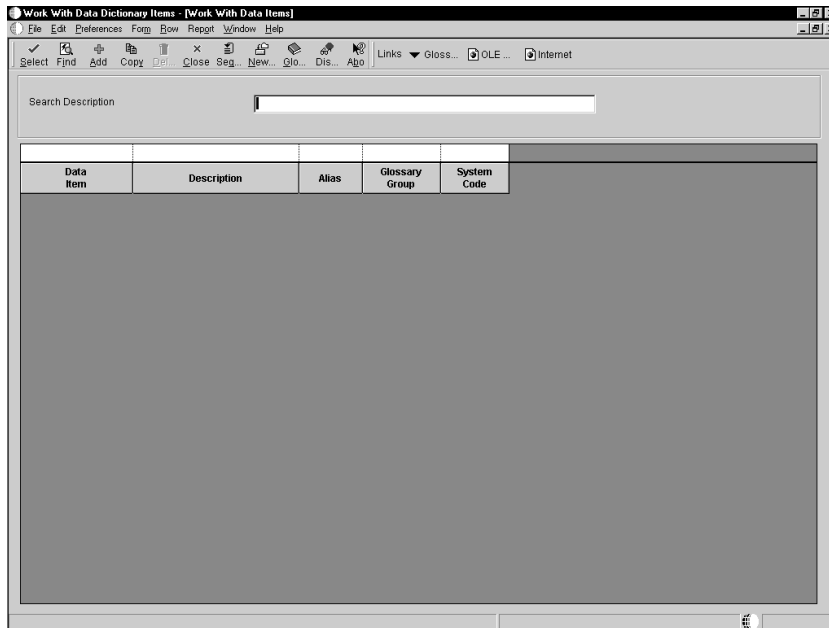
- Locating an existing error message
- Creating a simple error message
- Creating a text substitution error message
- Attaching an interactive message data item

Locating an Existing Error Message

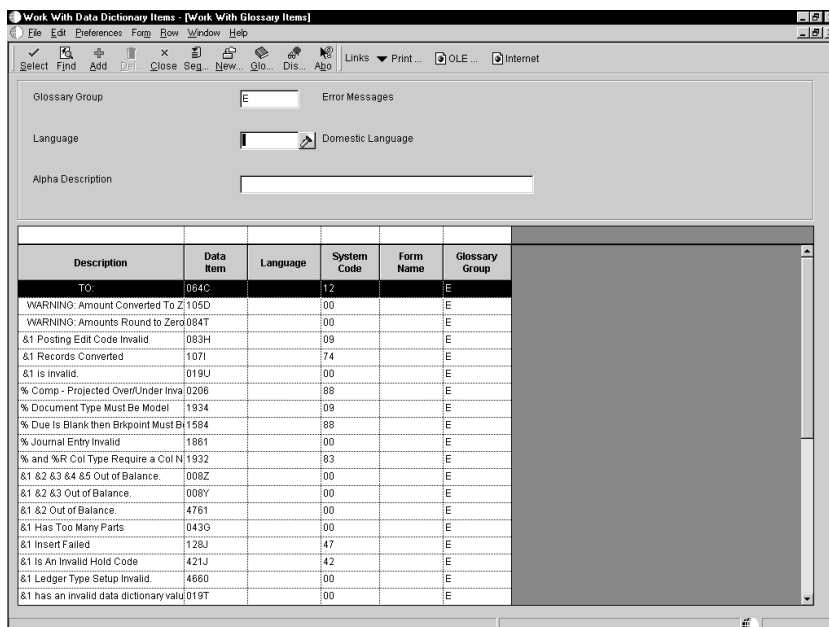
You should use an existing message if possible instead of creating a new one. Use Work with Glossary Items to locate an existing message. You can narrow your search by using glossary group types and descriptions similar to the error you want.

To locate an existing error message

1. On Work with Data Dictionary Items, choose Glossary Data Item from the Form menu.



2. On Work with Glossary Items choose the message you wish to use.



Creating a Simple Error Message

Simple error messages (static messages) contain literal text, for example, “Enter a valid date.” Simple error messages do not use text substitution.

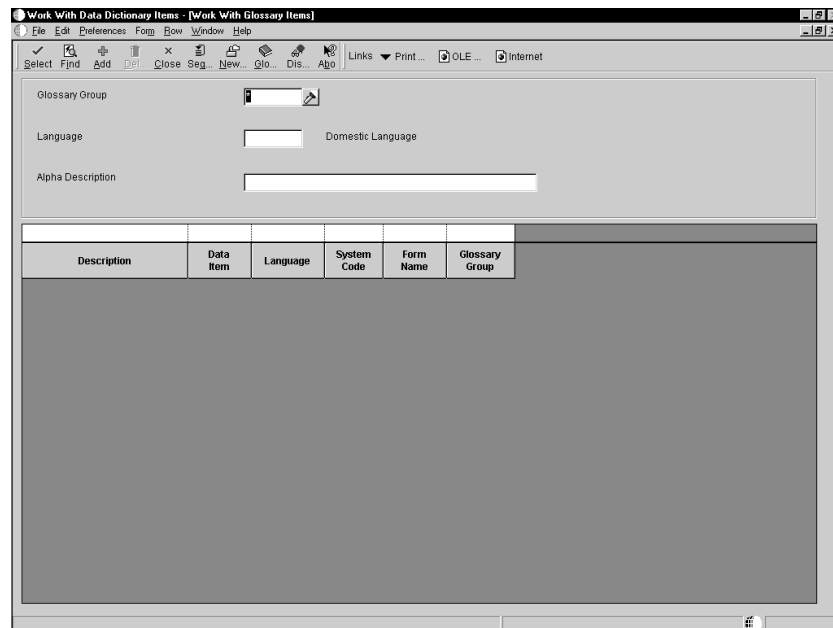
Adding an Interactive Message Data Item

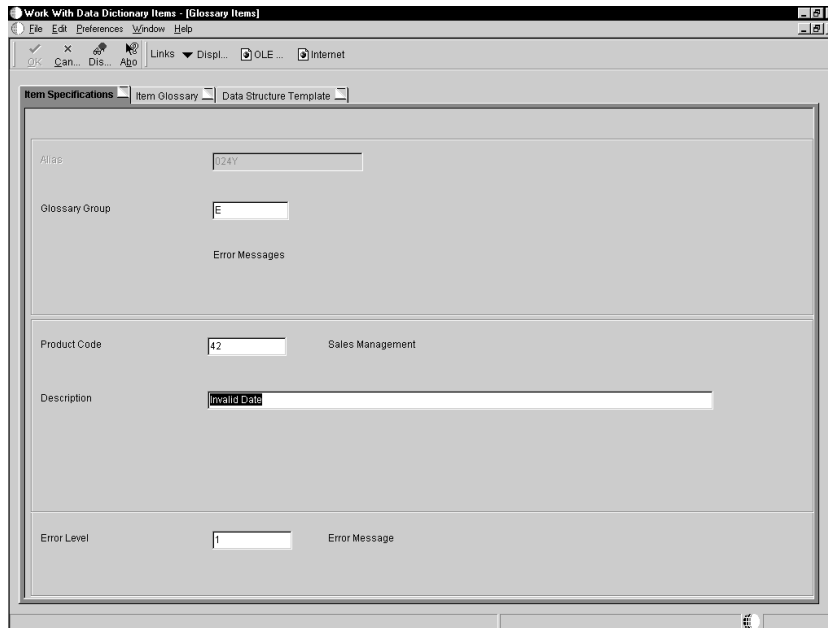
You can create a new data item or select an existing item to display a message.

Before the error message has functionality, you must connect it to a form control using event rule logic.

► To add an interactive message data item

1. On Work with Glossary Items, click Add.





2. On Glossary Items, click the Item Specifications tab, and then complete the following fields:
 - Glossary Group
 - Product Code
 - Description
 - Error Level

Creating a Text Substitution Error Message

A text substitution error message is an error message that allows variable text to be substituted directly into the glossary text for the error message. This allows you to display the same basic message with some values that vary.

For example, suppose a user enters an invalid search type in a field. The error message “Search type *xx* is not contained in the 00 ST table” appears. The *xx* in the message will be replaced by the search type the user entered.

The following tasks are required to create a text substitution error message:

- Adding an interactive message data item
- Defining the glossary text for a runtime message
- Attaching a data structure template to a message

If you want to use text substitution error messages in batch processes, you must create a business function to pass values to the data dictionary data structure.

The event rule engine (set user error) will not pass values to the text substituted error message.

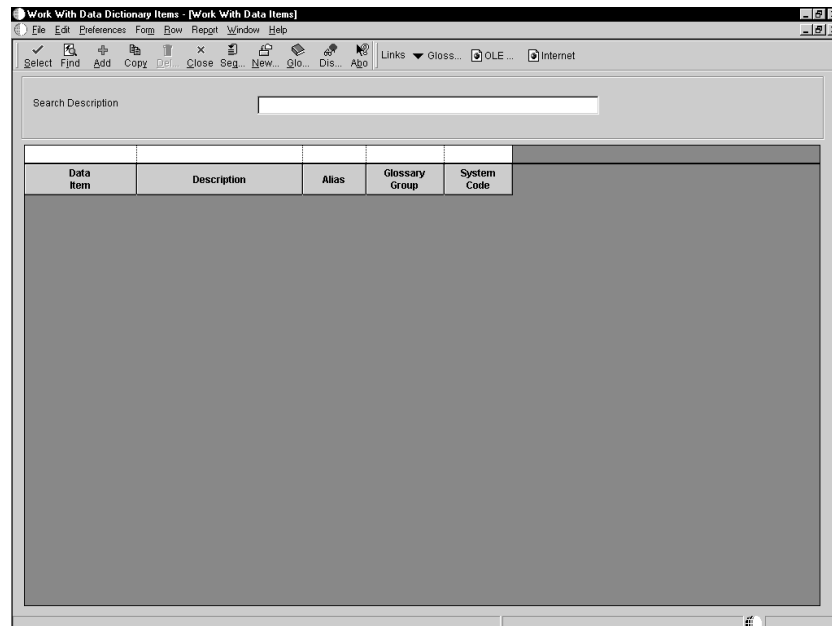
Adding an Interactive Message Data Item

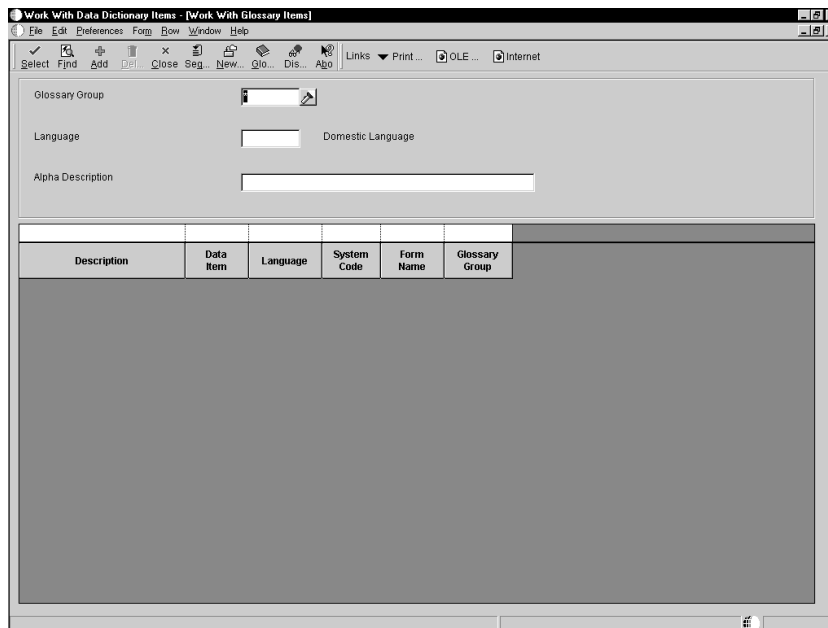
You can create a new data item or select an existing item to display a message. This example creates an error message in the Data Dictionary for error ID 018A.

Before the error message has functionality, you must connect it using event rule logic.

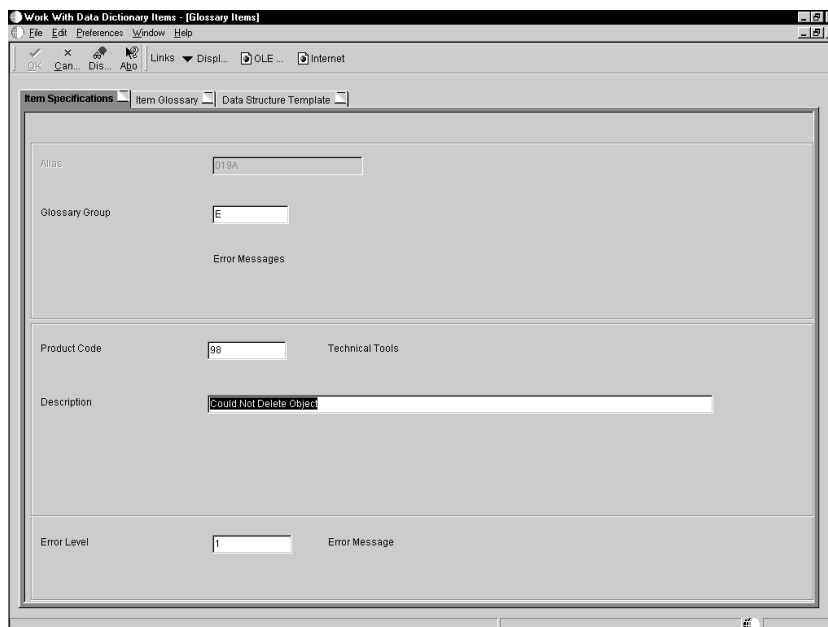
► To add an interactive message data item

1. On Work with Data Dictionary Items, choose Glossary Data Item from the Form menu.





2. On Work with Glossary Items, click Add.



3. On Glossary Items, click the Item Specifications tab, and then complete the following fields:
 - Glossary Group
 - Product Code
 - Description
 - Error Level

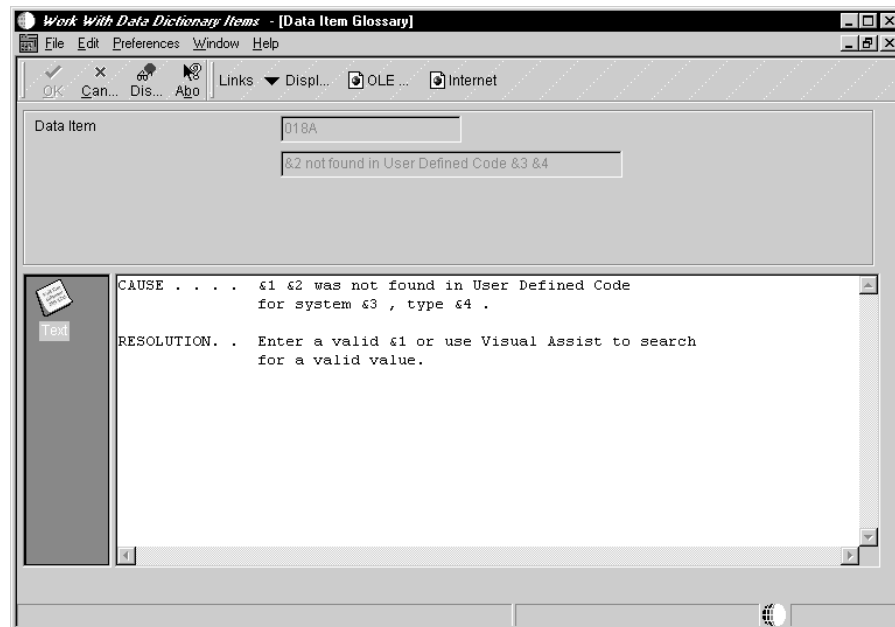
When you type the description, use & followed by a number for each subset variable. In this example, the short description is *&2 not found in User Defined Code &3 &4*.

Now you are ready to define the glossary text to display at runtime.

Defining the Glossary Text for a Runtime Message

► To define the glossary text for a runtime message

1. On Work with Glossary Items, click Glossary.



2. Type the text of your message. Use “&x” to assign variables in the text. These variables correspond to members in a data structure. Put a period with no blank spaces at the end of your glossary so you do not have any repetition problems.

The order of the data structure is the index of the &x. If the structure contains four members, assign &1 to the first member, &2 to the second, and so on.

Description = &1

UDC Value = &2

System = &3

Type = &4

3. When your text is complete, click OK.

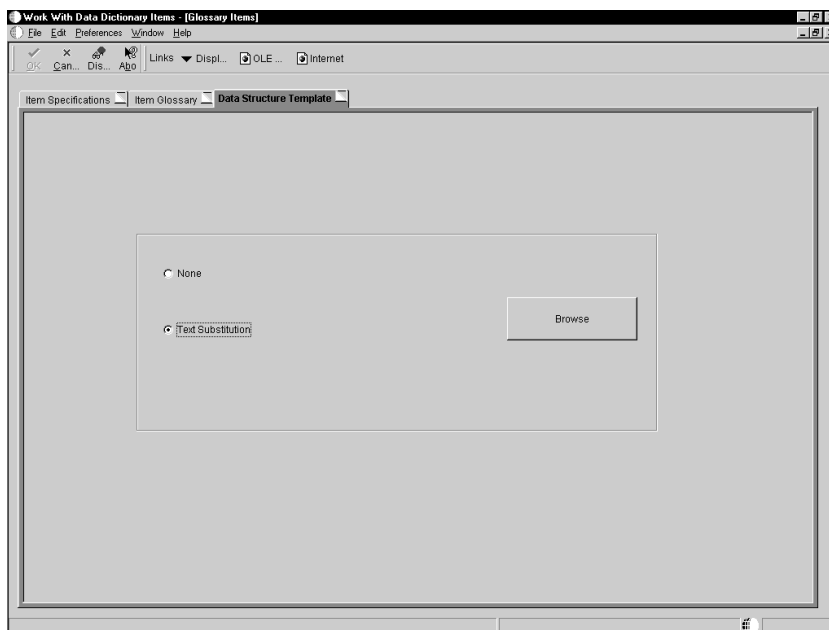
OneWorld returns to the Glossary Header Revisions form.

Attaching a Data Structure Template to a Message

Each text substitution error message requires a corresponding data structure. If the data structure does not already exist, then you must create it. Make sure the members in your message match the members in your data structure. See *Data Structures*.

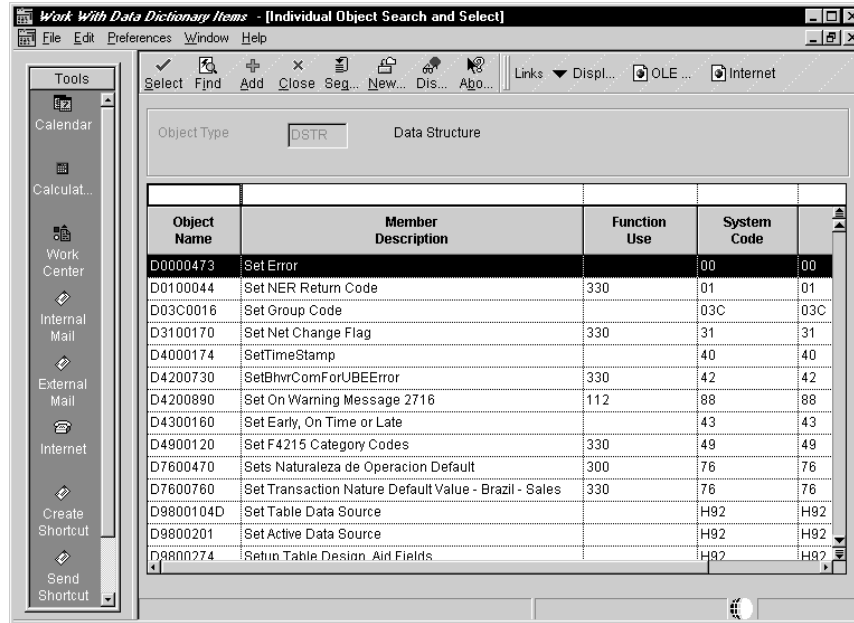
► To attach a data structure template to the message

1. On Glossary Items, click the Data Structure Template tab.



You can browse to locate an existing structure. However, if there is not an existing structure that meets your needs for the interactive message, you must create one.

2. To locate an existing structure click the Browse button.



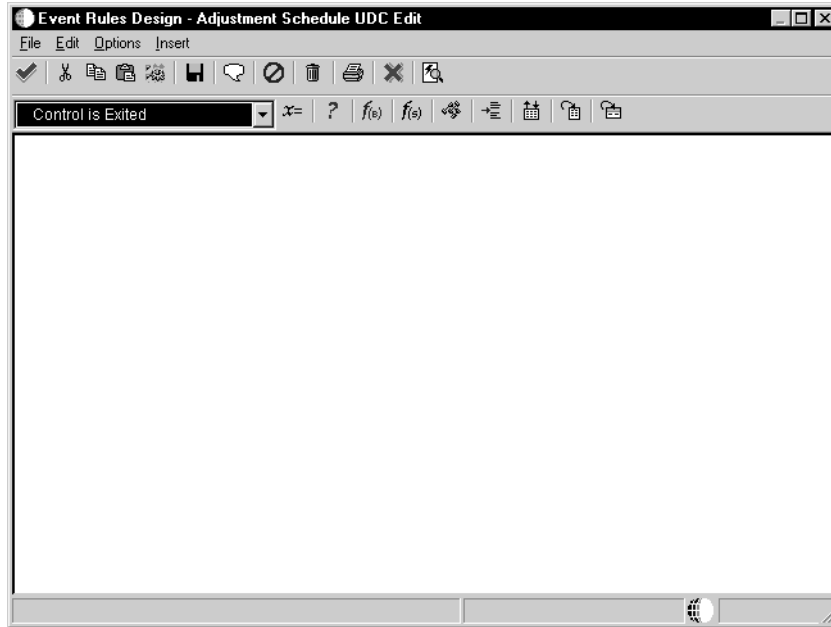
3. On Object Search, locate the data structure and click OK.

Attaching an Interactive Message Data Item

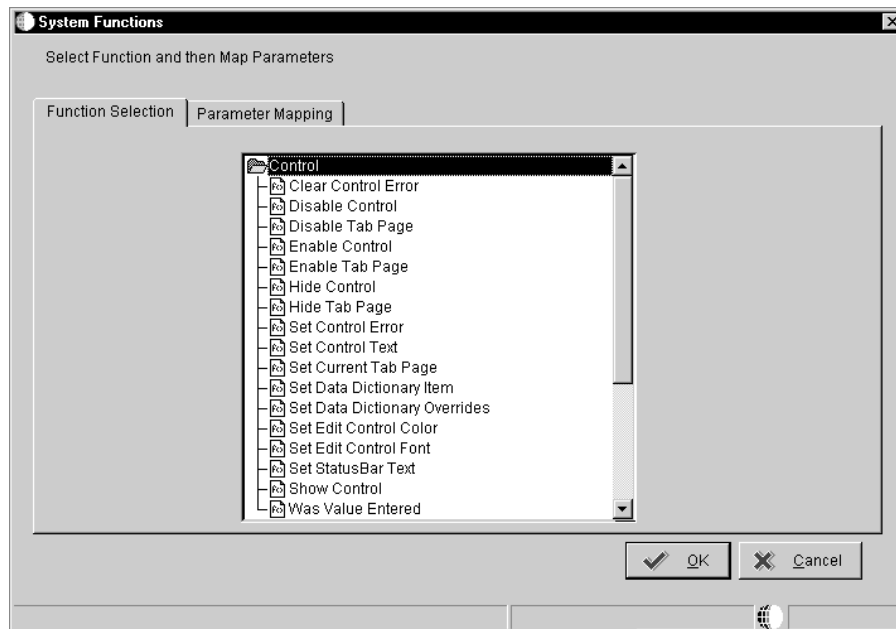
To make the error message display at runtime, you must attach a system function.

► To attach a message data item

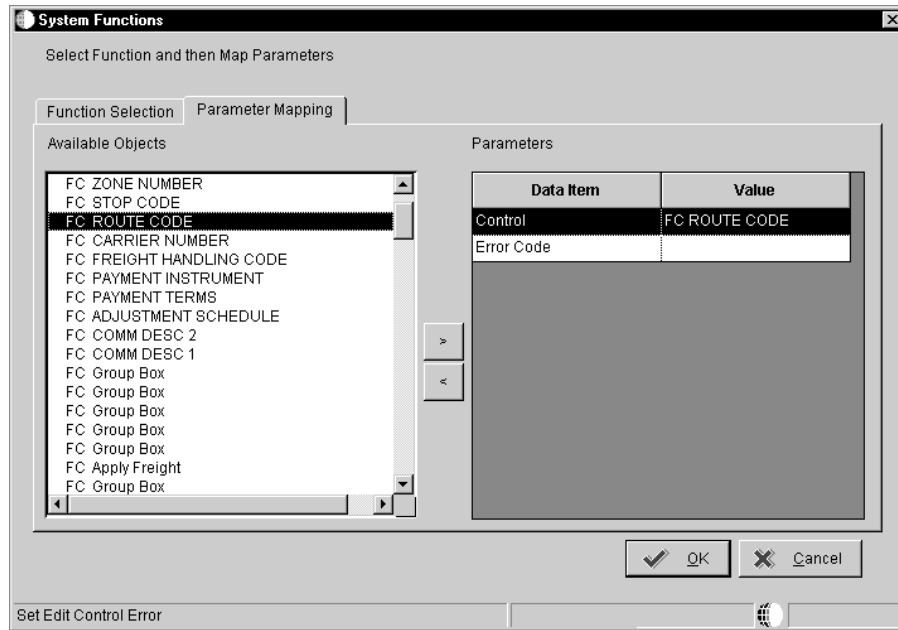
1. On Event Rules Design, click the system function button.



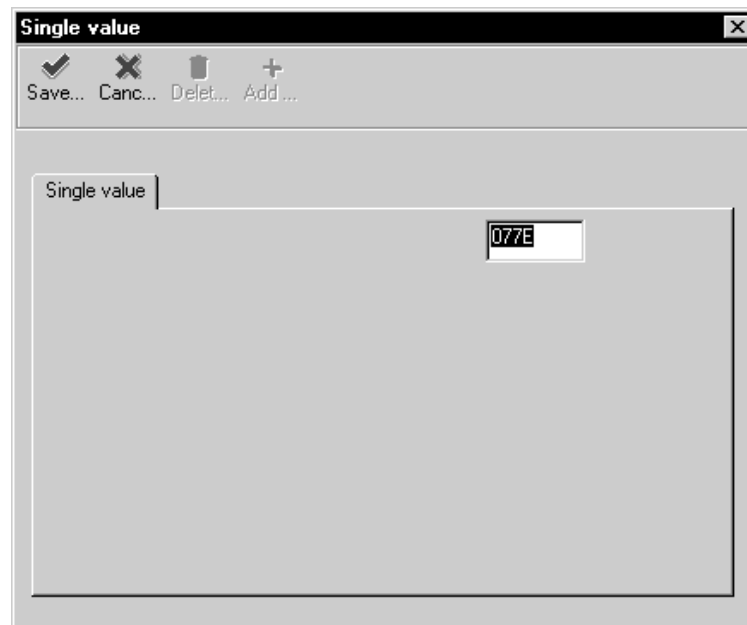
2. Select the system function you wish to use.



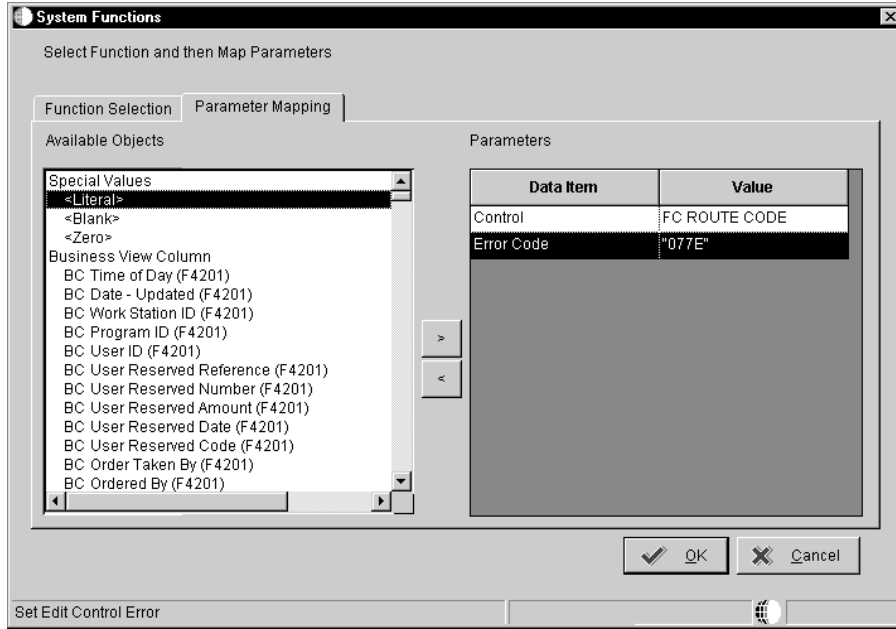
3. Use available objects to complete the control value in Parameters.



4. Choose <Literal> for the Error Code Value in Parameters.



Use the number of the data dictionary item that contains the error message you want to appear for the literal value.



Working with the Send Message System Function

The Send Message API involves the following tables:

- F01131 - Message Header table
- F01131M - Multi-level Message
- F01133 - Detail table
- F00165 - Media Object Storage

The Message Header table (F01131) and the Message Detail table (F01133) are used by the Message Center application for delivery of messages. They contain information on who is to receive the message, the current status of the message, who sent the message, what date the message was sent, which mailbox the message belongs with, and all other pertinent mail delivery information. The Media Object Storage table (F00165) contains the subject and body of the message.

The Send Message system function is comprised of three components:

- The addressing and destination
- The message, with or without text substitution
- The action message, the act of calling other programs from a message

The following table lists the parameters and their functions for the Send Message system function.

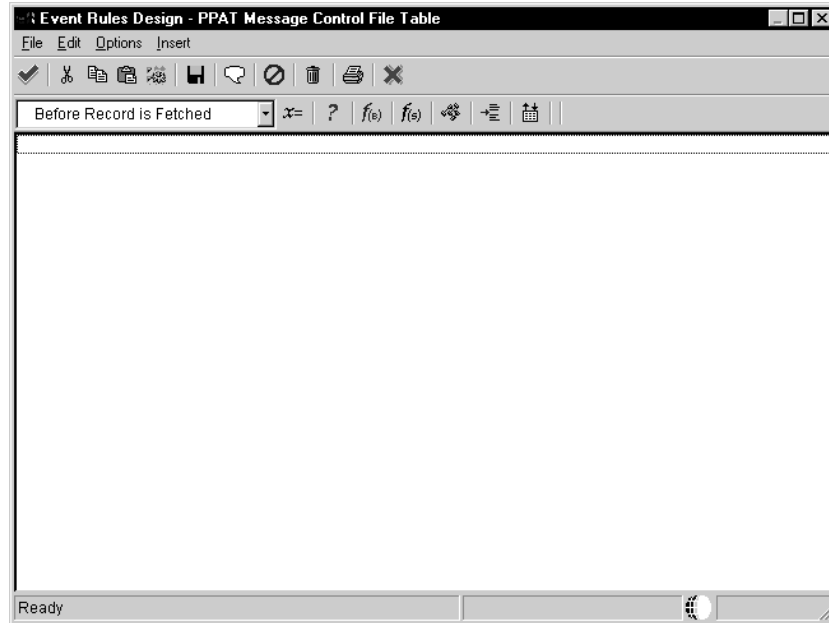
Parameter	Description	Typical Values	Mandatory
Recipient	The address book number of the person who is to receive the message.	Address Book Number - Data Dictionary item AN8	YES
Mailbox	The Mailbox to which the message should be delivered, for example, "Personal in Basket"	The two-character field associated with a Mailbox, which is found in UDC 02/MB.	YES
*Subject	The subject of the message.	A brief description of the message, or Blank if a message template will be used.	NO "None" should be selected if a message template is used.
*Text	The main body of the message.	The message that will be sent, or Blank if a message template will be used.	NO "None" should be selected if a message template is used.
Active	This indicates whether there should be a connection between the message being sent and another form.	When this item is selected the user will be prompted through the form interconnect process.	NO
DDMessage	The Message Template from the data dictionary.	Any valid data dictionary item whose glossary can be used for a message.	NO
MessageKey	This is a returned parameter that holds the key value for the message sent.	Data dictionary item, for example SERK.	NO

If the Subject and Text fields are used in conjunction with the DDMessage field, then the DDMessage (MessageTemplate) will be concatenated onto the Subject and Text fields.

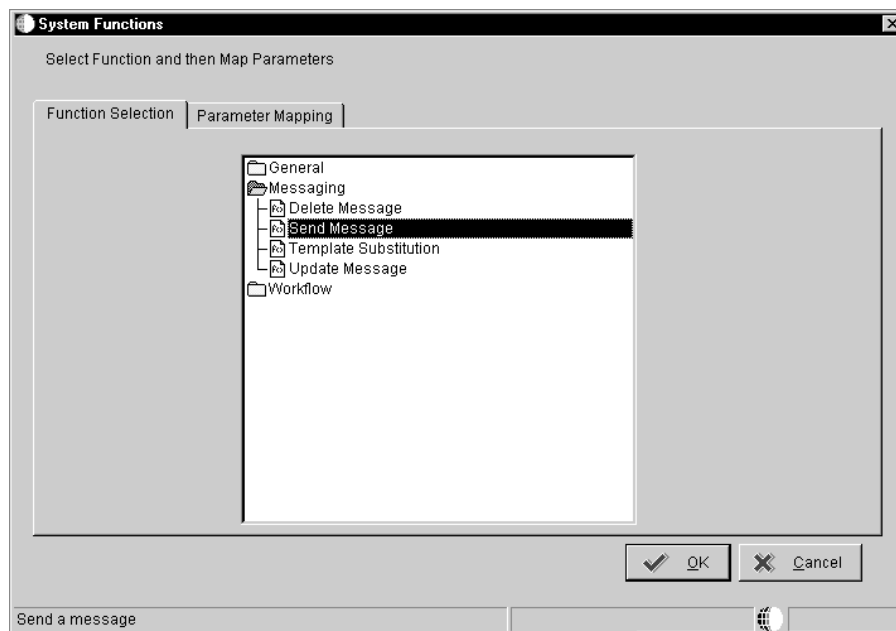
The following example illustrates how to use the send message system function.

► To use the send message system function

1. On Object Management Workbench, check out the table with which you want to work.
2. Ensure the table is highlighted and then click the Design button in the center column.
3. On Table Design, click the Design Tools tab and then click *Start Table Trigger Design Aid*.

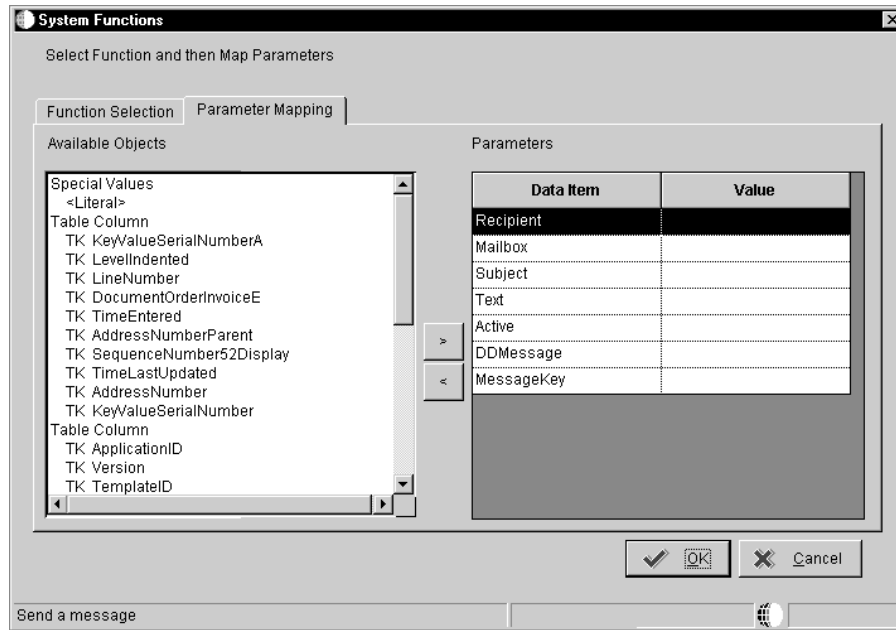


4. Click System Functions.



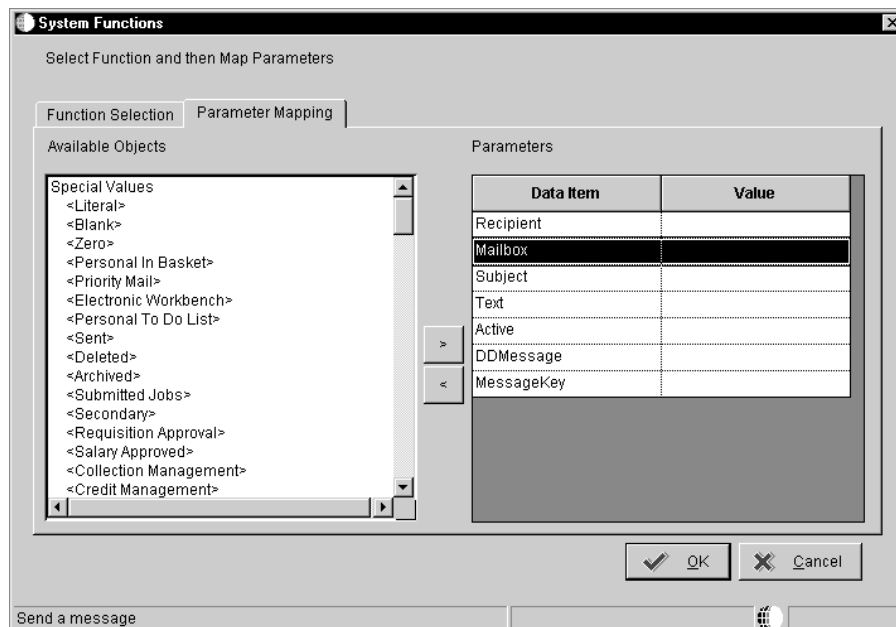
5. Designate who is to receive the message in the Recipient field.

This field typically uses an address book number. You can also use an e-mail address here.



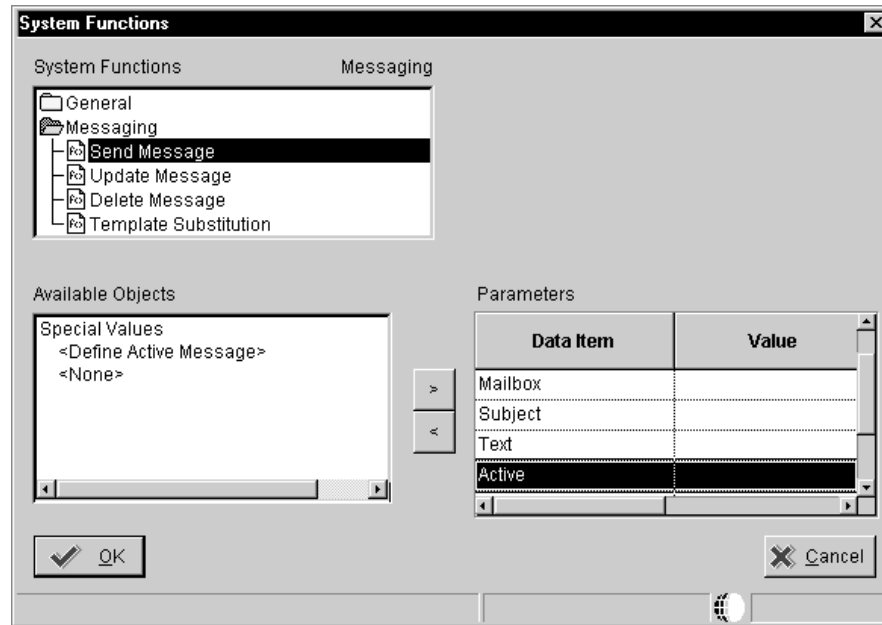
6. Next, designate the Mailbox.

The mailbox indicates what type of message is being sent. Mailboxes are stored in the UDC table 02/MB. Mailboxes from UDC table 02/MB will be displayed when focus is on the Mailbox field. The following form is an example of what might be displayed for mailboxes.



7. Click the data item called Active, and then choose Define Active Message to begin the form interconnection process.

The active message portion of the message works like form interconnections from event rules.

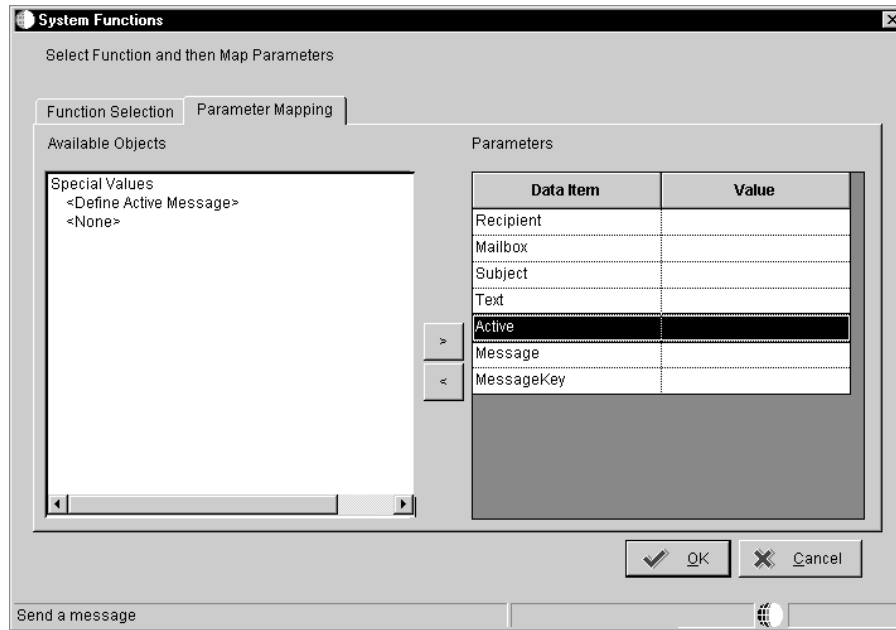


Defining an Active Message

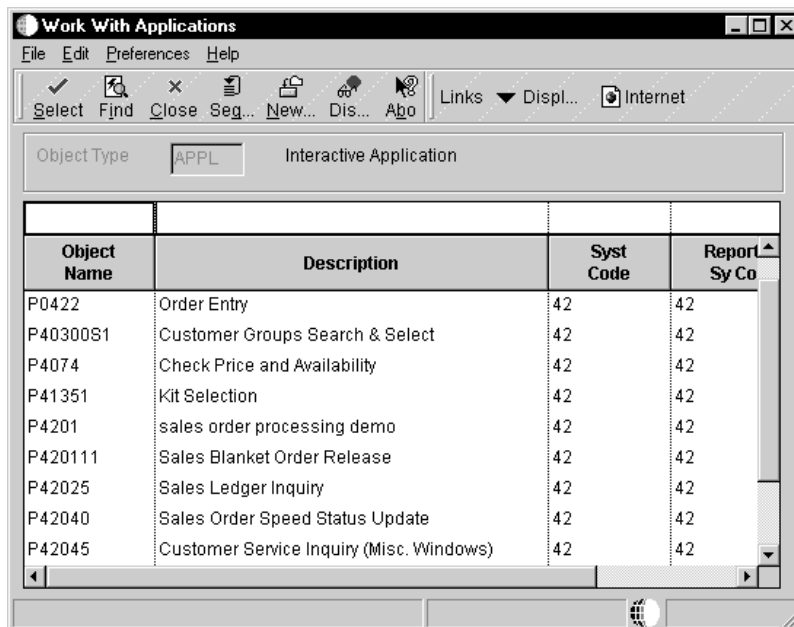
The following forms illustrate what forms appear as you define an active message.

► To define an active message

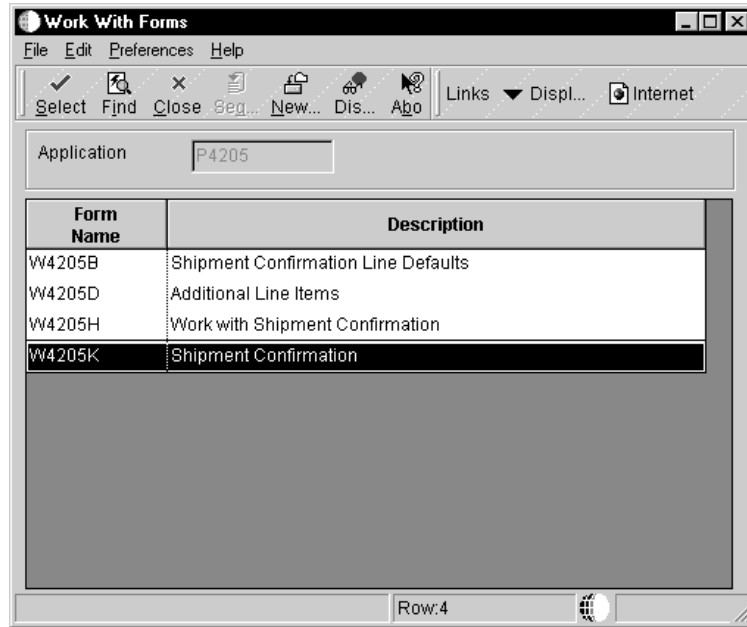
1. Double-click on Define Active Message.



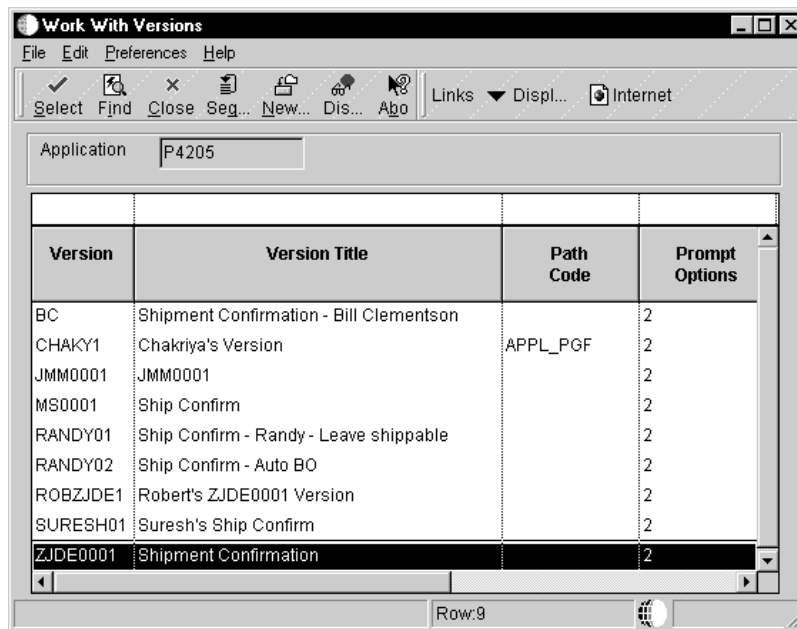
Work with Applications appears.



2. Choose the application you wish to work with.

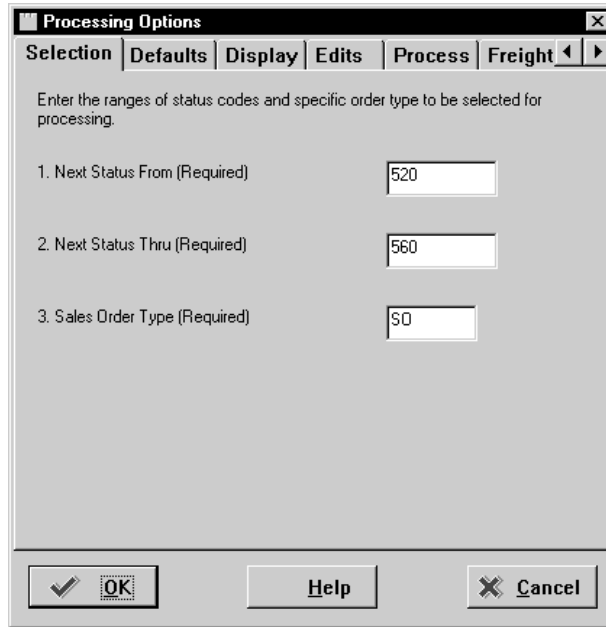


3. Choose the form you wish to work with.

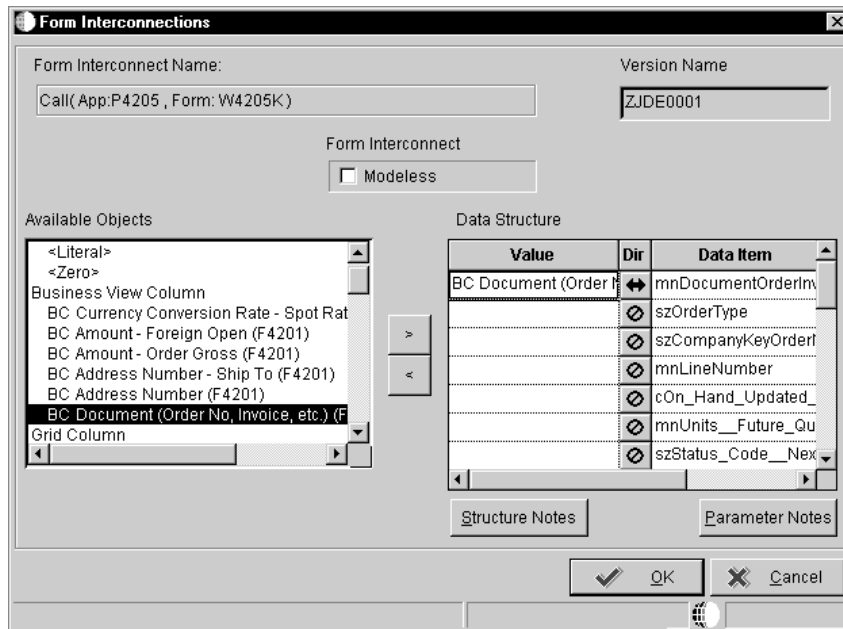


4. Choose the version you wish to work with.

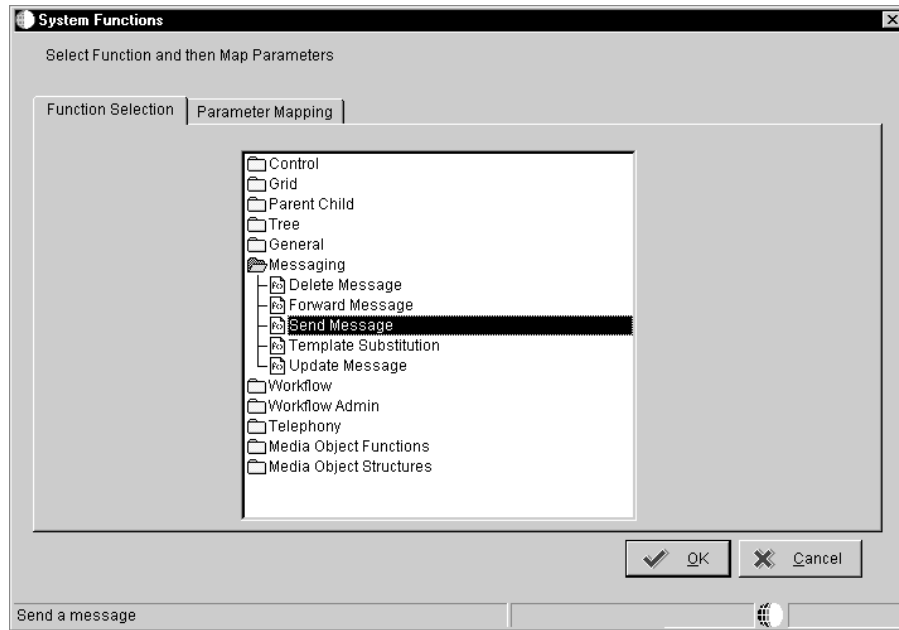
The Processing Options form may appear if there are processing options attached to the form.



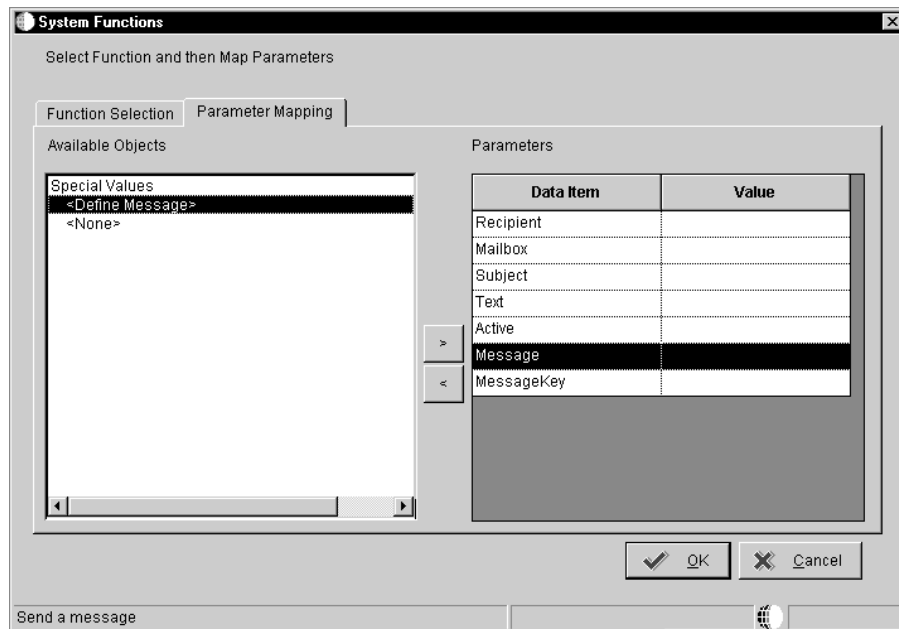
The Form Interconnections form displays, allowing assignment of all the values to be passed.



5. Designate the values you wish to pass.

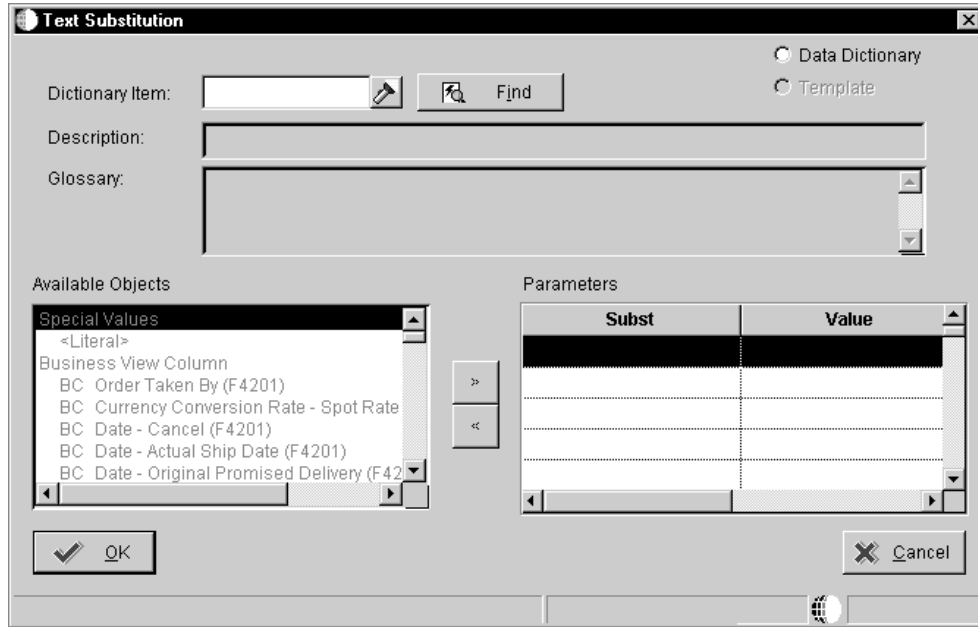


6. On System Functions select Define Message.

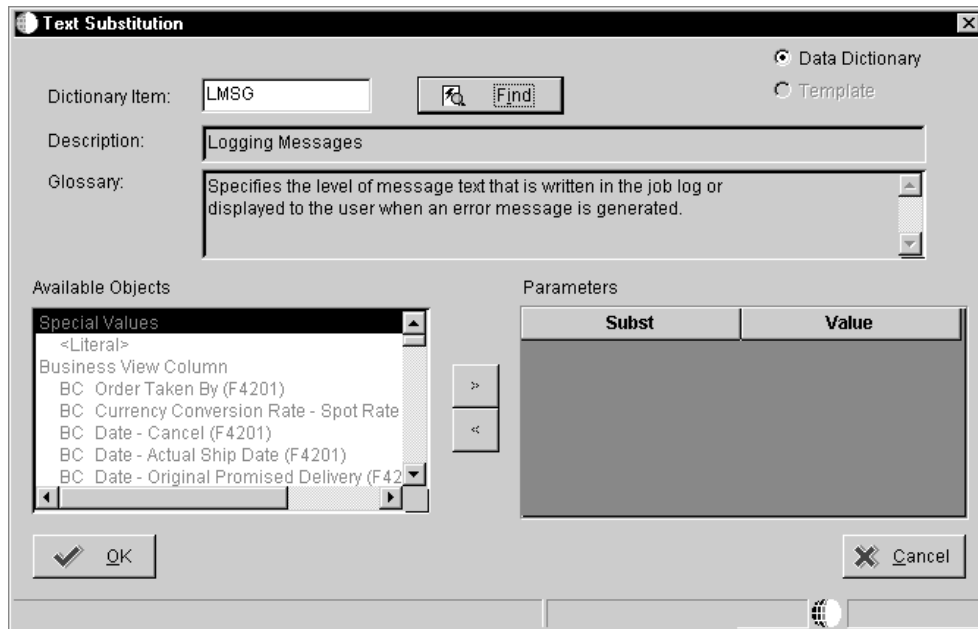


You will be prompted for the selection of the message.

7. Enter the appropriate alias in the Dictionary Item field and do a Find.



The substitution points are numbered &1, &2, etc. The parameter numbers correspond to the substitution numbers. Substitution variables can appear in both the Description and the Glossary of the message.



8. Map the appropriate objects to the substitution parameters so that the message will be complete.



Batch Error Messages

The error message system gives end users a consistent interface to review errors when working with OneWorld batch programs. When a batch program has finished processing all messages regarding the success or failure of the job, a message is sent to the end user using the Work Center. To make these messages useable, a tree (or parent/child) structure is used to group related messages. To provide additional flexibility and functionality, you can use text substitution, and you can make a message “active,” meaning that the user can open a form by clicking on the message.

This section describes the following:

- Understanding batch error messaging
- Creating a level-break message



Understanding Batch Error Messaging

The Work Center displays the error messages that appear after a batch job has completed. When you create these batch error messages, you need to determine what possible messages OneWorld users will need. You might create a number of different messages that are generated when a journal entry report is run. For example, you might create a single message stating that the report completed normally if the report balances. Or, you might create multiple levels of messages describing errors if the report is out of balance. The first level might state that the report completed with errors and additional additional levels of errors would then explain the specifics of the error.

Level-Break Messages

A level-break message is a message you can create that acts as a container for other error messages that are produced by the batch process. Level-break messages can be action messages, which contain a shortcut to an application and require action on the part of the user, or nonaction messages, which can be messages that instruct the user to review some information.

Understanding the role of a level-break message requires taking a closer look at how errors are created in batch jobs. Most errors are created by edits that occur at level breaks in the entire batch process. It is at these different level breaks that you want to group together any errors that may have occurred. The mechanism that groups these errors together is the level-break message. This implies that each reporting program that uses the Work Center application program interface (API) to manage errors will need to create one level-break message for each phase of level-break.

See Also

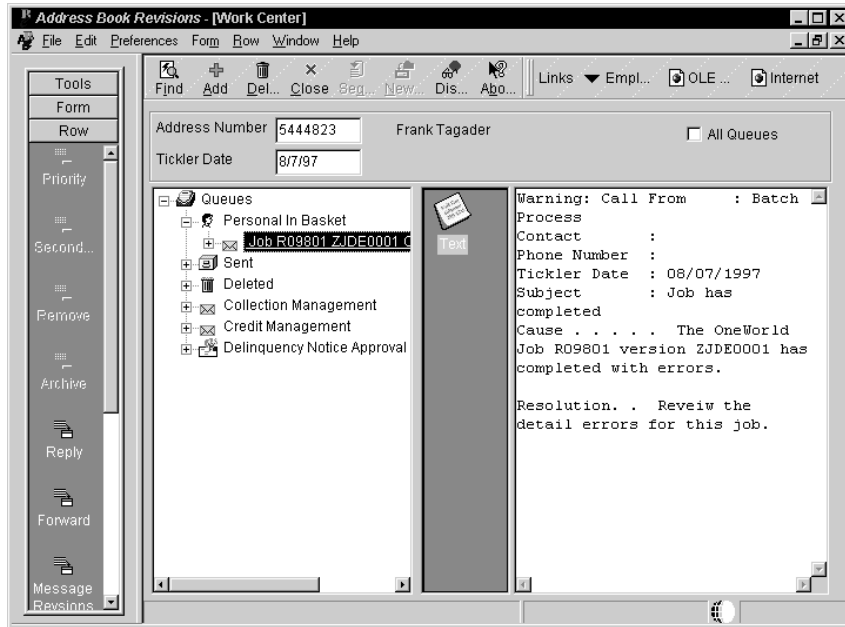
- *Understanding Level Breaks* in the *Enterprise Report Writing Guide* for more information about level breaks within sections.

The following examples show the how messages might appear when an out-of-balance journal entry upload has completed.

First-Level Messages

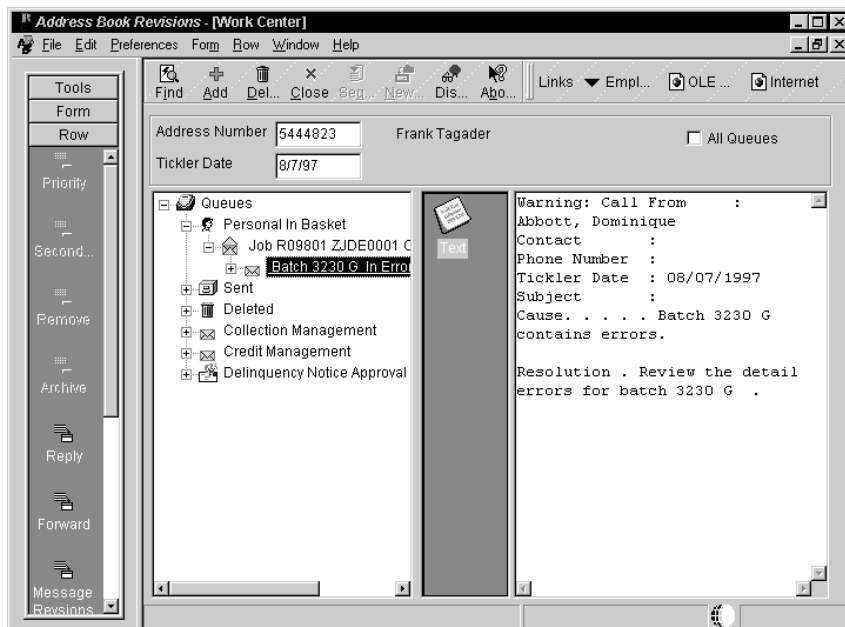
First-level messages appear when users open their personal in baskets. The plus symbol next to the message indicates that there are additional levels beneath it.

First-level messages might show the name of the batch job, explain that it completed with errors, and instruct the user to review the detail of the errors.



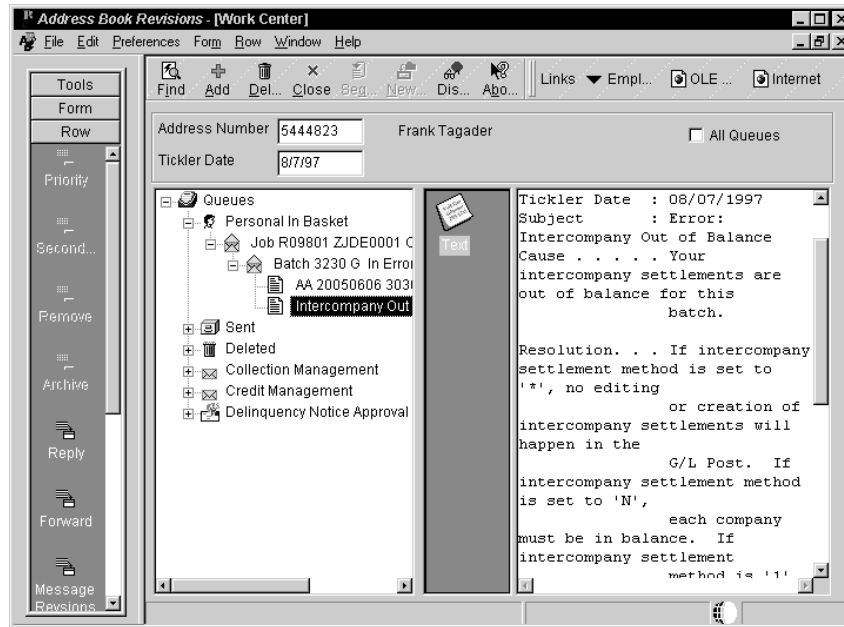
Second-Level Messages

Second-level messages appear when the user double-clicks the plus sign next to the first-level message. In this example, the second-level messages informs the user the batch number that needs to be reviewed.



Third-Level Messages

Third-level messages appear when the user double-clicks the plus sign next to the second-level message. In this example, the third-level message instructs the user that the batch job completed with errors because it was out of balance, and then offers several solutions that will fix the problem.



Text Substitution Error Messages

Any error messages that you write need to be as informative to the user as possible. You can accomplish this through text substitution, which allows the system to embed variable text (such as dates, amounts, and so on) into a message that will be substituted at run time. You set up text substitution messages using the data dictionary, which helps ensure consistency of jargon and terms used. For example, the message “Voucher Batch 2453 contains errors,” uses the value 2453 as a parameter to the message. This value is substituted at runtime, and the rest of the information from the message is kept in the data dictionary error glossary. This message is only the short description in the data dictionary. When you open the message, the complete data dictionary glossary with text substitution appears.

See Also

- *Creating a Text Substitution Error Message* in the *Error Handling* section of this guide for procedures on creating interactive and batch message data items, defining the glossary text for a run time message, and creating a new data structure for the message.

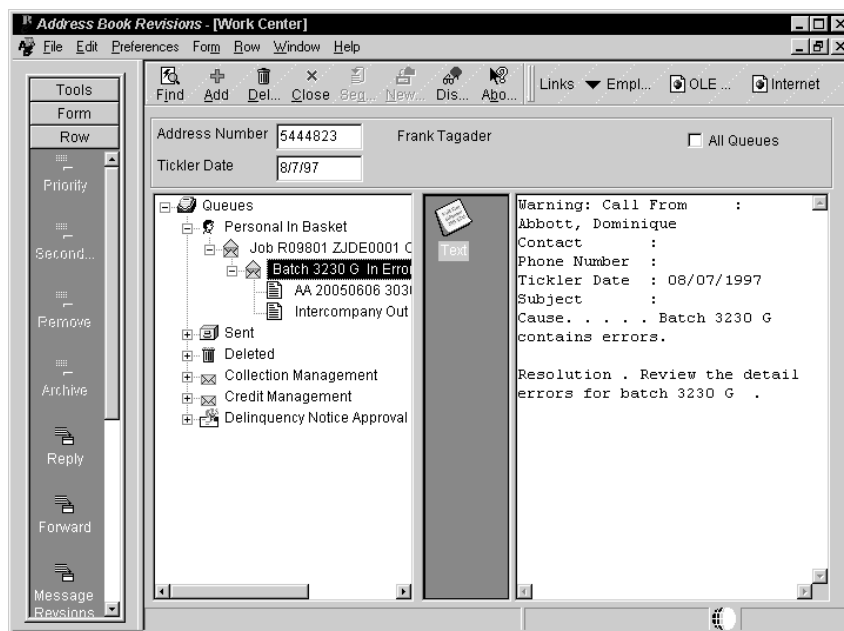
Action Messages

After a user reviews errors and decides what needs to be fixed, the user usually needs to go to a revision program to fix the errors. The Work Center allows you to set up level-break messages that will give the user the ability to fix the errors by calling the corresponding revision program directly from the Work Center. The term “action message” refers to the ability to call a OneWorld application and to pass the variables needed to that application. For example, the user would be able to automatically load the OneWorld Voucher Revision form with the record in error by clicking a shortcut within the error message. It is up to you to call the appropriate application and pass the correct values to that application. Note that in some cases it may not make sense to call a particular program for a given level. Action messages are highlighted in the grid to differentiate them from nonaction messages.

Understanding How Level Break Messages Work

Level break and action messages are used to group together (or package) one or more errors.

All individual messages are level-break messages except for the final error messages. The Work Center API creates and manages Job Completed level-break messages, but you create all other messages. The level-break messages are not error messages. They are packaging or categorizing messages that you set up to communicate to the user information about in which batch, document, line, and so on the error occurred. You set these messages using the `jdeSetGBRError` or the `jdeSetGBRErrorSubText` function.



In this example, “Job R89004 ZJDE0001.c” and “Batch 3230 G in Error” are level break messages. “Job R89004 ZJDE0001.c” is a level one message that is automatically generated by the system and “Batch 3230 G in Error” is a level two message. “AA 20050606 3031” and “Intercompany Out of Balance” are actual error messages.

The level-break message consists of two distinct pieces. The first piece of a level-break message is the text for the message, and the second indicates whether the message will be an action message.

All level-break messages contain the text piece, but they might not all be enabled as action messages. The decision to enable the action message piece of the level-break message is left to the report designer.

You use appropriate level breaks within your design to group related messages. It is at these level breaks that the Work Center API should be called. Usually a level break should be set up any time a record is no longer processed and a new record is about to be read. This concept is consistent even if the report has several nested record breaks.

Work Center API

When you call the Work Center API, it assumes a child/parent order. In other words, when the API is called, it assumes that any error that is in the run time error message stack belongs to the level sent into that instance of that API call. This means that all of the errors in the error space at that time, whether they have been set through business functions or through OneWorld event rules, are packaged or grouped together as children of the level that was passed to the Work Center API. These error messages are then cleared from the error space so that the next group of messages can be made based on a new set of records.

The timing of the calls to the Work Center API is critical. Most likely the reporting program will start by editing the header-level record, which will lead to a set of detail records. The detail records are the first to be read and processed. Thus the calls to the Work Center API will most likely be sending level-break numbers in descending level-break order.

For example, the actual series of level-break calls to the API might look like this: 4,4,4,3,4,4,3,2,4,4,3,2,1.

This series shows that the call structure started four levels down. The first call at level 4 would let the Work Center API find any messages that have occurred at that time and create child messages using the level 4 message as the parent. Note that if no errors occurred, then no messages would have been created. This call sequence example shows that the API was called at a level 3 after three calls to level 4. When the call to level 3 is made, the Work Center API remembers if any level 4 messages have been written or not. In other words, if no errors occurred when any of the level 4 calls were made, then the Work

Center API will not create the level 3 message. If there was at least one error at any of the level 4 calls, the level 3 and the level 2 message will be created.

You must call the Work Center API at every level. As explained earlier, the Work Center error messages are created based on a parent/child structure. Therefore, if a level call is skipped, the API has no way to group the child messages and child levels already created.

For example, the following level call structure is valid: 6, 6, 5, 4, 3, 4, 4, 3, 2, 1. The call sequence 6, 6, 4, 3, 4, 4, 3, 2, 1 is invalid because when level 6 is called, there is no call to level 5.

The Work Center API must be called with a level 1 when the reporting job is about to complete. Hence, level 1 is the parent to all errors and level-break messages and issues the “job completed” message. The level 1 call to the Work Center API is essential. A level 1 call to the API will not only ensure that no orphan Work Center records are created, but will also clean up all allocated storage used by the Work Center system. The level 1 call to the API should only occur once in the report. Generally it is done in the End Section event of the primary section of the report.

Creating a Level-Break Message

Level-break messages act as a container for error messages. You can create level-break messages that are active messages or nonactive messages, as explained in *Understanding Batch Error Messaging*.

You create level-break messages by creating data dictionary items, error data structures, business function error structures, and business functions. For additional information about these topics, refer to the related sections in this guide.

This section describes the following:

- Creating a data dictionary item for a level-break message
- Creating a data structure for the data dictionary item
- Creating a level-break message business function data structure
- Creating a level-break business function
- Calling the Work Center initialization API
- Calling the processing Work Center APIs
- Terminating the process

Creating a Data Dictionary Item for a Level-Break Message

The data dictionary item is the text portion of the level-break message. Before you create a data dictionary item, you may want to review the level-break messages that have already been created. It is possible that the level-break message that you want to use already exists. This is the message you see in the workcenter.

To create a data dictionary item for a level-break message

1. To search through the list of existing level-break messages, use Work with Data Dictionary Items (P92001).

Work With Data Dictionary Items - [Work With Data Items]

Search Description: LM0025

Data Item	Alpha Description	Alias	Glossary Group	System Code
Capacity	Load	CPCY	D	19
LoadCache	Load Cache	PRLOAD	D	H93
LoadCode1	Load Code 1	BU21	S	19
LoadCode10	Load Code 10	BU30	S	19
LoadCode11	Load Code 11	BU31	S	19
LoadCode12	Load Code 12	BU32	S	19
LoadCode13	Load Code 13	BU33	S	19
LoadCode14	Load Code 14	BU34	S	19
LoadCode15	Load Code 15	BU35	S	19
LoadCode2	Load Code 2	BU22	S	19
LoadCode3	Load Code 3	BU23	S	19
LoadCode4	Load Code 4	BU24	S	19
LoadCode5	Load Code 5	BU25	S	19
LoadCode6	Load Code 6	BU26	S	19
LoadCode7	Load Code 7	BU27	S	19
LoadCode8	Load Code 8	BU28	S	19

For every level-break message created, there is a data dictionary item and at least one business function created to reference that item. To find out what level-break messages already exist, use query in the Object Management Workbench (OMW) to locate all business functions (BSFN) that start with the first three letters, BLM.

Object Management Workbench - [Object Librarian Search and Select]

Object Name	Description	System Code	Reporting System Code	Object Type	Object Use	Object Category	Function Type	OCM Category	A Pri
BLM0001	Set Level Batch	00	00	BSFN			2		
BLM0002	Set Level Store & Forward Voucher	04	04	BSFN			3		
BLM0003	Set Level Voucher	04	04	BSFN			3		
BLM0004	Set Level Payment	04	04	BSFN			3		
BLM0005	Set Level Document	00	00	BSFN			2		
BLM0006	Set Level Store & Forward Voucher Line	04	04	BSFN			3		
BLM0007	Set Level Store & Forward Voucher Jour	04	04	BSFN			3		
BLM0008	Set Level Store & Forward Voucher Jour	04	04	BSFN			3		
BLM0009	Set Level S/F Journal Entry	09	09	BSFN	330		3		
BLM0010	Set Level S/F Journal Entry Line	09	09	BSFN	330		3		
BLM0011	Set Level Invoice	03B	03B	BSFN			3		
BLM0012	Set Level Store & Forward Batch	04	04	BSFN			3		
BLM0013	Set Level 3 Recurring JE	09	09	BSFN	330		3		
BLM0014	Set Level 4 Recurring JE	09	09	BSFN	330		3		
BLM0015	Batch Out Of Balance PPAT Error Messa	00	00	BSFN	330		3		
BLM0016	SetLevel_RequestForApproval	43	43	BSFN	330		3		
BLM0017	SendPPAT_RequisitionApproved	43	43	BSFN	330		3		
BLM0018	SendPPAT_RequisitionRejected	43	43	BSFN	330		3		
BLM0018A	SendPPAT_RequisitionRejectedMessage	43	43	BSFN	330		3		
BLM0019	Set Level - Order Level Errors	40	42B	BSFN			3		
BLM0020	Set Level - Line Level Errors	40	42B	BSFN			3		
BLM0023	Set Level 3 Error for Indexed Computatio	09	09	BSFN			3		
BLM0024	Set Level 4 Error for Indexed Computatio	09	09	BSFN			3		
BLM0025	Set Level A/R Receipt Pre-Post error	03B	03B	BSFN	330		3		
BLM0026	Set Level Store & Forward Invoice	03B	03B	BSFN	330		3		
BLM0028	Set Level Store & Forward Invoice Jour	03B	03B	BSFN	330		3		

The BLM naming convention refers specifically to J.D. Edwards-created level-break message business functions. J.D. Edwards recommends naming your own level-break message business functions using the following criteria: BLMxxyy where the xx refers to the system code you are using (between 50 and 55) and the yy is a unique number. This

unique number should be carried forward to other items. The maximum number of characters is eight.

Note: Use the description field to help you decide which existing messages may work in your report.

If you find several level-break message business functions that seem to fit your report design, write them down and verify the text on the data dictionary item itself, by using the data dictionary and inquiring on data items LM followed by the same numbers that followed the corresponding BLM function name.

Data Item	Description	LM* Alias	Glossary Group	System Code
CostCenterLastTrip	Depot - Prior Trip	LMCU	D	49
FutureUseLanguageMas	Future Use Language Master Date 1	LMSTDT1	D	H79
FutureUseLanguageMas	Future Use Language Master Date 2	LMSTDT2	D	H79
FutureUseLanguageMas	Future Use Language Master Date 3	LMSTDT3	D	H79
FutureUseLanguageMas	Future Use Language Master Flag 1	LMST1	D	H79
FutureUseLanguageMas	Future Use Language Master Flag 2	LMST2	D	H79
FutureUseLanguageMas	Future Use Language Master Flag 3	LMST3	D	H79
LaborLoadingMthd	Labor Distribution Method	LMTH	D	06
LaborLoadingMethod	Labor Distribution Method	LMTH	D	05
LAST_CLOSING_METER	Last Closing Meter Reading	LMEC	D	22
LAST_MAINTENANCE	Last Maintenance Date	LMND	D	28
LicenseRenewalMonth	License Renewal Month	LMON	D	12
LifetimeMileMeter	Lifetime Mile Meter	LMR	D	13
LimitExceededFlag	Limit Exceeded Flag	LMEX	D	52
ListMember	List Member	LMEM	D	80
LocalAreaMarital	Local Area Marital Status	LMST	D	05
LocalMaritalStatus	Local Marital Status	LMS	D	06
LocalMaritalStatus001	Local Marital Status 1	LMS1	S	05
LocalMaritalStatus002	Local Marital Status 2	LMS2	S	05
LocalMaritalStatus003	Local Marital Status 3	LMS3	S	05
LocalMaritalStatus004	Local Marital Status 4	LMS4	S	05
LoggingMessages	Logging Messages	LMSG	D	00
LotMasterCardexYN	Lot Master Cardex (Y/N)	LMCX	D	40

The LM naming convention refers specifically to J.D. Edwards-created level-break messages with text substitutions. These are glossary group Y. J.D. Edwards recommends naming your own level-break message with text substitutions using the following criteria: Lxxyy where xx refers to the system code you are using (between 50 and 55) and the yy is a unique number. The maximum number of characters is eight.

For example, if business function BLM0025 is a level-break message that you want to see the description for, inquire on data item LM0025.

2. If the text description meets your criteria, verify any substituted variables that the level-break message uses.

If the text in the LM message contains substituted variables (such as “&1”), you need to verify the data items of the data structure used by the level-break message. When you verify the data structure, it is important that data items in the structure are the exact data items that you will be passing in your report.

To verify that the data item is in the data structure, use the OMW and inquire on data structure DELM followed by the same numbers used before.

Object Name	Description	System Code	Reporting System Code	Object Type	Object Use	Object Category	Function Type	OCM Category	A Pri
DELM001	Set Level Batch	00	00	DSTR	360				
DELM002	Set Level Store & Forward Voucher	04	04	DSTR	360				
DELM003	Set Level Voucher	04	04	DSTR	360				
DELM004	Set Level Payment	04	04	DSTR	360				
DELM005	Set Level Document	00	00	DSTR	360				
DELM006	Set Level Store & Forward Voucher Line	04	04	DSTR	360				
DELM007	Set Level Store & Forward Voucher Jour	04	04	DSTR	360				
DELM008	Set Level Store & Forward Voucher Jour	04	04	DSTR	360				
DELM009	Set Level SF Journal Entry	09	09	DSTR	360				
DELM010	Set Level SF Journal Entry Line	09	09	DSTR	360				
DELM011	Set Level Invoice	03B	03B	DSTR	360				
DELM012	Set Level Store & Forward Batch	04	04	DSTR	360				
DELM013	Set Level Recurring	09	09	DSTR	360				
DELM014	Set Level Recurring Line	09	09	DSTR	360				
DELM015	Batch Out Of Balance PPAT Error Messa	00	00	DSTR	360				
DELM016	SendRequestForApproval	43	43	DSTR	360				
DELM017	Send PPAT to Approver	43	43	DSTR	360				
DELM018	SendRejectionNotice	43	43	DSTR	360				
DELM018A	SendPPAT_RequisitionRejectedMessa	43	43	DSTR	360				
DELM019	Set Level - Order Level Errors	40	40	DSTR	360				
DELM020	Set Level - Line Level Errors	40	40	DSTR	360				
DELM021	Set Level - Job Completed Message	00	00	DSTR	360				
DELM022	Set Level - Job Completed With Errors	00	00	DSTR	360				
DELM023	Indexed Computations Level 3 Data Stru	09	09	DSTR	360				
DELM024	Indexed Computations Level 4 Error Dat	09	09	DSTR	360				
DELM025	Set Level AR Receipt PrePost error	03B	03B	DSTR	360				

The DELM naming convention refers specifically to J.D. Edwards-created level-break message data structures. J.D. Edwards recommends naming your own level-break messages using the following criteria: DELMxxyy where xx refers to the system code you are using (between 50 and 55) and the yy is a unique number. This unique number should be the same number used previously. The maximum number of characters is eight.

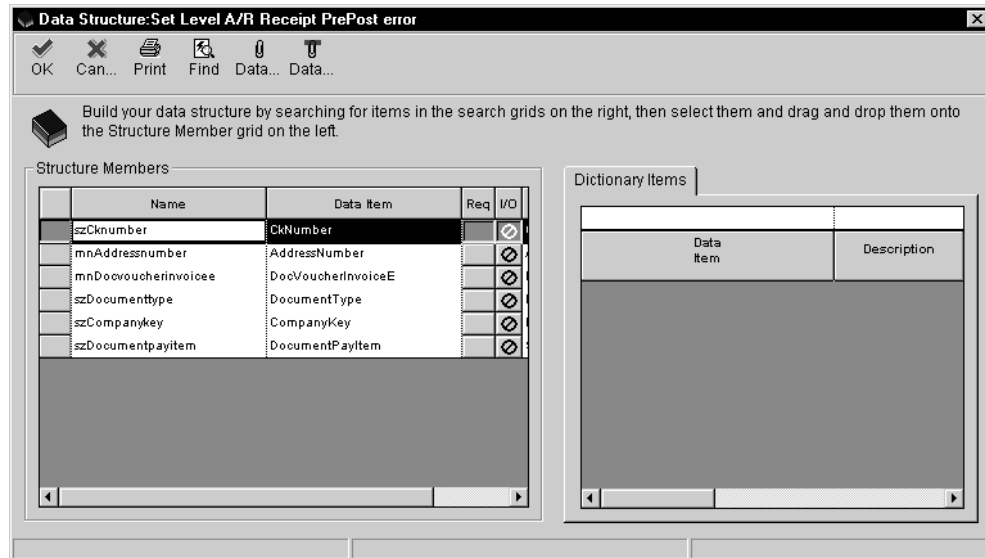
For example, DELM0025 would be the data structure created for the text substitution of level-break message LM0025. You may need to check the structure out to view individual fields.

Creating a Data Structure for the Data Dictionary Item

Each level-break message created requires a corresponding data structure to go with it. The data structure must be defined with the exact data items that will be substituted when the message is displayed. For example, a level-break message describing some error for Address Book Number would imply that the corresponding data structure would have data item AN8, the unique number that identifies an entry in the Address Book system. This restriction implies that many new data structures are going to be created.

To create the data structure, use OMW. The name should use the same unique number that was used when creating the Lxx data dictionary item. This number is to be appended to DELxx. For example, level-break message L55025 would use DEL55025 as the name of the data structure.

You will also need to create a typedef for this data structure. You include this typedef into the business function you create. The typedef converts the data structure to C so it can be used in the business function.



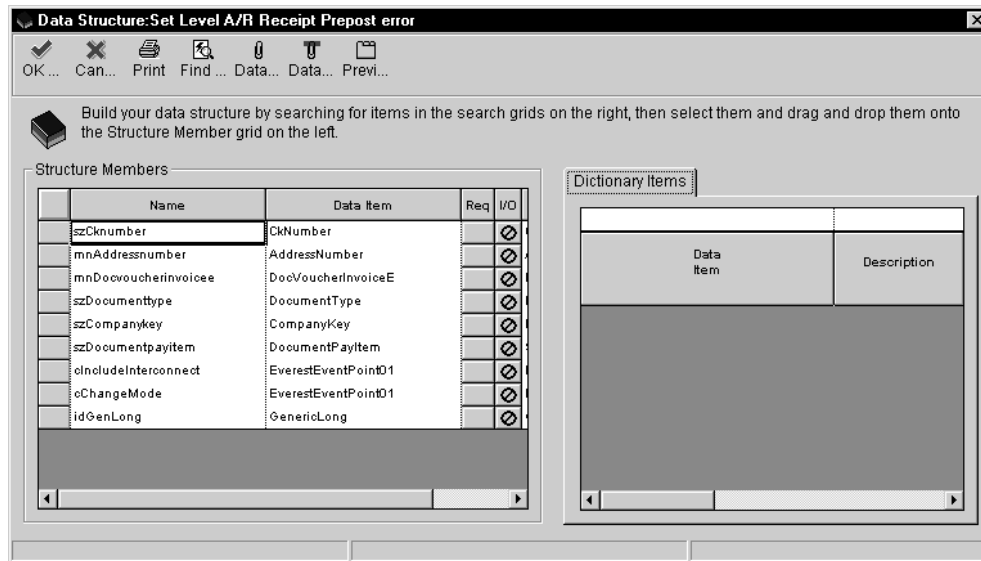
Refer to Data Structures for more information about data structures.

Creating a Level-Break Message Business Function Data Structure

You create business functions to deliver level-break messages. Each business function that you create needs to have several standard parameters in addition to the variables that are used for the text substitution and the variables needed for the message to be active. This means that you must create a second data structure to pass all of these variables.

The DLM naming convention refers specifically to J.D. Edwards-created business function data structures. J.D. Edwards recommends naming your own level-break messages using the following criteria: DLxxyy where xx refers to the system code you are using (between 50 and 55) and the yy is a unique number. This unique number should be the same number used previously. The maximum number of characters is eight.

For example, the J.D. Edwards-created level-break message LM0025 would use DLM0025 as the name of the business function data structure. You must have a business function for each level break message. This data structure is not to be confused with the data structure that was created for the data dictionary item in the previous procedure. The difference between the two is that the data structure for the business function is used to move data variables to the level-break function. The data structure used for the data dictionary item is used to store data that is mapped to the glossary for a particular data dictionary item.



When you create the data structure for the business function, include the following items:

- All data items used for the level-break text substitution message.
- All data items needed for the message to be active (that is, any variable or variables needed to load the form data structure for a given OneWorld application). Any data items that are used for the form interconnection only need to be renamed so that the letters FI_ appear after the Hungarian prefix. For example, jdFI_GLDate, or mnFI_Openamount. If you are not making the message active, you do not need to include these items.
- Add data item ev01 and change the variable name from OneWorldEventPoint01 to cIncludeInterconnect. This parameter is used as a flag to determine if the message is action. This parameter should be a common parameter to all level-break messages, even if the original intention of the level-break message is not to call a OneWorld application. This allows you to use level break messages in different applications, but not launch into application. You must have a 1 in the data structure value to launch an application.
- Add data item genlng and change the variable name from GenericLong to idGenlong. You use this parameter to control all Work Center messaging. It is not intended to be used for anything else other than as a work field for the system.

Creating a Level-Break Business Function

After you have created the DLxx data structure, you create the business function that processes the level-break errors and performs all of the mapping for the active message.

► To create a level-break business function

1. Create an object in the Object Management Workbench for the new business function.

The name of the business function is the unique number preceded by BLxx. For example, if you named the level-break message data dictionary item L55025, you would name the business function BL55025. The J.D. Edwards naming convention is BLMxx.

2. Enter the name of the actual function on the Business Function Design form within Object Management Workbench.

When you name the function, the standard is to start the function with the name SetLevel_xx, where xx refers to the system code. Append the function with other descriptive words to identify what the level-break message is designed to do. For example, you might name the business function BLM0025 Set Level A/R Receipt Prepost Error.

3. In Business Function Design, attach the business function data structure to it by highlighting the row and then choosing Parameters from the Row menu.

This data structure should be the DLxx data structure that you created earlier.

4. Choose Form Create to generate a .h file.
5. Copy and paste the business function's typedef into its header file.
6. Create a typedef for the text substitution data structure DExx. To create the typedef, you will need to create a "dummy" function line.
7. After you create the typedef, paste the data structure template into the Structure Definitions section of the business function header file.
8. Modify the source file for the level-break message by copying the source code of an existing BLMxxxx.C business function.
9. Once you have copied the source into the new level-break business function, review the copied source and make the appropriate changes.

Sample Source Code Highlights

The following sample of the shell source code shows which pieces need to be included and where you will be required to enter your own code.

You need to manually map fields from the business function's data structure to the dsTextData data structure. This is the data structure for text substitution in the level-break message. You also need to manually map fields from the business function's data structure to the dsFormData data structure. This is the data structure that is used for the active message.

In the "Variable declarations" section, you should have the following lines to declare the level-break message variables:

```
char  szForm[11];      /* Name of form in application */
char  szDDitem[11];   /* Data dictionary name of the level message */
char  szDLLName[11]; /* Name of the application DLL */
char  szDsTmp1[11];  /* Name of the text substitution data structure */
```

In the "Declare structures" section, you enter your own code for the appropriate type of level-break message. The following are examples from an existing business function:

```
DSDELM0002 dsTextData; /* Instance of text substitution structure */
FORMDSW0411Z1D dsFormData; /* Instance of form interconnect structure */
```

In the "Set pointers" section, you should have the following lines to ensure that the level-break message will work:

```
if (lpDS->idGenLong == (ID) 0)
{
    jdeSetGBRError (lpBhvrCom, lpVoid, (ID) 0, "4363");
    if (hUser)
    {
        JDB_FreeBhvr (hUser);
    }
    return ER_ERROR;
}
else
    lpDSwork = (LPDS_B0100011A) jdeRetrieveDataPtr (hUser, lpDS->idGenLong);
```

In the "Main Processing" section, you should have the following lines. Substitute your own items for the italicized items:

```
strncpy    ((char*) szDsTmp1, (const char*) ("DELM002"), sizeof (szDsTmp1) -1);
strncpy    (szDDitem, (const char*) ("LM0002"), sizeof (szDDitem));
memset     ((void*) (&dsTextData), (int) ('\0'), sizeof (dsTextData));
```

In the "Assign values from lpDS data structure to dsTextData here" section, you enter your own code for the appropriate level-break message. When you assign

values you map the business function data structure items to the data dictionary data structure items. The following are examples from an existing business function:

```
strncpy (dsTextData.szEdiuserid,(const char *) (lpDS-> szEdiuserid),
        sizeof(dsTextData.szEdiuserid));
strncpy (dsTextData.szEdibatchnumber,(const char *) (lpDS-> szEdibatchnumber),
        sizeof(dsTextData.szEdibatchnumber));
strncpy (dsTextData.szEditransactnumber,
        (const char *) (lpDS->szEditransactnumber),
        sizeof(dsTextData.szEditransactnumber));
```

In this example, *dsTextData.szEditransactnumber* is the data dictionary data structure item and *lpDS->szEditransactnumber* is the business function data structure item. Use the Strncpy API is used for strings and the Mathcopy API for math numerics. If you use memcpy for dates, the characters are assigned directly.

In the “Assign values from lpDS data structure to dsTextData here” section, you should have the following lines to ensure that the level-break message will work:

```
JDBRS_GetDSTMPLSpecs (hUser, (char*) szDsTpl, &lpDSwork->lpBlob->lpTSDSMPL);
if (lpDSwork->lpBlob->lpTSDSMPL != (LPDSTMPL) NULL)
{
    lpDSwork->lpBlob->lpTSTEXT=(LPSTR) AllocBuildStrFromDstmplName( (LPDSTMPL)
        lpDSwork->lpBlob->lpTSDSMPL, (char*) szDsTpl,
        (LPVOID) &dsTextData);
    strncpy (lpDSwork->lpBlob->szDDItem, (const char *) (szDDItem),
        sizeof(lpDSwork->lpBlob->szDDItem));
}
if(lpDS->cIncludeInterconnect == '1')
```

In the “Form interconnect processing” section, you enter your own code for the appropriate level-break message. The following is an example from an existing business function. In the “Form interconnect processing” section, you should have the following lines to ensure that the level-break message will work.

```
strncpy (szDLLName, (const char *) ("P0411Z1"), sizeof (szDLLName));
memset ((void *) (&dsFormData), (int) ('\0'), sizeof (dsFormData));
memset ((void *) (szForm), (int) ('\0'), suzeif (szForm))
strncpy ((char *) szForm, (const char *) ("W0411Z1D"), sizeof (szForm) -1);
```

In the “Assign values from LpDS data structure to dsFormData” section, you enter your own code for the appropriate level-break message. The following is an example from an existing business function. This shows how you pass information from the data structure for the business function to the data structure for the form.

```

strncpy (dsFormData.EDUS, (const char *) (lpDS->szEdiuserid),
        sizeof(dsFormData.EDUS));
strncpy (dsFormData.EDBT, (const char *) (lpDS-> szEdibatchnumber),
        sizeof(dsFormData.EDBT));
strncpy (dsFormData.EDTN, (const char *) (lpDS->szEditransactnumber),
        sizeof(dsFormData.EDTN));
ParseNumericString(&dsFormData.EDLN, "1.0");
dsFormData.EV01 = '1';

```

In the “Get the form data structure id from the SVRDTL table” section, you should have the following lines to ensure that the level-break message will work:

```

lptam = jdeTAMInit (FILENAME_SVRDTL);
strncpy ((char *) lpDswork->lpBlob->szForm, (const char *) (szForm), size of
        (lpDswork->lpBlob->szForm) -1);
if (lptam != (LPTAM) NULL)
{
    lpASVRdtl = TAMAllocFetchByKey (lptam, INDEX4_ASVRDTL, szForm, 1);
    if (lpASVRdtl != (void *) NULL)
    {
        JDBRS_GetDSTMPLSpecs(hUser, (char*)lpASVRdtl->szFITemplateName,
                            &lpDswork->lpBlob->lpFIDSMPL);
        if (lpDswork->lpBlob->lpFIDSMPL != (LPDSTMPL)Null)
        {
            lpDswork->lpBlob->lpFITEXT=(LPSTR) AllocBuildStrFromDstmplName
                ((LPSTMPL)
                 lpDswork->lpBlob->lpFIDSMPL,
                 (char*)lpASVRdtl->szFITemplateName
                 (LPVOID) &dsFormData);
            strncpy (lpDswork->lpBlob->szDLLName, (const char *) (szDLLName),
                    sizeof(lpDswork->lpBlob->szDLLName));
        }
        TAMFree(lpASVRdtl);
    }
    TAMTerminate(lptam);
}
}

```

In the “Function Clean Up” section, you should have the following lines to ensure that the level-break message will work:

```

if(hUser)
{
    JDB_FreeBhvr(hUser);
}
return (ER_SUCCESS);
}

```

Calling the Work Center Initialization API

After you create the business function, you must compile and check it in to the object librarian. Then, you must attach the Work Center APIs. The first step in

attaching the APIs is to call an event rule in the Work Center. You usually call this on the init section in the parent section.

► **To call the Work Center initialization API**

1. Within Report Design, create a global work field. When creating the work field, use the data dictionary item GENLNG.
2. Call the Work Center initialization function. The report finds this business function. This function is named InitializePPATapi (source file B0100025.C – this is the actual name). F01131 is EditJDEM Error Message.

Generally, this function is called in the primary section of the report using the Initialize Section API.

Use the following table to identify which parameters to send:

Parameter	Description	Allowed Values	Typical Values
szUserid	User ID override. If not passed, the user who ran the UBE will receive all messages. If this field is filled in, it will supersede the following Address Number override parameter.	Leave blank or use any valid user ID. If left blank, all messages go to the UBE submitter.	Blank value
mnAddressnumber	User ID override. This field would be used if the user ID is not known but the address for that user is.	Leave blank or use any address number of a valid user. If left blank, all messages go to the UBE submitter.	Blank value
cDoNotLogWarnings	If this feature is turned on, no warning messages will be sent to the Work Center.	Leave blank or use the number "1." If left blank, all warnings and errors are processed.	Blank value
cAllowUserIDToChange	If this feature is turned on, you can change the user who will be receiving the messages on the fly. Some restrictions do apply.	Leave blank or use the number "1." If left blank, all messages will be sent to one user.	Blank value

szMailboxdesignator	Mailbox (or queue) override. If this field is turned on, then messages will be sent to the overridden mailbox. (Work Center use.)	Leave blank or use any valid mailbox designator. If left blank, all messages will be written to the default queue.	Blank value
idPPATworkField	Work field used by the Work Center API. This is where the "GENLNG" work field should be passed.	This must be a work field defined in the UBE. Use the GENLNG data item for this.	The GENLNG work field
cSendErrorsToReport	If this feature is turned on, no messages (except "Job Completed") will be sent to the Work Center. The messages will then be available to read through another Work Center function.	Leave blank or use the number "1."	Blank value

3. Choose the F01131 Edit JDEM Error Message function within the API.

cAllowUserIdToChange Parameter

This parameter on the initialize API works in conjunction with the szUserId parameter on the ProcessErrorsToPPAT API. It allows you to set up the UBE so that when any batch errors are encountered, errors are sent to the user who created the original records and not to the person submitting the job (such as the night operator). For example, if a single batch job contains 1,000 transactions created by 50 users, then only those users who created transactions with errors will receive error messages. The night operator will still receive a message, but it would be a message such as "Job completed normally" or "Job completed normally with errors." Other users whose transactions were successful will not receive any error messages.

To set up this functionality, you need to enter a "1" in the cAllowUserIDToChange parameter when you initialize the batch error processing system. When you process the level 2 level-break message and then call the ProcessErrorstoPPAT API, you can still specify who will receive the messages by using the szUserId parameter. You can determine who should receive the message by looking at the transaction record.

Calling the Processing Work Center APIs

After the Work Center system has been initialized, you must determine the various level-break points within the report and call the Work Center system at each of these points to group the errors.

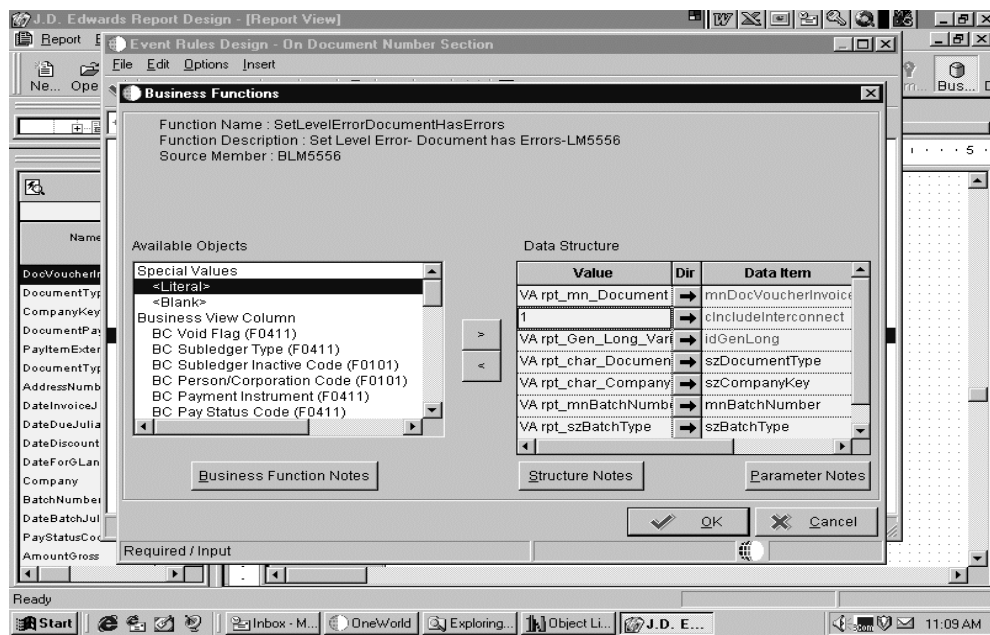
► To call the processing Work Center APIs

1. From Report Design, establish all of the appropriate level-breaks for the reporting batch job.

You need to analyze the events that will logically group all errors at a given event. This typically happen at events in which all editing has been completed for a group of records or immediately after all edits for an individual record have taken place.

2. For each level-break established, do the following:
 - Call the level-break message business function at the appropriate level-break. The level-break message business function should relate to the type of error grouping that you want to occur at the particular level-break. For example, SetLevel_SFVoucher would group errors related at the voucher level-break. For reports, this business function is typically called in the Do Section. If the interconnect is blank then it is not calling an action message.

Note: The level you send the API will never be 1; it is only sent once, when you terminate the process.



- Call the Work Center error message function immediately after the call to the level-break message. This function is called ProcessErrorsToPPAT. B0100011.c processes batch errors to JDEM and makes repeated calls.

Use the following table to identify which parameters to send. This is where you designate a level for the data dictionary level break message. Start your message with a level 2. The system automatically generates a level 1.

Parameter	Description	Allowed Values	Typical Values
mnLeveloftotaling	This parameter is the message level that you want the Work Center API to process.	Any valid number from 1–20. Note that the number “1” should only be used when the UBE is almost finished. A “1” will write the “Job Completed” message.	Numbers 1–20
idDataBaseWorkField	Work field used by the Work Center API. This is where the “GENLNG” work field should be passed.	This must be a work field defined in the UBE. Use the GENLNG data item for this.	The GENLNG work field

Parameter	Description	Allowed Values	Typical Values
cErrorPreProcessFlag	This parameter allows the UBE to call the Work Center API at some event to flush the “error space” and to group any errors that are there by a level break message, which has yet to be sent.	Leave blank or use the letter “P.” If left blank, errors that are in the “error space” are considered to be grouped by the level message being sent in the first parameter.	Blank value Default
szUserid	When the “cAllowUserIDToChange” flag has been set in the initialization function, this parameter will accept the new user ID.	Leave blank or use any valid user ID. If left blank, all messages are written to the last user ID used.	Blank value Default

Terminating the Work Center Process

After all messages have been sent to the Work Center, you must terminate the Work Center process before the reporting job is finished.

► **To terminate the Work Center processes**

When the batch program is about to terminate, call the Work Center error message function, ProcessErrorsToPPAT, one last time, sending it to level 1. The level 1 indicates the level of totalling is equal to 1 and it is completed. This will create the “job completed” message and will free any work space created by the Work Center API.

It is important that every report design using the Work Center API to process errors calls the API at the end of processing using a level 1. This should also be done by reporting jobs that are monitoring for any critical errors and need to terminate early.

When the report has finished processing, it creates Work Center messages, which can then be read using the OneWorld Work Center application. Batch

errors are processed to JDEM system. Messages created are sent to the user who ran the report unless you specify that the message be sent to another user or administrator.

If no errors are encountered, the API sends the message that the job completed successfully to the Work Center.

Debugging



Debugging

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

You can use a debugger to stop program execution so you can see the state of the program at a specific point. This allows you to view the values of input parameters, output parameters, and variables at the specified point. When program execution has been stopped, you can step through the code line by line to check such issues as flow of execution and data integrity.

This section describes the following:

- Working with the Event Rules Debugger
- Debugging Business Functions Using Visual C++
- Working with Visual C++ Debugger
- Debugging Strategies

In OneWorld, there are two debugging tools you can use:


- OneWorld Event Rules Debugger
- Microsoft Visual C++ Debugger

You use the Event Rules Debugger to debug event rules and the Visual C++ Debugger to debug C business functions. You can use the Event Rules Debugger to debug:

- Interactive applications
- Reports
- Table Conversions

Overview of the Debugging Process

The debugging process consists of determining where problems occur and fixing those problems. You should isolate each problem to a particular area, and then examine exactly how the program operates in that area.

If you make a change in your program while you are debugging with the Event Rules Debugger, you must: 

- Stop the Event Rules Debugger
- Rebuild debug information
- Reset breakpoints
- Run the application

Following are some features available in the debuggers you can use:

Go	<p>Visual C++ and Event Rules Debugger – Restarts program after a breakpoint has been reached.</p> <p>Visual C++ – Once you start an application, it will run until the breakpoint is reached.</p> <p>Event Rules Debugger – The application must initially be started from OneWorld Explorer.</p>
Breakpoints	<p>Visual C++ and Event Rules Debugger – Breakpoints 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.</p>
Step	<p>Visual C++ and Event Rules Debugger – The Step command will execute the current line of code. It 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 that line is executed.</p>
Step Into	<p>Visual C++ and Event Rules Debugger – This is 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. In OneWorld, the Step Into command can be used to debug a second application that is called from within a OneWorld calling application.</p>
Skip	<p>Visual C++ and Event Rules Debugger – Skip is used to skip execution of a line of code. The line of code which is not executed will turn red.</p>
Step Over	<p>Executes a single line of code. If the line of code is a function call, report interconnection, or form interconnection then that call will not be stepped into.</p>

Stop

This stops the debugging process.

If you stop the Event Rules Debugger, the application continues to run, as if the debugger had not been started.

In Visual C++ the debugging process stops and the running application is also terminated.

Watch

Visual C++ and Event Rules Debugger – This lets you display the value of variables while the program is running. It also lets you inspect expressions, so you can see how a particular expression changes when variables change.

Interpretive and Compiled Code

OneWorld uses both interpretive and compiled code.

Interpretive code refers to code that is compiled as it runs. The translation from program instruction to machine instruction happens at runtime. Interpretive code lies within the application. Event rules scripting code used within Form Design and Report Design is interpretive. Interpretive code allows you to customize without going through a compile process every time you change something.

Compiled code is compiled and stored in an object file that may be called independently. The translation from program instruction to machine instruction happens at compile time. For example, table event rules, named event rules, and business functions are compiled. They are outside the application. They are also less subject to change. You can use Visual C++ to debug compiled code. The event rules scripting language can be translated into C, Java, or your current language of choice. The logic only needs to be interpreted one time. Interpretive code is more flexible.

Working with the Event Rules Debugger

The Event Rules Debugger provides the essential debugging tools (breakpoints, step commands, and variable inspection), you need to debug a OneWorld interactive or batch application. You can use the Event Rules Debugger to debug named event rules and table event rules. When the Event Rules Debugger builds debug information for an application, it includes named event rule and table event rule information for that application.

There are two main steps to set up and use the Event Rules Debugger:

- Running the Event Rules Debugger and setting breakpoints
- Running and debugging the application, report, or table conversion

If you want to save the debug information table but do not want the Debugger active during your work, you can deactivate debug information. You can activate the table at any time to continue debugging.

Working with Event Rules Debugger includes:

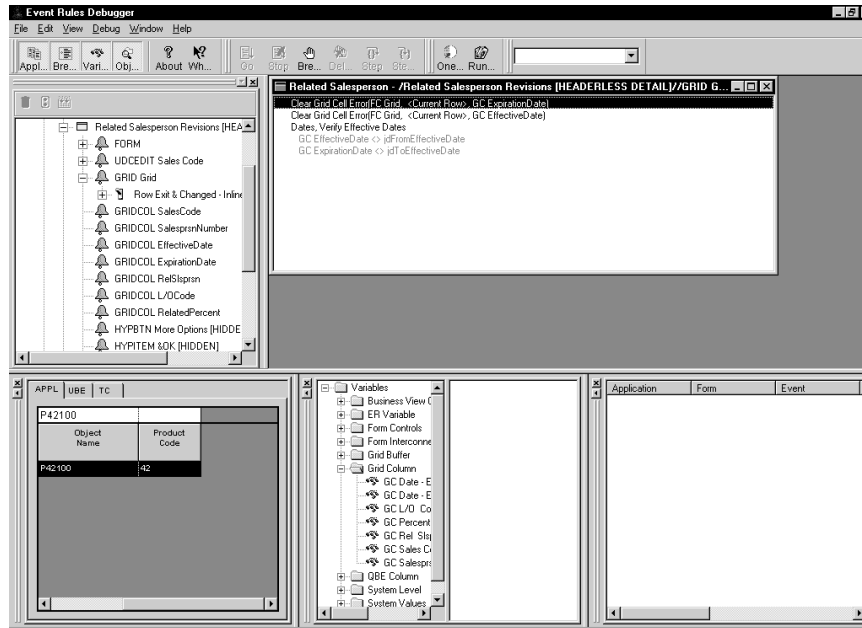
- Understanding the Event Rules Debugger
- Debugging an application

Understanding the Event Rules Debugger

The Event Rules Debugger is a standalone tools program that consists of five main controls:

- Object Browse window
- Event Rules windows
- Breakpoint Manager
- Variable Selection and Display window
- Search combo box

All windows except the event rule windows are dockable to any side of the main application. You can right-click on a window to dock it or to hide it. If you close the Debugger, it will retain your docking settings so they are the same the next time you run the Debugger.



Object Browse Window

The Object Browse window lists applications that have debug information built and that are available for debugging. You can drill down through the tree to a specific event and open an Event Rules window for that event.

Event Rules Window

The Event Rules window displays the event rules for one event. The event name and path are displayed in the title bar for each Event Rule window. You can right-click to toggle the title between its long and short versions. The Event Rules windows show which line in the event rules is currently being executed.

The left hand side of an Event Rule window displays icons that describe the state of a line in the event rules, including the following states:

- Breakpoint
- Disabled
- Current line of execution

Through the Event Rules window you can set and remove breakpoints, using any of the following methods:

- Double-click on the line in event rules
- Choose a line and from the Debug menu choose Breakpoints
- Right-click on a line and choose breakpoints from the menu that appears
- Choose a line and press F9

Variable Selection and Display Window

The Variable Selection and Display window consists of two views. The left view contains a tree control that lists the variable types as parent nodes and the current variables of that type as child nodes. The variables displayed are relative to the current Event Rule window. The right view is the variable display view. This view displays user selected variables and their current values. You can add a variable to the Variable Display view by double-clicking the desired variable in the Variables Tree.

You can change the value of variables while you are debugging an application. To change the value of a variable, double-click on the variable in the Variable Display window. Change the value of the variable. The new value appears in the Variable Display window. If you enter an inappropriate value, for example if you change a numeric value to an alpha value, the new value is not set and the value is not changed.

There are three special values that are displayed for variables:

blank	The value for the variable contains only blanks. This applies to string and character types only.
null	The value for the variable is set to ASCII “\0”.
unknown	The value for the variable could not be obtained from the engine. This happens when the applications are not running or the variables are out of scope.

Note: Variable inspection and modification is not available for debugging named event rules and table event rules.

Breakpoint Manager

Breakpoints tell the debugger where or when to break execution of a program. When the program is halted at a breakpoint, you can examine the state of your runtime structures, step through your event rules, and evaluate expressions using the Variable Watch window.

The Breakpoint Manager tracks which breakpoints are set and where they are located in an application. When you set a new breakpoint, an entry is made in the Breakpoint Manager. That entry contains the application name, form name, event name, and event rule line.

Right-click in Breakpoint Manager to perform the following operations:

- Delete breakpoints

- Delete all breakpoints
- Go to a breakpoint

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

Search Combo Box

You can use the Search combo box on the toolbar to search for event rule text. Enter the text you wish to search for in the Search combo box and either press Enter or F3. If the search text is found in your event rules text, the text is highlighted. If you press Enter or F3 again, the next occurrence of your search text is highlighted.

The search control allows for 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”.

Following are the special characters you can use to perform more advanced searches.

^	The caret (^) indicates the beginning of the string. For example, the expression “ <code>^</code> ” matches an “A” only at the beginning of the string.
^	The caret (^) immediately following the left bracket ([) is used to exclude any remaining characters within brackets from matching the target string. For example, the expression “ <code>[^0-9]</code> ” indicates that the target character should not be a digit.
\$	The dollar sign (\$) matches the end of the string. For example, the expression “ <code>abc\$</code> ” will match the substring “abc” only if it is at the end of the string.
 	The alternation character () allows the expression on either side of it to match the target string. For example, the expression “ <code>a b</code> ” 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.

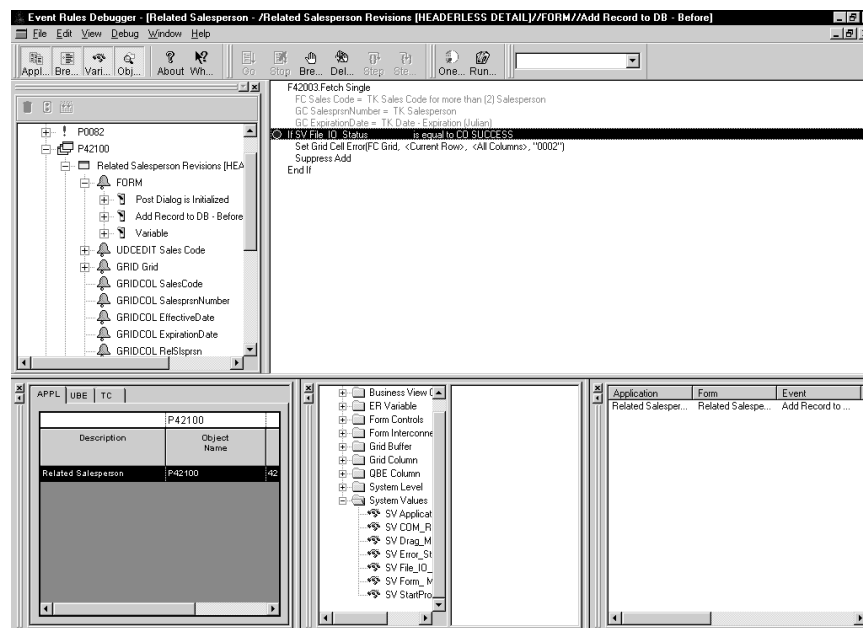
- + The plus (+) is similar to the asterisk, except that there should be at least one match of the character to the left of the + sign in the expression.
- ? The question mark (?) matches the character to its left 0 or 1 times.
- () The parenthesis affects the order of pattern evaluation and also serves as a tagged expression that you can use to replace a matched substring with another expression.
- [] Brackets ([and]) enclosing a set of characters indicate that any of the enclosed characters may match the target character.

Debugging an Application

The Event Rules Debugger allows you to debug interactive or batch applications.

▶ To debug an application

1. From the Cross Application Development Tools menu (GH902), open Debug Application.



2. Choose the object you want to debug.
3. Choose a form (for interactive applications) or section (for batch applications) and an event to view.
4. Choose the event rule line where you want to set a breakpoint.
5. From the Debug menu, choose Breakpoint.

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

You can remove the breakpoint by choosing Breakpoint from the Debug menu. The options on the Debug menu toggle on and off.

6. Minimize, but do not close, Debugger.
7. From OneWorld Explorer or from the Object Librarian, run the application you selected to debug.

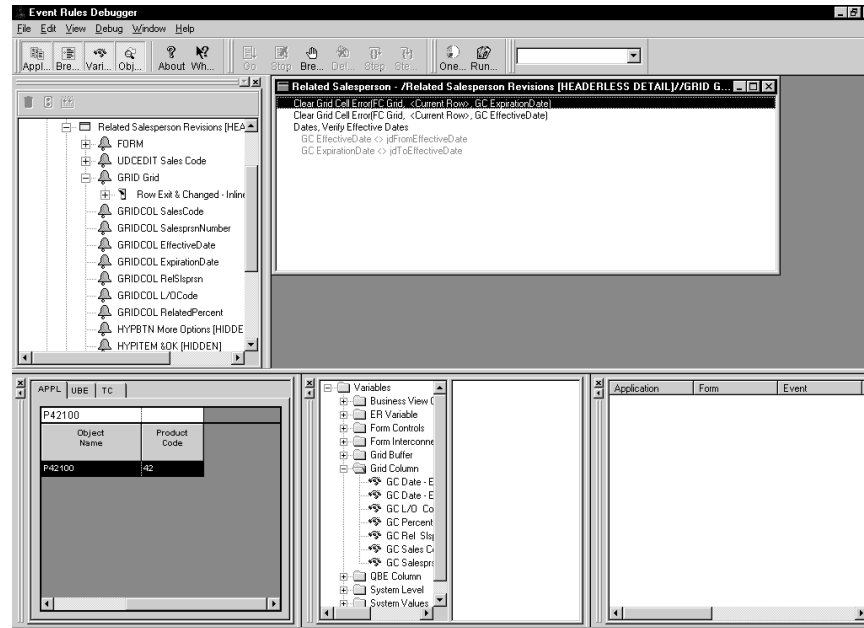
As your application encounters a breakpoint, the application pauses and displays the Event Rules Debugger.

While execution is stopped you can use the variables view to inspect and modify the values of runtime structures.

8. From the Debug menu, choose one of the following options:
 - Go
 - Stop
 - Step Over
 - Step Into

Inspecting or Modifying a Variable

As you debug an application and encounter a point at which the interactive or batch application fails, modify the appropriate variable to correct it. Save your modifications and rerun the application to see if further debugging is required.



► To modify a variable

1. On event Rules Debugger, double-click a variable in the Event Rules variables window.
2. Revise information in the following field:
 - New Value

Just in Time Debugging

The Event Rules Debugger also uses “Just in Time Debugging.” Just in Time Debugging occurs after you build the debug information, run the Event Rules Debugger, and then run the application.

If the runtime engine has a problem resolving an event rule object or calling a business function, a message box appears that allows you to activate the Debugger. If you choose to activate the Debugger, then the Debugger is brought to the top and the event rule line that could not be processed is displayed with a yellow arrow to the left of it.

Setting Breakpoints

You should set at least one breakpoint on a starting event, such as *Dialog is Initialized*. Any event that you want the Debugger to stop on must have at least one line of event rule code.

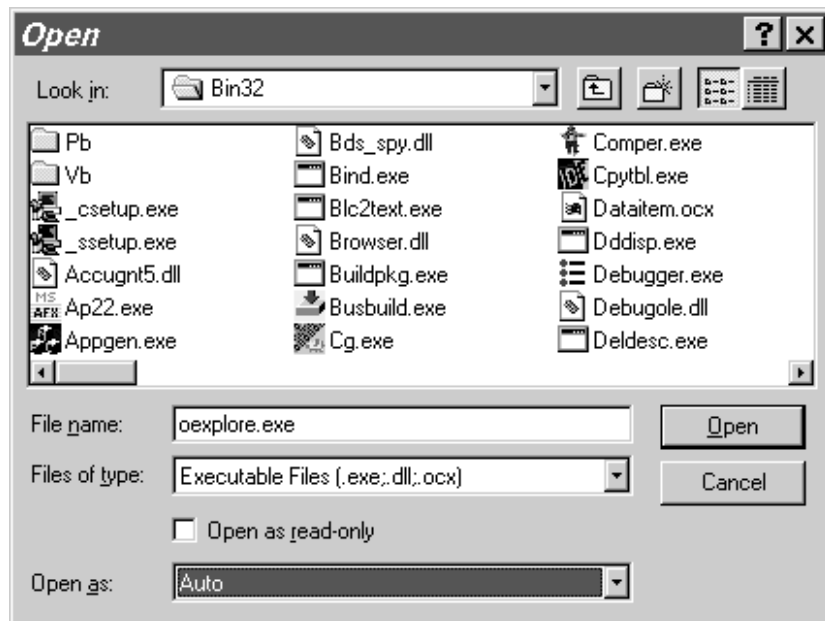
Debugging Business Functions Using Microsoft Visual C++

You can use Microsoft Visual C++ to debug business functions that are written in C. Starting with OneWorld B7331, you must use version 6.0 of Microsoft Visual C++. You can debug business functions attached to interactive applications or to batch applications.

Debugging Business Functions Attached to Interactive Applications

► To debug a business function attached to an interactive application

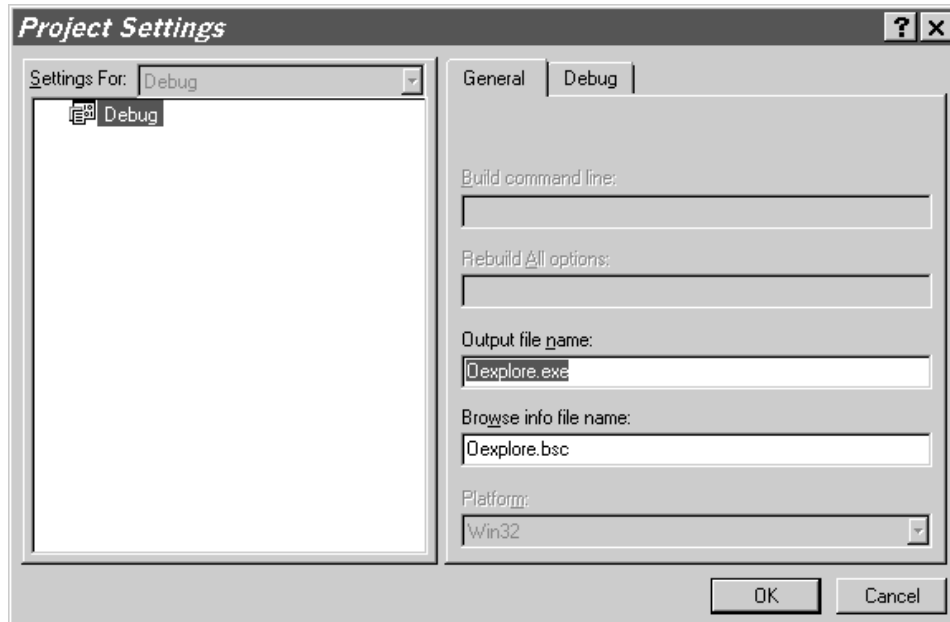
1. Make sure that OneWorld is not running. It must be closed to debug in this manner.
2. Open Visual C++ and make sure that all workspaces have been closed.
3. From the File menu, choose Open.
4. Choose “List Files of Type” to accept executables (.exe).
5. Select your OEXPLORE.EXE on path \b7\System\bin32 and click OK.



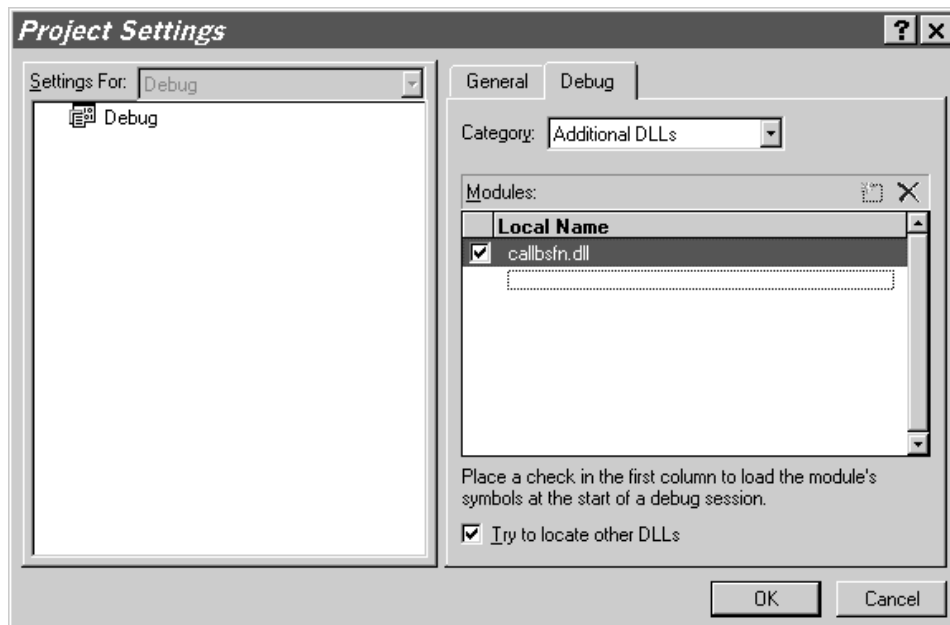
This creates a project workspace.

6. Choose your OEXPLORE.EXE project heading.

7. From the Project menu, choose Settings.

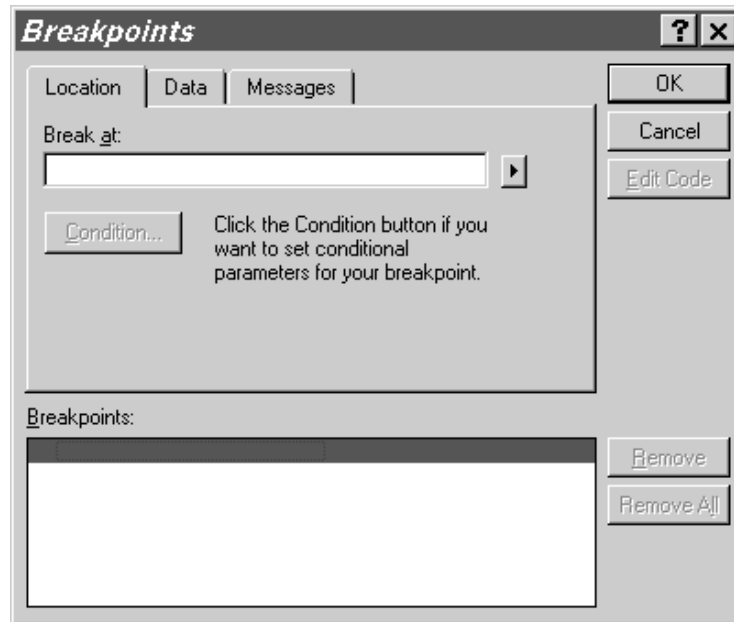


8. Click the Debug Tab.



9. In the Category list, choose Additional DLLs.
10. Click the Browse button to select the CALLBSFN.dll or other appropriate DLL on path \b7\bin32. (Note that this path may be different depending on your path code.)
11. Click OK.

12. Choose your .h and .c files for the source you want to debug from file open.
13. From the Edit menu, choose Breakpoints to set breakpoints in your code.



If a “cannot open *.pdb” message appears, click through it.

If a message appears notifying you that breakpoints have been moved onto the next valid lines, it may indicate a source code and object mismatch. You may need to rebuild your business function.

14. From the Build menu, choose Start Debug, Go.

This causes the OneWorld signon box to appear.

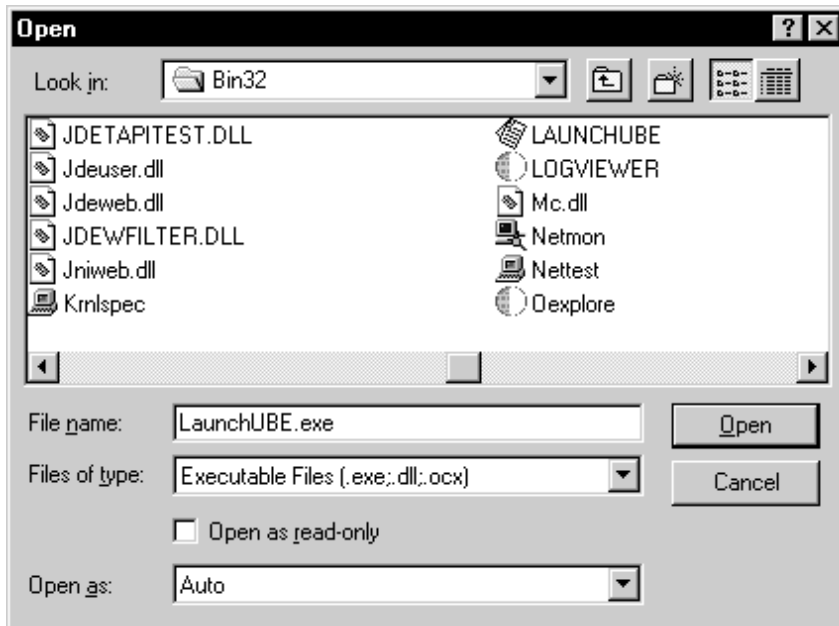
15. Log into OneWorld as you normally would.
16. Execute the application. When it reaches the business function in Debug, it will open or display the C code in Visual C so that you can step through it.

If you are debugging business function event rules and C business functions, you can use the OneWorld debugger and the Visual C++ debugger together. Follow the steps above until you log into OneWorld. At this point follow the steps for the OneWorld debugger. Program execution stops if C code is accessed. You can then use Visual C++ to continue debugging. This is useful if you are trying to locate a problem and you are not sure whether the problem is in a C business function or the application calling the business function.

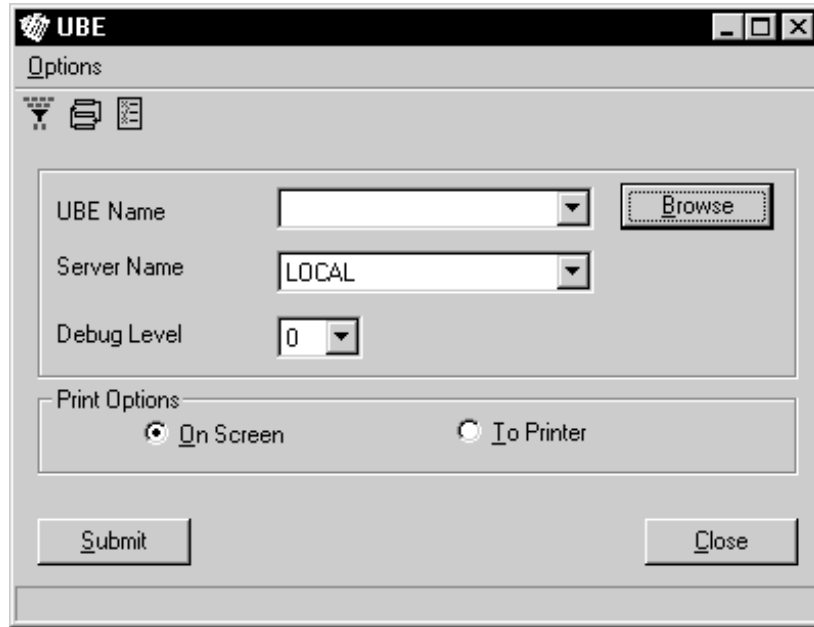
Debugging Business Functions Attached to Batch Applications

► **To debug a business function attached to a batch application**

1. Make sure that OneWorld is not running. It must be closed to debug in this manner.
2. Open Visual C++ and make sure that all workspaces have been closed.
3. From the File menu, choose Open.
4. Choose “List Files of Type” to accept executables (.exe).



5. Select LaunchUBE.exe



6. On LaunchUBE browse for the report and version you wish to use.
7. Complete the Server Name field by selecting where the report will run.

You can choose Local or specify a server.

8. Set the Debug Level
9. Choose one of the following Print Options
 - On Screen
 - To Printer
10. Click Submit to submit the report.

Working with the Visual C++ Debugger

You must use the Microsoft Visual C++ Debugger to debug business functions written with the Event Rules scripting language or C code. You can run the entire OneWorld system through the Visual C++ debugger (that is, you can start the oexplore.exe file from within the Visual C++ Debugger). This allows you to “step out” of the CASE tool-generated application code into the business functions that are called in the event rules.

You can use the debugger to debug a C program and interactively stop and start it as needed. During debugging, you can check specific values of variables and parameters to ensure a program is running correctly. You can also step through the code to see what code is actually being executed.

The debug commands are listed in the Debug menu. You can customize the toolbar to contain debug buttons, which you can use instead of the menu.

Useful Features of the Visual C++ Debugger

The Go Command

You can run a program using the Go command from the Debug menu. The program will run until completion unless you set up breakpoints.

The Step Command

The Step command is available on the Debug menu and executes the current line of code. When the line of code has been executed, the yellow arrow cursor appears on the next line of code to be executed.

The Step Into Command

You can access the Step Into command from the Debug menu. Use this command when the current line of code contains a function call. The debugger steps 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. If the source code of the function to be stepped into does not exist on the workstation, the debugger skips over the line of code as though the Step command was used.

Stepping into a standard C function takes you into the function, which you may not want to do. If so, use the Step Over command to skip those functions.

Setting Breakpoints

You use breakpoints to run the program until it reaches a certain line of code. If a breakpoint is set, the Go command executes until that line of code.

You set a breakpoint by placing the cursor anywhere on the line of code to stop before. Select Breakpoints from the Debug menu. A red octagon, similar to a stop sign, appears to the left of the line of code where the breakpoint is set. When the program is run, all lines of code up to the breakpoint are executed. To continue execution after the breakpoint, you can use Step, Step Into, or Go.

Using Watch

You can use Watch to inspect what values variables are set to. To use Watch, double-click on the item to watch and drag it to the Watch window.

Locals Window

All local variables and parameters to a function are listed with their data types and values in the Locals window. You can modify the values of all items in the Locals window during debugging. This is useful when debugging infinite loops.

Example: Debugging with the Visual C++ Debugger

In the following example, a breakpoint, Watch, Locals, and the Step command have been used.



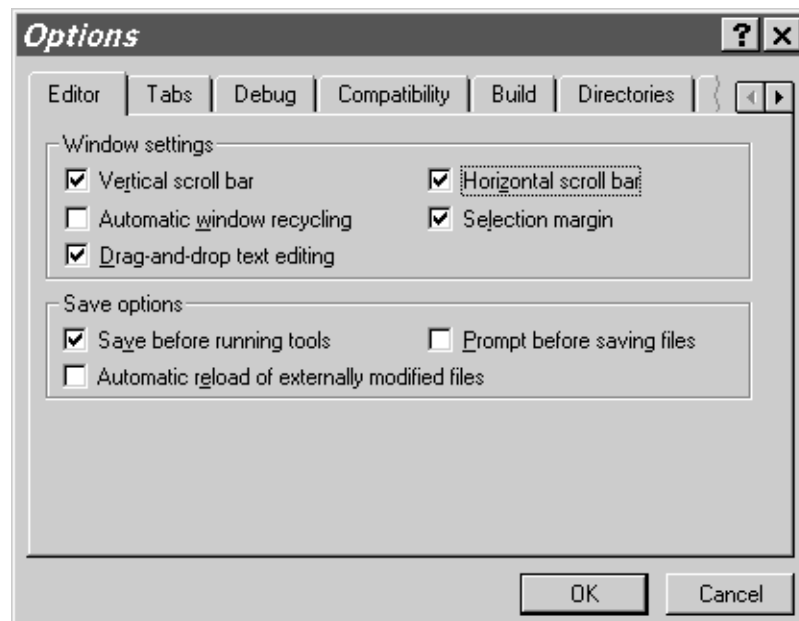
- A breakpoint has been set on the line of code with the call to printf. The red “stop sign” is to the left of the code. The call to printf has been executed with the Step command.
- The current line to execute has a yellow arrow to the left of it. It contains a call to the getchar function. That line of code has not yet been executed.
- szGreeting can be seen in the Locals window and in the Watch window. You can scroll these windows horizontally so you can see the entire line. You can see the beginning of the value of szGreeting in the Watch window in the example.
- A plus sign next to a variable in the Locals and Watch windows indicates the data may be expanded. This is useful for inspecting the values of complex data structures.

Customizing the Environment

You can modify the Visual C++ debugging environment. You can customize the editor, colors, fonts, debugger, default directories, workspace, and help.

► To customize the environment

1. From the Tools menu, select Options.



2. On Options, customize your environment as desired.

Refer to the Visual C++ online helps for more information.

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?

The cause of an unhandled exception is a failure to handle memory correctly. 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 the problem resides in C code, you can find the problem by tracing into the code.

If the problem exists in the OneWorld-generated code, finding the problem may be more difficult. The debugger provides error messages that are of help. The most common problem has to do with missing objects. If there is a business function that is being called that does not exist, the tool issues an error message that identifies the missing function. For example, "Business function load failed - CALLBSFN.DLL." In this case, you can either rebuild the business function or check it out and build it using BusBuild to correct this error.

Remember that ALL business functions are built into larger DLLs. The most generic of these is CALLBSFN.DLL. Most application specific DLLs are not in CALLBSFN. For example, J.D. Edwards financial business functions use CFIN.DLL.

The object (BSN func) may need to be checked out to the workstation again and built through BusBuild again in CALLBSFN.DLL (or the specific DLL).

If other objects are missing, termination will be 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, it immediately causes an unhandled exception in the program.

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

Make sure that the program is terminating in the same place. If the program is failing to free memory after its use, the program may eventually have insufficient memory to run. If so, you must reboot the workstation to free memory.

Is the Application Encountering Errors?

When a business function is called from event rules, its processing is unknown to the user. This is both a benefit and a hindrance.

- It is a benefit because it should not be a concern to the user how the function works, only that it does work.
- The disadvantage is that it is not uncommon to call a function with more than 50 parameters, any one of which might cause errors.

There are two solutions to this sort of issue:

- Review the function specifications to make sure that you have hooked it up correctly.
- Step into the code to see what is going wrong.

Is the Output of the Program Incorrect?

If the output of the program is incorrect it means that there is a logic flaw in the code. Trace into the code to find the issue.

Where Could the Problem Be Coming From?

Spend some time thinking about where the source of the problem might be.

For example, consider the “Null pointer error.” Somewhere in the code memory is a memory leak. To help you determine where the leak is, you can toggle between the grid rows in order to force the execution of the *Row is Exited* event. If the problem occurs, you have narrowed your search for the leak, because you know that it is in the grid processing.

Is the Function Being Called as Expected?

You can set a breakpoint at the beginning of a called function to ensure that it is being called and that the input is what you expected.

Set the break point at the beginning of the business function or at the beginning of an event like *Grid Row is Fetched*.

- Step through every code path.

Use the debugger to test the code. Both the J.D. Edwards Debugger and the Microsoft Visual C++ Debugger provide you with the ability to modify values as the code is running. Use this ability to trace through *every* line of code. This is also a powerful way to test everything that could happen to your code, not just what should happen to it. This form of testing will help demonstrate whether you have handled errors correctly.

For example, step through an If branch and input bad data to see if errors occur as expected.

- Find the cause of the problem, not the symptom.
- Look for problems with object transfer/reinstall.

Debug Logs

There are several sections of the jde.ini file that relate to debugging. The first statement that must be checked is in the [JDE_CG] section. The line Target = Release should be modified to read Target = Debug. An application install of OneWorld will deliver the jde.ini file with Target = Release. To debug business functions, the jde.ini files must be changed to Target = Debug, and the business functions that you wish to debug must be rebuilt.

```
[JDE_CG]
STDLIBDIR=c:\MSDEV\LIB
TPLNAME=EXEFORM2
ERRNAME=CGERR
TARGET=Debug
INCLUDES=c:\MSDEV\INCLUDE;$(SYSTEM)\INCLUDE;$(SYSTEM)\INCLUDEV;
$(SYSTEM)\CG;$(APP)\INCLUDE;
LIBS=c:\MSDEV\LIB;$(SYSTEM)\LIB32;$(SYSTEM)\LIBV32;$(APP)\LIB32;
MAKEDIR=c:\MSDEV\BIN
USER=DEMO
```

You can output to 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. This is a very useful debugging tool when you have narrowed a problem down to a specific issue involving the JDEDDB APIs.

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

The memory frequency setting allows you to validate the memory heap at a particular 1000th location. You can set breakpoints and go into the code.

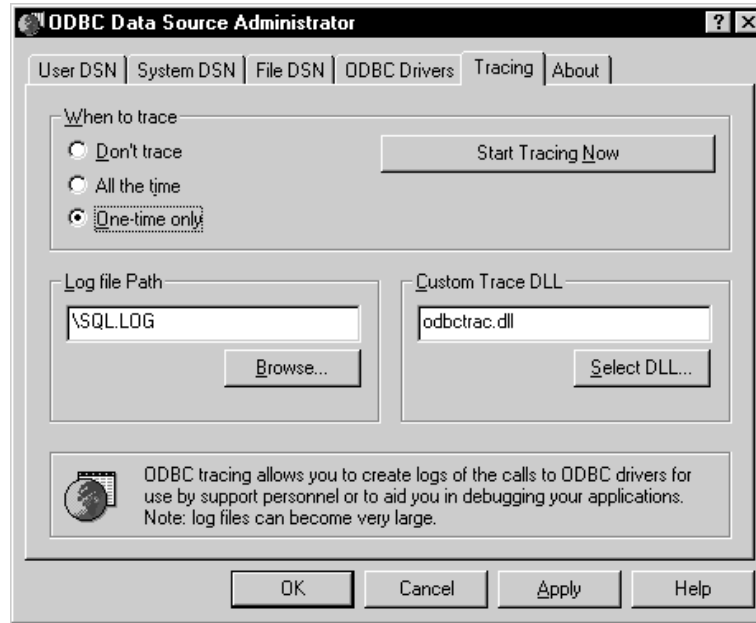
To view logs right click anywhere in the window and choose Log Viewer to view log files. For more information about other logs you can view for troubleshooting, refer to the appendices in the *Server and Workstation Administration Guide*.

SQL Log Tracing

You can use SQL Log Tracing to help you determine the exact SQL statement that is generated to the database.

To turn on SQL Log tracing

1. From the Control Panel on your workstation, choose the 32 bit ODBC driver.
2. Click the Tracing tab.
3. Choose one of the following options:
 - All the Time
 - One Time Only
4. Specify the log output path in the Log file Path.



Debug Tracing

Use debug tracing to trace database and runtime events, business functions, and system functions. You can turn on debug tracing and output the results in a log file. This debug log is also useful for checking to see how the tool constructs a SQL statement.

Turning on Debug Tracing

► To turn on debug tracing

1. In the `jde.ini` under `[DEBUG]`, change `Output=NONE` to one of the following values.

<code>Output=FILE</code>	This prints both database tracing and runtime tracing.
--------------------------	--

<code>Output=EXCFILE</code>	This prints runtime tracing only.
-----------------------------	-----------------------------------

2. Change `Level=` to suit specific debugging needs. Possible values for `Level` are contained in the comment line following the `Level=` line. Any combination is acceptable. Commas should separate values. The possible values for `Level` are:

<code>EVENTS</code>	This prints when an event is started and when it finishes.
---------------------	--

<code>BSFN</code>	This prints when business functions are entered and when they return.
-------------------	---

<code>SF_?</code>	This prints when system functions execute. In place of the <code>?</code> you can designate a specific type of system function, for example control or messaging.
-------------------	---

System Function Tracing

System functions are grouped by category (same categories as design time). Where reasonable, information has been provided about what the system function is acting upon. For example, in many of the grid system functions the row number is printed. For most control system functions the alias of the control and the control ID is printed.

The following example shows the jdedebug.log output running Journal Entry with the following jde.ini options

```
Output=EXCFILE
```

```
Level=EVENTS,BSFN,SF_GRID,SF_CONTROL
```

```

RT: >>>Beginning ER: Dialog is Initialized App: P0911 Form: W0911I
RT: <<<Finished ER: Dialog is Initialized App: P0911 Form: W0911I
RT: >>>Beginning ER: Post Dialog is Initialized App: P0911 Form: W0911I
RT: <<<Finished ER: Post Dialog is Initialized App: P0911 Form: W0911I
RT: >>>Beginning ER: Add Button Clicked App: P0911 Form: W0911I
RT: >>>Beginning ER: Dialog is Initialized App: P0911 Form: W0911A
RT: SYSFN: Hide Control <> 0
RT: SYSFN: Disable Control <ICU> 5258
RT: SYSFN: Hide Grid Column COL: 5
RT: SYSFN: Hide Control <ATDOW> 5392
RT: SYSFN: Hide Control <REMA> 5405
RT: SYSFN: Hide Control <> 5295
RT: SYSFN: Hide Control <> 5385
RT: SYSFN: Hide Control <DOC> 5297
RT: SYSFN: Hide Control <KCO> 5299
RT: SYSFN: Hide Grid Column COL: 7
RT: SYSFN: Hide Grid Column COL: 8
RT: SYSFN: Hide Grid Column COL: 9
RT: SYSFN: Hide Grid Column COL: 11
RT: BSFN: Calling : BatchOpenOnInitialization App: P0911 Form:
W0911A
RT: BSFN: Returned 0: BatchOpenOnInitialization App: P0911 Form:
W0911A
RT: BSFN: Calling : GetAuditInfo App: P0911 Form: W0911A
RT: BSFN: Returned 0: GetAuditInfo App: P0911 Form: W0911A
RT: <<<Finished ER: Dialog is Initialized App: P0911 Form: W0911A
RT: >>>Beginning ER: Clear Screen Before Add App: P0911 Form: W0911A
RT: SYSFN: Enable Control <PCTOW> 5390
RT: SYSFN: Hide Control <ATDOW> 5392
RT: SYSFN: Hide Grid Column COL: 5
RT: SYSFN: Hide Control <REMA> 5405
RT: SYSFN: Enable Control <CRCD> 5273
RT: SYSFN: Enable Control <LT> 5292
RT: SYSFN: Enable Control <LT> 5351
RT: SYSFN: Show Grid Column COL: 6
RT: SYSFN: Hide Grid Column COL: 12
RT: SYSFN: Show Control <LT> 5292
RT: SYSFN: Show Control <CRDC> 5271
RT: SYSFN: Hide Control <LT> 5351
RT: SYSFN: Hide Control <CCD0> 5358
RT: BSFN: Calling : GetLocalComputerId App: P0911 Form: W0911A
RT: BSFN: Returned 0: GetLocalComputerId App: P0911 Form: W0911A
RT: <<<Finished ER: Clear Screen Before Add App: P0911 Form: W0911A
RT: >>>Beginning ER: Post Dialog is Initialized App: P0911 Form: W0911A
RT: <<<Finished ER: Post Dialog is Initialized App: P0911 Form: W0911A
RT: >>>Beginning ER: Add Last Entry Row to Grid App: P0911 Form:
W0911A
RT: <<<Finished ER: Add Last Entry Row to Grid App: P0911 Form:
W0911A

```


Web Applications



Developing Web Applications

The OneWorld Development Tools make it easy for you to develop and maintain applications for use on both Windows clients and Web clients, even if the clients require slight differences in the design of the forms. You can create Windows, Java, or HTML applications. As you design a form, Form Design Aid allows you to view it in one of three modes. Your design team should decide which modes to use for which client types. For example, you might use Mode 1 for forms to be generated in Windows, Mode 2 for forms you want to generate in Java, and Mode 3 for forms you want to generate in HTML.

Most of the controls you put on a form are appropriate for any of the modes, but you can choose to show or hide (enable/disable) any control on the form for each of these three modes. In this way, you can develop one set of forms from which you generate three versions of the application (one for each mode). If you customize any of the OneWorld Web applications, or if you develop new ones, you must use the Web Generation Facility to generate them into Java or HTML applications. Refer to *Working with Controls* for more information about using different modes and customizing the controls on forms for different modes.

You typically create an application in Mode 1, the default mode. This application is automatically enabled for Windows. You can then use Mode 2 or Mode 3 to modify your Mode 1 application for Java or HTML.

Although you can develop or change Web-based applications (Java or HTML), on your workstation, you need a workstation that is web enabled to test the application. All OneWorld clients are web enabled. The following components are required for your client:

- Internet Explorer 4.01 or above
- OneWorld Web Generation Facility (Installed with OneWorld client)
- Connectivity to a OneWorld Web server

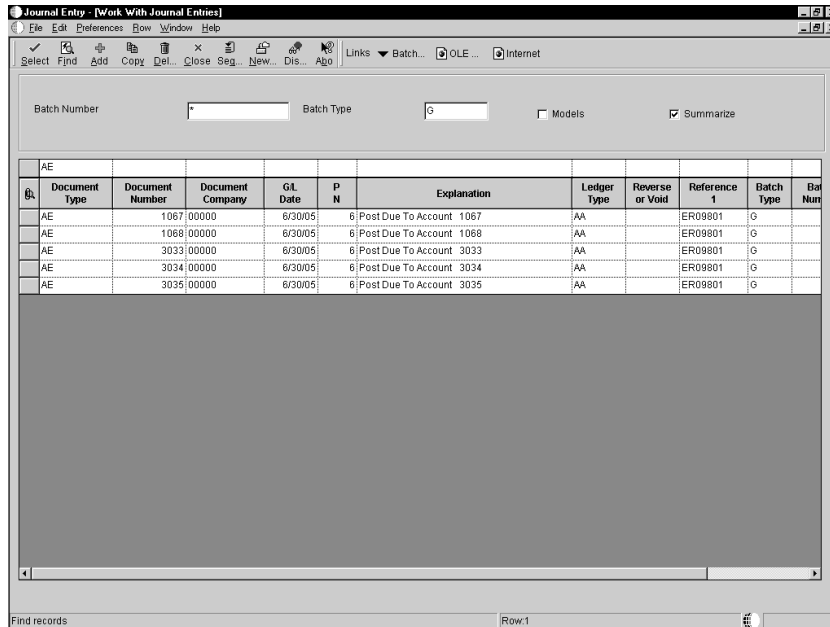
If you customize any of the J.D. Edwards web applications you must regenerate them to create the specified Java or HTML applications. Or, if you create new applications, you must generate those to create Java or HTML applications.

In the examples below, you can see the kinds of changes you might want to make in a windows application to make it easier for use on the Web. Typically you use fewer fields in a Web application to make it easier for an end user to view and use.

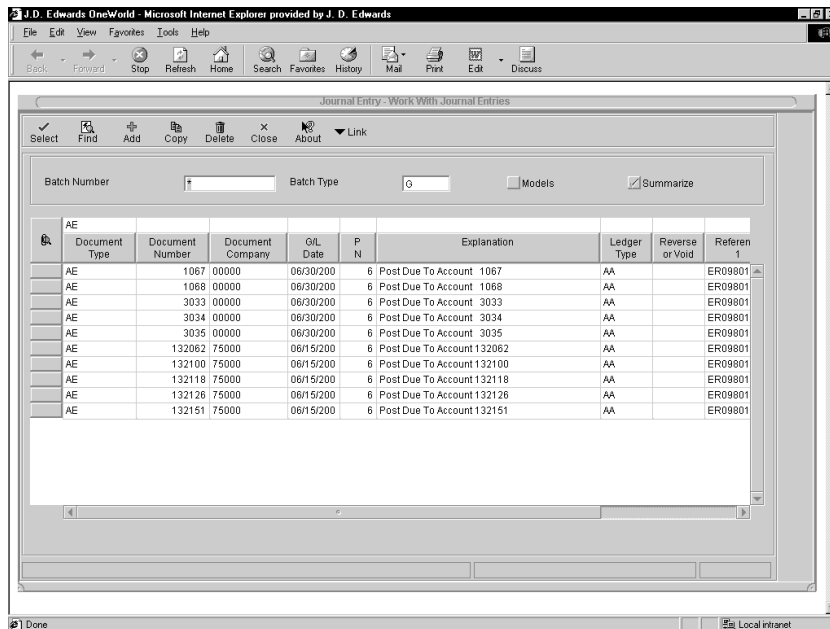


Note: Changing the font size in your style sheets can alter the appearance of the form, including displacing controls, and wrapping or hiding control text.

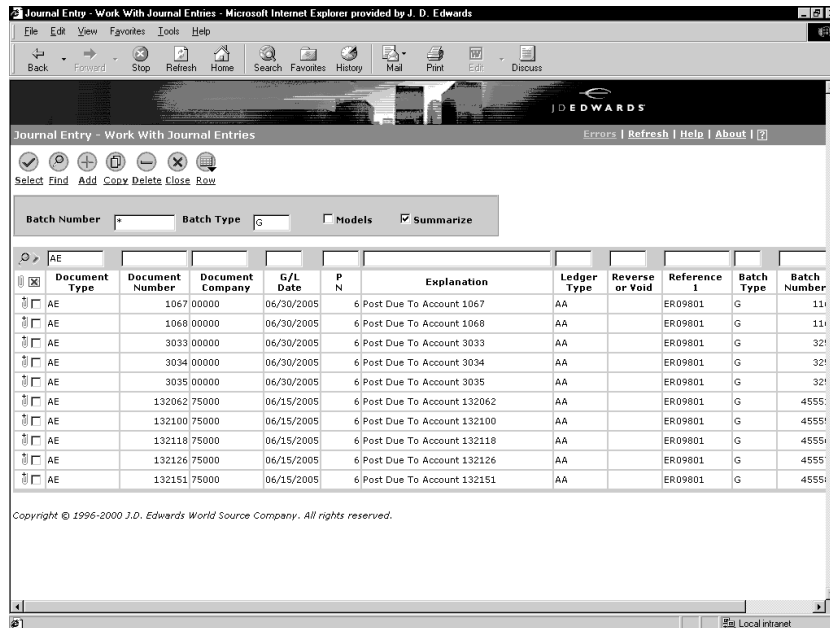
Here is the Windows version (Mode 1) of an application:



You can modify your Mode 1 application and use another mode, for example Mode 2, to create an application for Java:



You can use another mode, for example Mode 3, to further modify your Mode 1 application for an HTML application:



After you create an application in form design you can use Menu Design to attach the application to a menu and designate which mode the application should run in. You should specify the mode on the menu so that the user can tell which application to use. See *Menu Design* for more information about designing menus.

To test a Java application you must:

- Save your application in Form Design
- Generate the application to the Web server
- Attach the application to a menu and designate which mode it should run in
- Execute the application

To create web applications, use OneWorld Development Tools in the same manner as you would for standard applications.

If you use OneWorld Development Tools to create your own applications, and these applications are designed for use as web applications, you should consider the following in order to ensure optimal performance:

- Each data and business function request executes on a data or enterprise server; not the web server. As a result, your web client performance is limited by the speed of your network or modem connection as well as network traffic bottlenecks. If possible, you should limit or bundle data and business function requests in a manner most efficient for WAN and Internet connections. If you have several table I/O calls serially linked (for example, each one is dependent on the previous call), consider using

named event rules to combine all calls, which are compiled and executed on the enterprise server.

- Whenever possible, design the web-ready application to perform logic using event rules. This provides better performance than if you use C language business functions or event rules business functions (named event rules). Remember, whenever you call a business function the logic is performed on the remote OneWorld enterprise server. However, as mentioned above, if the event rules contain several table I/O or business function calls serially linked, these will perform better as a business function (named event rule) on the enterprise server.
- Design the web-ready application to get data in manageable chunks. If your application performs a business function request before and after each grid line, your performance will suffer. Also, you should try to avoid using business functions as a means to request data; use business views or table I/O instead.
- Educate users in the importance of using search criteria in query by example columns. This is important given the constraints of transferring data at modem speeds across typical Internet connections. For example, users should routinely try to narrow their searches, because data retrievals are configured to only obtain 10 records at a time. If you develop custom applications for web use, consider defining query by example fields as required entries.

Application P98305W is a version list application created specifically for the web client. You can use this application to look up web versions of applications.

Developing web applications contains the following topics:

- Understanding the HTML client
- Generating web applications

See Also

- Designing Forms Using Multiple Modes*

Understanding the HTML Client

The HTML client automatically posts (refreshes) on the following critical events:

- Form startup
- Set focus on grid (only when ER exists)
- Checkbox (only when ER exists)
- Radio button (only when ER exists)
- Bitmap clicked
- URL clicked
- Tab is selected
- Node expanded in tree control
- Drag-and-drop action occurs in tree control

In addition to the critical events listed above, the client also refreshes the page when non-critical events occur that contain hide/show or enable/disable grid functions. The following is a list of these non-critical events:

For Edit controls	Control is exited
	Control is exited and changed-inline
	Control is exited and changed-asynchronous
For Grid controls	Column is exited
	Column is exited and changed-inline
	Column is exited and changed-asynchronous
For Parent/Child form grid	Tree node level is changed
For Tree control	Tree node is selected

These post rules for critical and non-critical events form the default post model used by the HTML client. The HTML post model can cause usability issues for

high volume data entry applications and inquiry applications used frequently on the web. The issue occurs when an application does hide/show or enable/disable grid on non-critical events forcing the HTML page to post. The refresh makes the screen flash. The duration of the flash depends on network traffic, hardware, configuration, and so forth. Consequently, you might want to override the HTML post model in some situations.

Use the HTML Auto Refresh option in a form's property settings to turn off the automatic post on all non-critical events for that form (the default is to have the attribute turned on). Use the HTMLPOST option in an event rule to turn on the post for a single non-critical event (the default is to have the attribute turned off). Enabling posting overrides a disable. For example, if you turn off posting for a form, but turn it on in the ER for a single event, posting is suppressed for all events on the form except for the one with the HTMLPOST option turned on in its ER.

The Windows and Java clients ignore both the form and event attributes.

Generating Web Applications

The Java & HTML Generator allows you to easily and intuitively generate a OneWorld application in either Java or HTML, or both. The look and feel of the tool is very similar to that of the other OneWorld Development Tools.

After you generate a OneWorld form or application in Java or HTML, the tool stores the output in an appropriate database on the Web server. You select the database by specifying the environment. The output of the generation is a Java or HTML object, which is stored as persistent data in the database. The system retrieves the objects at runtime from the database. You can use the Web Generation tool to generate menus, data dictionary items, forms, tables, business views, named event rules, and reports.

Generate a form when any of the following conditions occur:

- You change or add an existing Windows mode form
- You enable or disable Java or HTML modes on an existing form
- You create a new form

Generate all objects when either of the following conditions occur:

- You initially install OneWorld
- You upgrade OneWorld

Generate other objects when the following condition occurs:

- You modify or add objects such as menus, tables, reports, and so on

Note: Users without administrative rights are restricted to generating forms and reports privately. Administrators can generate all objects publicly.

Generating web applications is composed of the following topics:

- Logging in
- Setting generator options
- Generating forms
- Generating reports
- Generating other objects

- Generating all objects

See Also

- Understanding Workstation jde.ini Settings* in the *OneWorld System Administration* guide for information on gaining administrative rights to the Java & HTML Generator

Logging in

Before you can use the generation tool, you must log in to the web server to which you want to generate objects.

► To log in

1. Launch the Java & HTML Generator application.

The method you use to launch the Java & HTML Generator varies based on your environment and set up. See the J.D. Edwards Knowledge Garden for the most recent set up information.

2. On Java & HTML Generator, click the Admin tab, and then enter the web server you want to use in the Server Name field.
3. Click OK.

Setting Generator Options

Before generating objects, you can define a variety of parameters that will affect how the Java & HTML Generator functions, including defining the TAM path, error and ER log locations, protocol, and language.

Note: Some options are available only to users with administrative privileges.

The following table describes the parameters you can configure for the Java & HTML Generator.

Function	Description
Define remote TAM path	Enter a TAM path in the <i>Remote TAM Path</i> field on the Admin tab to use a remote TAM for generation.
Define protocol	Click the HTTP or the HTTPS option on the Admin tab to generate web applications with one protocol or the other.

Generate error log	To generate an error log when the Java & HTML Generator runs, click Error Log on the Admin tab, and then indicate where you want the file to be output.
Generate ER log	To generate an ER log when the Java & HTML Generator runs, click ER Log on the Admin tab, and then indicate where you want the file to be output.
Generate objects publicly or privately	A user with administrative rights may choose to generate objects privately or publicly. Click the Personal or Private option on the Admin tab to generate web applications one way or the other.
Choose output language	To generate objects in multiple languages, click the languages you want on the Language tab. For each non-default language you choose, you must indicate a TAM path.

See Also

- Understanding Workstation jde.ini Settings in the OneWorld System Administration* guide for information on gaining administrative rights to the Java & HTML Generator

Generating Forms

If you changed or added an existing Windows mode application, if you added a new form, or if you enabled or disabled modes, you must generate the form before you can use the web version of the form. You can generate all the forms at once, or you can generate the forms for an application or system code. The Java & HTML Generator always generates Java for the modes you select. Additionally, you can choose to generate HTML for the selected modes as well.

Complete the following tasks:

- Generate selected forms
- Generate all forms

To generate selected forms

1. On Java & HTML Generator, click the Form tab.
2. To generate the forms in an application, click Application, and then complete the following fields:

- Object Name

Enter the system name of the application.

- Form

To generate a specific form only, enter the system name of the form. If you leave this field blank, the system generates all the forms in the application.

- Version

To generate a specific version of a form, enter the system name of the version. If you leave this field blank, the system generates all versions of the form or forms.

Go to step 4.

3. To generate the forms in a system code, click System Code, and then complete the following field:

- Object Name

Enter the system code.

Go to step 4.

4. Click Generate Business Views to generate the business views associated with the object you specified in step 2 or 3.
5. Click the modes (1, 2, or 3) for which you want to generate forms.

You can generate forms in one or more modes.

6. To generate forms in HTML, click *Generate HTML version*.

When you select this option, the system generates HTML versions of the forms for each selected mode.

7. If you are generating HTML versions of the forms, you can put a footer on the forms by entering a task ID in the Footer Menu field.

If you enter a task ID, the ID will be generated as a hyperlink, allowing a user to launch a OneWorld task. For example, if you entered the task ID for the Journal Entry application (2/G0911), a user could launch Journal Entry by clicking on the form's footer.

Leave this field blank if you do not want footers to appear on the forms.

8. Click OK.

▶ To generate all forms

1. On Java & HTML Generator, click the Form tab.
2. Click Generate Business Views to generate the business views associated with the forms.
3. Click the modes (1, 2, or 3) for which you want to generate forms.

You can generate forms in one or more modes.

4. To generate forms in HTML, click *Generate HTML version*.

When you select this option, the system generates HTML versions of the forms for each selected mode.

5. If you are generating HTML versions of the forms, you can put a footer on the forms by entering text in the Footer Menu field.

If you enter a task ID, the ID will be generated as a hyperlink, allowing a user to launch a OneWorld task. For example, if you entered the task ID for the Journal Entry application (2/G0911), a user could launch Journal Entry by clicking on the form's footer.

6. Click Generate All.

Generating Reports

If you added or changed reports (batch applications or batch versions), you must generate the report to make the report available on the web. Batch applications are available in Java format only.

You can generate all of the reports and their versions in the system, or you can generate selected reports based on report name or system code.

Complete the following tasks

- Generate selected reports
- Generate all reports

▶ To generate selected reports

1. On Java & HTML Generator, click the Reports tab.
2. To generate a specific report or batch version, click Report, complete the following fields, and then click OK:
 - Object Name

Enter the system name of the report.

- Version

To generate a specific version of a report, enter the system name of the version. If you leave this field blank, the system generates all versions of the report.

3. To generate the reports and their versions in a system code, click System Code, complete the following field, and then click OK:

- Object Name

Enter the system code.

To generate all reports

1. On Java & HTML Generator, click the Reports tab.
Ensure all the fields on the Reports tab are blank.
2. Click Generate All.

Generating All Objects

Generate all objects only at initial OneWorld installation, when you have extensive changes or additions to all applications, or if you are installing an upgrade. This process takes several hours.

To generate all objects

1. On Java & HTML Generator, click the Admin tab.
2. Click Generate All.

Generating Other Objects

If you have administrative rights to the Java & HTML Generator, you can generate other objects in addition to forms and reports such as menus, data dictionary items, and tables. Occasional changes to such objects in OneWorld might impact how OneWorld applications function without affecting their appearance. In such cases, you can save time by generating only those objects that have changed without having to generate an application and all of its supporting objects.

Complete the following tasks:

- Generate menus
- Generate data dictionary information
- Generate business views and tables
- Generate data structures
- Generate named event rules (NERs)

See Also

- Understanding Workstation jde.ini Settings in the OneWorld System Administration* guide for information on gaining administrative rights to the Java & HTML Generator

▶ **To generate menus**

1. On Java & HTML Generator, click the Menu tab.
2. Click Generate All.

▶ **To generate data dictionary information**

1. On Java & HTML Generator, click the DD Info tab.
2. Click one of the following options, depending on the type of object you want to generate:
 - Data Dictionary
 - Table ID
 - Business View

Skip this step if you want to generate all data dictionary items.

3. Enter the name of the object you want to generate in Object Name.
 Leave this field blank if you want to generate all data dictionary items.
4. Click OK to generate a single object. Click Generate All to generate all data dictionary items and their related table ID and business view references.

▶ **To generate business views and tables**

1. On Java & HTML Generator, click the BSVW & Table tab.
2. Click one of the following options, depending on the type of object you want to generate:

- Business View
- Table

Skip this step if you want to generate all business views and tables.

3. Enter the name of the object you want to generate in Object Name.

Leave this field blank if you want to generate all business views and tables.

4. Click Generate Data Dictionary to generate the data dictionary items associated with the object or objects to be generated.
5. Click OK to generate a single object. Click Generate All to generate all business views and tables.

▶ **To generate data structures**

1. On Java & HTML Generator, click the BSFN & Data Structure tab.
2. Click one of the following options, depending on the type of object you want to generate:
 - BSFN's data structure
 - UBE's data structure
 - BSFN view for storefront
3. Enter the name of the object you want to generate in Object Name.
4. Click OK.

▶ **To generate named event rules (NERs)**

1. On Java & HTML Generator, click the NER tab.
2. Enter the name of the object you want to generate in Object Name.

Leave this field blank if you want to generate all NERs.
3. Click Dependencies to generate the dependencies associated with the object or objects to be generated.
4. Click OK to generate a single object. Click Generate All to generate all NERs.

Performance



Performance

This section includes tips and tools to help you enhance the performance of your applications.

About performance contains the following topic:

- Performance Tips



Performance Tips

This chapter describes some things you can look for to improve performance. In particular it provides information about analyzing and improving the performance of:

- Data Dictionary
- Table Design
- Business Views
- Data Structures
- Data Selection and Sequencing
- Form Design
- Batch Applications
- Event Rules
- Business Functions
- Error Messaging
- Transaction Processing

Things to Look For

Does the application seem slow on all platforms? If yes, then further analyze the application.

Does the application seem slow on a specific platform, but not all platforms? If yes, then there may be a logical issue or index issue for the given platform.

There are several tools to perform this analysis. Use `jddebug.log` first to determine at what event the application seems slow and what is happening on that event.

Analyze database and business function use. In analyzing the tables, consider the following:

- Records should be read in only once unless all secondary fetches for a given record were cached.
- The total of all JDB_OpenTable and JDB_OpenTableFromCache should match the total number of JDB_CloseTable calls for each table.
- The total number of actual JDB_OpenTable calls (the amount of opens that are not cached) should be as close to one as possible. Values larger than one indicate that multiple database cursors are being retained for the table.

Next turn on table logging for the jdedebug.log. Review the database I/O. If the table I/O is still a concern, start the SQL log and analyze the information in the log.

If you have justified all table I/O, and the application is still slow on certain events, then review the business functions running on those events. Make sure the business functions are running efficiently.

Look for business functions accessing a table a large number of times for the number of records being processed. Can caching be used?

Look for values that can be saved in global event rule variables instead of performing redundant I/O calls.

Look for multiple calls to multiple business functions running over the same table that can be consolidated into one table I/O call.

Look for business function or table I/O calls that can be moved to different events in order to reduce the number of times that they are called.

Look for business functions calling other business functions unnecessarily.

Data Dictionary Performance

Triggers

Data dictionary triggers can be used for both formatting and validation. Triggers are the most costly way to format or edit data values. Most of the formatting that occurs in OneWorld involves date and math fields. These fields require the most overhead for editing and formatting. Make sure that all triggers are coded as efficiently as possible.

Overrides

You can use data dictionary overrides to override the data dictionary characteristics of a form control, grid column, or report field at design time. This causes additional overhead at application startup because of the extra processing required to evaluate and apply the overrides. The overrides that are

applied can have either a positive or negative affect on performance during runtime processing. For example if you use the override feature to disable some of the validation functionality, then performance will be increased, because disabling features causes some runtime overhead to be removed. Adding overrides such as edit/format triggers or next numbering will increase runtime overhead.

Validation

Validation is based on the data dictionary item that is attached to a control, column, or field. The overrides can also play a part in validation. Validation is the most costly for data items that have triggers. Validation of user defined codes also requires I/O to validate data values. General validation can be more costly for certain data types such as Math and Date. These data types require special logic to manipulate the OneWorld-specific data types.

Table Design Performance

Be careful using SELECT statements. Try to use existing indices for a table. It is better to use additional keys than to create a new index. Additional indices create overhead.

Use a partial key only for sequential and fetch next or just to look at a few more records.

If you open a table and then a fetch key, it destroys the pointer so do a fetch single instead. Almost any other database API will destroy the pointer also.

Opening a table the first time is a big performance hit.

The fetch matching key uses a greater than or equal key so it selects more than you may need. Use the correct JDB API for what you need.

Index Considerations

Adding a suitable index will almost always improve performance when selecting and fetching data from the database. However, each additional index adds maintenance overhead when records are added, updated or deleted in the database. The decision about adding a new database index over a table should balance these two factors.

OneWorld accesses databases efficiently. Use ordinary database design considerations, including normalization considerations, when determining the optimal design of database tables and indices.

Join fields should be the keys in two tables.

Limits on Row Set Size

The lowest common denominator for row size is the specification given in the SQLSERVER database.

SQL Server 6.5 can have as many as 2 billion tables per database and 250 columns per table. The maximum number of bytes per row is 1962. If you create tables with *varchar* or *varbinary* columns whose total defined width exceeds 1962 bytes, the table is created but a warning message appears.

Inserting more than 1962 bytes into such a row or updating a row so that its total row size exceeds 1962 produces an error message and the statement fails.

Considerations for Coexisting with World Software

Make sure tables in OneWorld match AS/400 tables. If the data is not structured the same it causes problems.

If you plan custom modifications in a coexistence environment, the World RPG programs must be able to read the structure of any tables to be read by World Software. This means that the tables World Software must access should be initially created in World database structure, not from OneWorld using AS/400 foreign databases (Access, Oracle, SQL Server). The World environment must be set up first before OneWorld is set up on top of it.

Make date and time conversions on AS400 and OneWorld the same.

Refer to the *Coexistence Guide* for more information about coexistence.

Index Limitations for Various Databases

Access32

The following index limitations apply to Access 32:

- The maximum number of indices per table is 32.
- The maximum number of fields in an index is 10.
- The maximum number of fields in a records is 255.

SQLSERVER

The following index limitations apply to SQLSERVER:

- The maximum number of clustered indices per table is one.
- The maximum number of nonclustered indices per table is 249.
- The maximum number of columns in a composite index is 16.

DB2 for OS/400

The following index limitations apply to DB2 for OS/400:

- The maximum number of identifiers for an index name is 10.
- The maximum size of an index is 1 terabyte.
- The maximum length of an index key is 2,000.
- The maximum number of columns per index key is 120.
- The maximum number of indexes per table is approximately 4,000.

This is not permitted, because the key length in this case is 1101 (>900).

Oracle

The following index limitations apply to Oracle:

- The maximum length for an index is 254 bytes, less the number of keys that allow NULL values.
- The maximum number of columns per index is 16 or a maximum key length of 900.

Example:

In the F32944 table, the index is defined as:

```
{
  MATH_NUMERIC ktkit;          /* 0 to 48 */
  char ktmc01[251];           /* 49 to 299 */
  char ktmc02[[252];         /* 300 to 550 */
  char ktmc03[251];          /* 551 to 801 */
  char ktmc04[251];          /* 802 to 1052 */
  MATH_NUMERIC ktseqn;       /* 1053 TO 1101 */
} KEY1_F32944, FAR *LPKEY1_F32944;

#define ID_F32944_KIT_CONFIGURED_STRING_ID_B 2L
```

Key Column Violation

When you define indexes on various columns, ensure that no two indices, one of which may be a primary unique index, are defined on the same column.

Example:

In the F03B08 table, the indexes were defined as:

```

#define ID_F03B08_COMPANY_FISCAL_YEAR 1L /* PRIMARY &
UNIQUE */

typedef struct
{
    char rdco[6];                /* 0 to 5 (ASC)*/
    MATH_NUMERIC rdctry;        /* 6 to 54 (DESC) */
    MATH_NUMERIC rdfy;
/* 55 to 103 (DESC) */
    MATH_NUMERIC rdpn;
/* 104 to 152 (DESC) */
} KEY1_F03B08, FAR *LPKEY1_F03B08;

#define ID_F03B08_COMPANY_FISCAL_YEAR_(ASCEND) 2L /*
NONUNIQUE */

typedef struct
{
    char rdco[6];                /* 153 to 158 (ASC) */
    MATH_NUMERIC rdctry;        /* 159 to 207 (ASC) */
    MATH_NUMERIC rdfy;
/* 208 to 256 (ASC) */
    MATH_NUMERIC rdpn;
/* 257 to 305 (ASC) */
} KEY2_F03B08, FAR *LPKEY2_F03B08;

```

This is not permitted, because two indexes are defined on the same columns.

Specification File Corruption

If you encounter specification file corruption, recreate the tables in the source database.

Table I/O Objects

Table I/O header size is 206 bytes.

Table I/O mapped item size is 181 bytes.

Maximum number of table I/O mapped items size to fit in 32K is $(32768-206)/181$ for about 180 items.

In order to leave space for literal values, use a mapping size not greater than 130 elements. The 130 elements relate to the number of mappings on any table I/O statement. A table may have more than 180 columns, but you can still map fewer than 180 of them without problems. If you need to use table I/O with more than 180 mappings, you should probably create several table I/O statements for specific requirements.

Business View Performance

Should I use a single-table or multiple-table business view?

In many instances, the application needs to access data from multiple database tables to perform a business operation. You can accomplish this in two ways:

- Use a multiple-table business view to access all the related data fields
- Use a single-table business view to access the data fields from the primary table, and use a business function or table I/O to access the data fields from secondary tables

The best choice is not always obvious, but you can usually decide by looking at the resulting number of database I/O operations that will be performed. A joined business view that uses cross data source joins causes slower performance.

If the secondary tables are “master” files, the second option is usually preferable because it makes best use of database caching. For example, suppose the primary table contains a company number and the related company name is stored in a secondary Company Master file. If, in practice, it is likely that the same Company Master record will be retrieved for several records in the primary table, then it is usually preferable to fetch the company name explicitly using a business function or table I/O.

OneWorld is particularly optimized to fetch user defined codes. User defined code tables should never be included in a multiple-table business view.

Should I restrict the number of fields in a business view?

You may need to choose between using an existing business view that may contain fields that are not needed or creating a new business view with only the desired fields. While any extraneous fields cause additional work on both the server and the workstation, this is usually *not* a significant performance concern. In cases involving tables that will be updated, OneWorld can automatically process all fields in a table, even if they are not included in the business view.

Minimizing the number of fields in a data structure is far more productive in improving runtime performance than minimizing the number of fields in a business view. You can create new business views that meet your specific needs that do not affect performance.

Unions

Unions are the most expensive database operation out of all the SQL SELECT statement types. In general you should avoid using them. An example of when you might want to use unions is if you are using the same part of tables at

different times. For example, an application uses the same part of 10 different tables, and you only need that part one part at a time. Post Processes is an example of this.

Joined Business View Versus Table I/O

Use a joined business view if you are reading and updating two or more major tables. Use table I/O if the table being read or updated is just a side affect of another action.

Consider writing applications with the minimum number of grid columns required for the application's basic functionality. You can add columns to the associated business view to supplement the basic application with minimal effort. Therefore, it is better to add columns to a business view than to add columns to a grid.

Data Structure Performance

Should I restrict the number of fields in a data structure?

Performance measurements indicate that you should attempt to minimize the number of fields in a data structure, particularly when Configurable Network Computing (CNC) capabilities require that the data structure be passed between workstation and server.

Data Structure Objects

Data structure header size is 237 bytes.

Data structure item size is 72 bytes.

Maximum number of data structure items to fit in 32K is $(32768-237)/72$ for 450 elements.

In order to leave space for literal values you should limit your structure size to 350 elements or less. Any data structure approaching a size of 100 elements or more should be considered carefully.

Data Selection and Sequencing

Use the following two system functions in the *Initialize Section* event to conditionally change the data selection for that section:

- Set User Selection
- Set Selection Append Flag

The Initialize Section event is normally used in the first level-one section of a report to conditionally select data based on values passed through the report data structure.

For example, Bill of Material Inquiry calls the Bill of Material report passing the Parent Item, Parent Branch, Type of Bill, and Batch Quantity. In the Initialize Section Event of the first level-one section of the Bill of Material report, the report data structure values are checked. If they are not equal to blank, the Set User Selection system function is used to add these selections.

Form Design

This list contains standards for increased performance across all form types.

- Limit the number of columns in the grid to the minimum required by the application.
- Keep the number of columns in the business view to the minimum required by the application.
- Limit the number of form controls, whether hidden or visible, to the minimum required by the application.
- Event rule variables should be used as work fields in place of hidden form controls.
- Disable data dictionary functions on form and grid controls that are not required, such as edits and default values, whether hidden or visible.
- Keep the amount of input and output performed for each grid row to the minimum required for the application. For example, avoid associated descriptions wherever possible.
- Use the Stop Processing system function whenever feasible to skip the processing of unnecessary event rules.
- Consider the design for the most efficient method of temporary data storage available at the time, such as cache versus linked list versus work files.

Find/Browse

This list contains standards for increased performance on Find/Browse forms.

- QBE assignments are not used (they negatively impact performance).
- The sort order on the grid should match both an index defined in OneWorld and a logical defined on the AS/400, either partially or completely. The logical file and index must contain at least all fields that are in the grid sort, and the fields selected for the grid sort must be in the same sequence as the logical/index fields. There may be additional fields in the index or logical that are not included in the grid sort. For example,

in a partial match, the grid sort can be KIT, MMCU and the logical and index can include KIT, MMCU, TBM, BQTY.

Header Detail and Headerless Detail

This list contains standards for increased performance on Header Detail and Headerless Detail forms.

- For maximum performance, the grid sort should match an index of the table in the business view. The index should exist on the AS400 and OneWorld tables.
- The sort order on the grid should match both an index defined in OneWorld and a logical file defined on the AS/400, either partially or completely. The logical and index must contain at least all fields that are in the grid sort, and the fields selected for the grid sort must be in the same sequence as the logical/index fields. There may be additional fields in the index or logical that are not included in the grid sort. For example, in a partial match, the grid sort can be KIT, MMCU7 and the logical and index can include KIT, MMCU, TBM, BQTY.

Batch Application Performance

Setting Up Level Breaks

If the sort definition for the section is MMCU, KIT, BQTY, TBM and you want to have a level break on TBM, the level-break indicator must be set for all four fields. A level-break header or footer can then be attached or triggered off a change in TBM.

Slow Performance

If the server side performance for your batch process is extremely and you cannot determine the cause, you may want to check with your system administrator to ensure that the RequestedService setting in the JDE.INI file has the appropriate settings.

Event Rules Performance

When you are creating logic, if a row did not change then skip it. Don't check every row for changes.

Put master business function end docs on the *Post Button Clicked* event.

On asynchronous processing, put the *Post Button Clicked* event only on OK and Cancel. This ensures that your form does not terminate while the business

function is still running. If the form terminates while the business function is still running, the business function cannot return values or handle errors.

Use system functions for repeated processes.

You can also use system functions to get to information that you cannot access, for example to write grid lines.

FetchKeyed is made up of Clear Selection, Select Keyed, and Fetch. Avoid using Fetch Keyed unless the keys are changing.

If keys are not changing use Select, and then in a loop use Fetch Next for better performance, only if you do need to read records sequentially.

You should hide and show fields on the *Dialog is Initialized* event. You should put logic on the *Post Dialog is Initialized* event.

Always code If statements for the usual.

Use explicit comparisons instead of implied. It takes more code, but it runs faster.

You should make sure that the lines in a loop really need to be in there. If something does not really need to be done each time, then do not put it in a loop.

Reduce the number of returns inside code so you have only one exit point.

Use named variables to store temporary values. Do not use hidden controls or grid columns to store temporary values.

Use jdeCache instead of work tables

In early versions of OneWorld, developers used temporary local tables to contain working arrays. This technique is now obsolete. The jdeCache routines provide the similar capabilities but with improved performance.

When multiple business functions access the same cache, locate the related functions in the same source member.

For example, an EditGridLine function adds records to a cache and a corresponding EndDocument function retrieves the cached records and inserts them to the database. If the business functions are placed in different source members, a change in the CNC configuration may cause the application to fail.

Should I use JDB API calls in a business function or table I/O commands in event rule code to manage the database?

From a performance standpoint, there is little difference between the two methods. Make the choice based on the context. For example, if event rule code needs to access the database, use event rule table I/O instead of writing and calling a business function.

When possible, use JDBFetchNext instead of multiple JDBFetch statements

When multiple related database records must be read from the database, it is generally preferable from a performance standpoint to select the records at once and use JDBFetchNext to loop through the qualifying records. The alternative of making multiple calls to JDBFetch is usually considerably slower.

Business Function Performance

You can have data dictionary edits in business functions.

If you have validation in your code and this validation is also part of a master business function, the tool will know that you have already validated and will return and say it is already done, instead of performing the validation twice.

If you go into a grid and do a custom select it is a big performance hit. You might want to restrict people from resequencing the grid. If there is a business need to resequence a grid, make sure there is an index over that table that matches the sort sequence.

Use business function caching to retain pieces in memory.

Do not use linked lists; use caching instead.

Create a structure and pass a pointer instead of adding items.

Make sure variables are used only if needed.

Do not use static variables. You should probably use a single record cache instead. An instance where you might want a static variable would be if you were getting parts of something repeatedly and totalling them.

Because jdeAlloc actually allocates space, you should not generally use it. You can, however, use jdeAlloc if you want to keep a storage area for multiple calls.

For example, suppose you have a function that is split into server and workstation components and you have parent/child information. The client makes a request for what the parent/child looks like. The server needs to know where to go back to. You can use jdeAlloc to keep the information in sequence.

jdeCallObject - When functions are built there is one .c in source. If you have more functions in here, then all functions run on the server if the main one does. You may, however, want some functions to run on the workstation and some on the server. Make sure that if you use more than one function that they are all completely dependent and you want them to run on the same place. jdeCallObject goes across DLLs and platforms.

Memory Allocation

A memory leak is when allocated memory is not freed when it is no longer needed. This can cause the performance of an application to degrade continuously as it runs. You should be aware that jdeCache allocates memory. A cache that is created in a BeginDoc master business function must be destroyed in the corresponding EndDoc function.

One frequent source of unexpected memory leaks comes from failing to free allocated memory when processing errors or other conditions which may traverse an alternate execution path in a business function.

Look at JDEDEBUG.LOG to identify possible memory leaks in business functions. You can also use a third party tool for detecting memory leaks.

Balance Table Opens and Closes

Be sure to match each jdbOpen with a corresponding jdbClose within the same business function for all possible execution paths. Serious performance problems can arise from unbalanced table opens and closes.

Error Messaging Performance

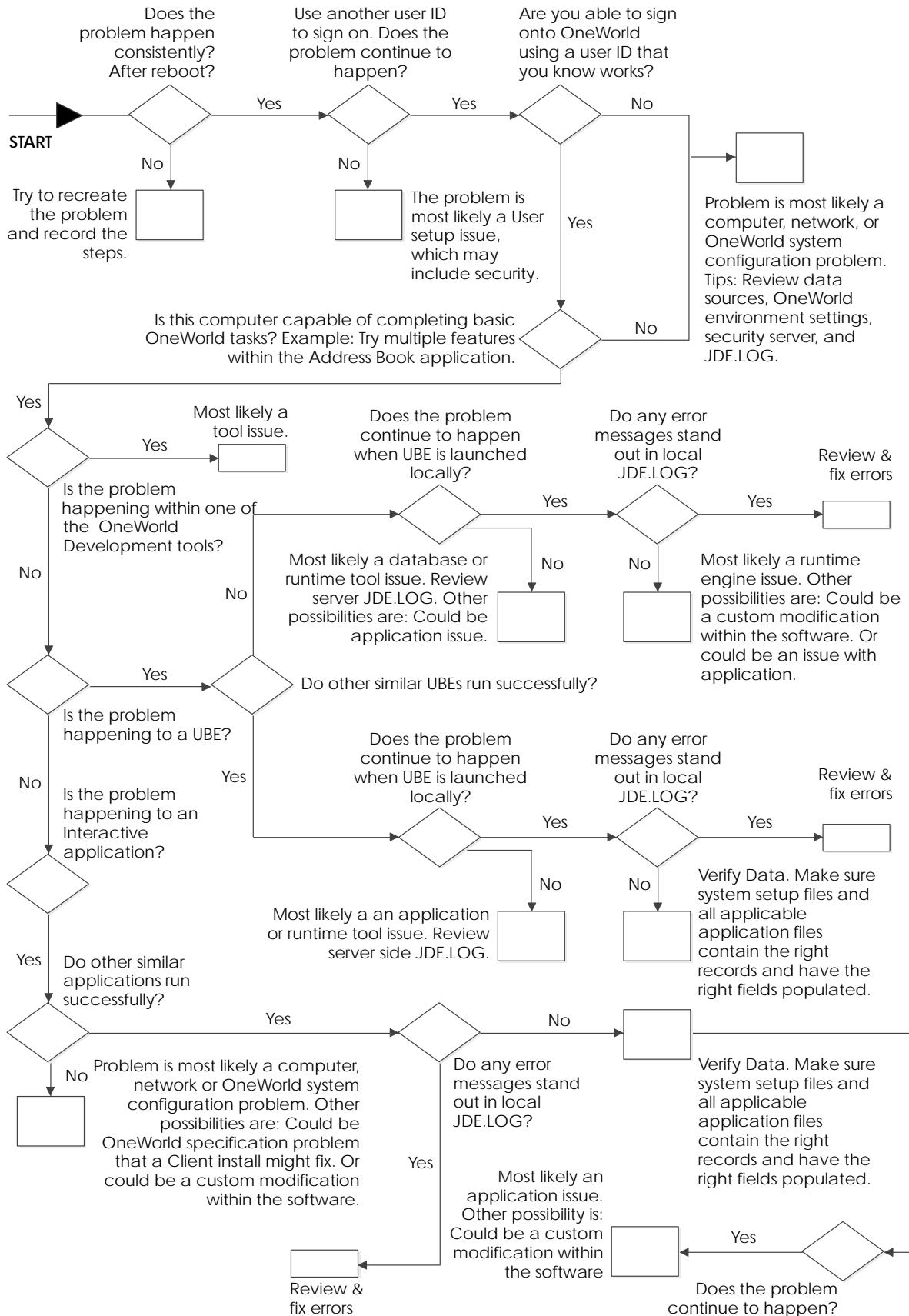
You can classify error messages in the data dictionary as errors or warnings. An error shows as a red stop sign and a warning is a yellow yield sign. Validation is automatically performed using the data dictionary. Other errors must be checked manually using event rules. Specify the field and error message number the error is from.

You must code an If statement in event rules so that if an error is already shown you do not show it again. If the error is only in the data dictionary, the error will show every time.

Transaction Processing Performance

When you design applications that use transaction processing, you should maintain as narrow a scope as possible and you should start your transaction at a point that allows for the shortest possible time between the start of the transaction and the commitment or rollback. When you update a record that is

part of a transaction, it is locked until the transaction is committed. If any part of a transaction fails, the whole transaction rolls back.



Appendices



Appendix A: OneWorld Modification Rules

Because the OneWorld Tools are comprehensive and flexible, you can customize certain aspects of business solutions and applications without making custom modifications. J.D. Edwards refers to this concept as “modless mods,” which are modifications that you can perform easily without the help of a developer. You can perform modless mods on the following:

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

This kind of flexibility improves efficiency and provides distinct advantages, such as the ability to do the following:

- Export grid records to other applications, such as a Microsoft Excel spreadsheet
- Resequence a grid on a different column
- Change grid fonts and colors
- Control major functionality using processing options

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

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

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



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

Both of these kinds of tables go through an automatic merge process during an upgrade where control tables are merged with new J.D. Edwards data, while transaction files are converted to the new specifications, keeping your existing data. For the object specification merges (such as business view, tables, data structures, processing options, event rules, and applications), there are processes in place that merge the specifications or overlay them depending upon the rules defined under *What an Upgrade Preserves and Replaces*.

What an Upgrade Preserves and Replaces

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

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

J.D. Edwards provides you with the Object Librarian Modifications (R9840D) report, which you can run before the upgrade process to identify objects that you modified. For details about what OneWorld preserves and replaces during an upgrade, see:

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

General Rules for Modification

The following general modification rules apply to all OneWorld objects:

- When adding new objects, use system codes 55–59. OneWorld uses reserved system codes that enable it to categorize different applications and vertical groups. By using system codes 55–59 for your custom usage, OneWorld does not overlay your modifications with J.D. Edwards applications.
- Do not create custom or new version names that begin with ZJDE or XJDE. These are reserved prefixes for standard version templates that J.D. Edwards sends out for you to copy, to create new templates or versions. Using these prefixes does not preserve your custom versions in case of a naming conflict.
- For upgrades, you should build a package from the last modified central objects set and perform backups of your development server, central objects, and Object Librarian data sources so you can access those specifications for comparison or for troubleshooting purposes. See the *OneWorld Upgrade Guide* for information.

Interactive Applications

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

What an Upgrade Preserves and Replaces

The following table shows the interactive application elements that are preserved or replaced during an upgrade.

Object	Preserved	Replaced	Comments
New applications	X		<p>There are two ways to create an application: create it from scratch, or copy an existing application using the Application Design Aid's "Copy" feature. This allows you to copy all of the application specifications, including event rules.</p> <p>If you use the Copy feature to copy an existing application for some modifications, during an upgrade your new application <i>does not</i> receive any changes J.D. Edwards might have made to the original application you copied.</p>

Object	Preserved	Replaced	Comments
New hyperitems added to existing forms	X		
New controls added to existing forms	X		
New grid columns added to existing forms	X		
Style changes	X		Style changes include fonts and colors. New J.D. Edwards controls have the standard base definitions, so if you adjust the style you need to do the same for any new controls added to an application.
Code-generator overrides	X		
Data dictionary overrides	X		
Location & size changes	X		If J.D. Edwards decides to place a new control in the new release of the software in the same place you have placed a custom control, the controls display on top of each other. This does not affect the event rules or the functionality of the application. After the upgrade, you can rearrange those controls using Application Design Aid.
Sequence changes for tabs or columns	X		The upgrade process adds new J.D. Edwards controls to the end of your custom tab sequence. You can review the tab sequence after an upgrade.
Custom forms on existing OneWorld applications		X	Instead of putting custom forms on existing OneWorld applications, it is preferable to create a custom application using system codes 55 - 59, and then place the custom form on that custom application. You can then add to existing applications form exits and row exits that call your custom forms within your custom applications. From a performance standpoint there is no difference if an external application is called from a row exit or a form within the application.

Reports

The following rule applies to Report Design Aid specifications:

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

What an Upgrade Preserves and Replaces

The following table shows the report elements that are preserved or replaced during an upgrade.

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

Object	Preserved	Replaced	Comments
New data dictionary variables added to existing reports	X		
Style changes	X		Style changes include fonts and colors. New J.D. Edwards controls have the standard base definitions, so if you have adjusted the style, you need to do the same for any new controls added to a report.
Location and size changes for objects	X		If J.D. Edwards decides to place a new object, such as a control, in the new release of the software in the same place you have placed a custom object, the objects display right next to each other. This does not affect the event rules or the functionality of the report in any way. After the upgrade, you can rearrange objects using Report Design Aid.
Data dictionary overrides	X		
Custom sections on existing OneWorld reports		X	Instead of adding custom sections to existing reports, use Report Interconnect and connect to a new custom report that uses system codes 55 - 59. There is no difference in the performance of a report being called through report interconnections.

Application Text Changes

What an Upgrade Preserves and Replaces

The following table shows the application text elements that are preserved or replaced during an upgrade.

Object	Preserved	Replaced	Comments
Overrides done in Application Design Aid	X		
Overrides done in Report Design Aid	X		

Object	Preserved	Replaced	Comments
Overrides done in Interactive Vocabulary Override	X		
Overrides done in Batch Vocabulary Override	X		

Table Specifications

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

What an Upgrade Preserves and Replaces

The following table shows the table specification elements that are preserved or replaced during an upgrade.

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

For custom tag files, be aware of data item changes in the J.D. Edwards data dictionary. From one release to the next, J.D. Edwards might change certain data item attributes such as data item size, which can affect data integrity and how data is stored in the database. For this reason, you might need to use the Table Conversion tool to convert the tag file data to the new release level. For base J.D. Edwards files, the upgrade process takes care of the data dictionary changes by upgrading the OneWorld database to the new release level. An upgrade preserves custom indices over the custom tag files.

Control Tables

Control tables contain user defined codes (UDCs), menus, and data dictionary items. An upgrade merges your control tables from one release level to the next using the Change Table process, which uses your control tables, *not* J.D. Edwards tables, as the basis to do the data merge.

What an Upgrade Preserves and Replaces

The following table shows the control table elements that are preserved or replaced during an upgrade.

Object	Preserved	Replaced	Comments
Data dictionary custom changes	X		This includes changes to row, column, and glossary text. The upgrade process uses your data dictionary as the base, and in case of a conflict with J.D. Edwards data items, your changes override. Create new data items using system codes 55 - 59.
User defined codes	X		The upgrade process merges any new hard-coded J.D. Edwards values. (Values owned by J.D. Edwards are system 90 and above, and H90 and above.) The process also reports any J.D. Edwards hard-coded values that conflict with your custom values.
Menus	X		In case of a conflict with J.D. Edwards base menus, your custom changes override.
Columns added or removed from existing J.D. Edwards control tables		X	

Business Views

Do not remove columns from existing business views. Changing business views that applications use can cause unpredictable results when you run the application. If you need to hide columns, do so at the application design level using either Application Design Aid or Report Design Aid. There is not much of a performance gain by deleting a few columns from a business view.

What an Upgrade Preserves and Replaces

The following table shows the business view elements that are preserved or replaced during an upgrade.

Object	Preserved	Replaced	Comments
New custom business views	X		
New columns, joins, or indexes added to existing OneWorld business views	X		
Columns removed from J.D. Edwards business views.		X	

Event Rules

What an Upgrade Preserves and Replaces

The following table shows the event rules elements that are preserved or replaced during an upgrade.

Object	Preserved	Replaced	Comments
Custom event rules for custom applications, reports, and tables	X		
Custom event rules for custom business functions	X		
Custom event rules on a new custom control	X		

Object	Preserved	Replaced	Comments
Events for J.D. Edwards applications, reports, and tables that do not have any J.D. Edwards event rules attached to the same event	X		
Events for J.D. Edwards business functions that do not have any J.D. Edwards event rules attached to the same event	X		
Events for J.D. Edwards applications, reports, and tables that have existing event rules attached to the same event		X	An upgrade disables and appends your custom event rules to the end of the event rules. The merge prints a report (R9840D) to notify you of any event rules that the upgrade process disabled.
Events for J.D. Edwards business functions that have event rules attached to the same event		X	An upgrade disables and appends your custom event rules to the end of the event rules. The merge prints a report (R9840D) to notify you of any event rules that the upgrade process disabled.

To restore your custom event rules to J.D. Edwards objects, highlight and drag the event rules back to the proper place in the event and enable them. Prior to an upgrade perform the following:

- Run the Object Librarian Modifications (R9840D) report to identify modified applications
- Print the event rules for the modified application so that you can see the logic behind the event when restoring custom event rules

Data Structures

What an Upgrade Preserves and Replaces

The following table shows the data structure elements that are preserved or replaced during an upgrade.

Object	Preserved	Replaced	Comments
Custom forms data structures	X		
Custom processing options data structures	X		
Custom reports data structures	X		
Custom business functions data structures	X		
Custom generic text data structures	X		
Modifications to existing J.D. Edwards forms data structures		X	
Modifications to existing J.D. Edwards processing options data structures		X	
Modifications to existing J.D. Edwards reports data structures		X	
Modifications to existing J.D. Edwards business functions data structures		X	
Modifications to existing J.D. Edwards generic text data structures		X	

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

Business Functions

For any new custom business functions, create a new (custom) parent DLL to store your custom modifications. See the *OneWorld Development Tools Guide* for details.

To bring your custom changes forward to the next release level, run the Object Librarian Modifications (R9840D) report to list all of the modified business functions, and use this report as a guide for manually refitting the business function changes.

Always use the standard API (jdeCallObject) to call other business functions from within a business function. Not following this and all other standards will cause problems.

To determine modifications to the source of existing base J.D. Edwards business functions, use a third-party source-compare tool, such as Microsoft WinDiff. To determine modifications to APIs within business functions, see the OneWorld online help feature for the most current APIs.

What an Upgrade Preserves and Replaces

The following table shows the business function elements that are preserved or replaced during an upgrade.

Object	Preserved	Replaced	Comments
New custom business function objects	X		
Modifications made to existing J.D. Edwards business function objects		X	NER BSFNs can be modified

Versions

For new custom versions, create a new version with a name that does not begin with XJDE or ZJDE. See the *OneWorld Foundation Guide* for details.

What an Upgrade Preserves and Replaces

The following table shows the versions elements that are preserved or replaced during an upgrade.

Object	Preserved	Replaced	Comments
Non-JDE versions	X		
Version specifications	X		
Processing option data	X		
All ZJDE and XJDE version specifications		X	
All processing option data for XJDE versions		X	

In addition, processing option data is copied but not converted for non-JDE versions using JDE processing option templates. A warning is issued at runtime, and some data may be lost.

Also, event rule modifications for custom versions on JDE templates are not reconciled with the JDE parent template.



Appendix B: Form Processing

Processing for the following form types is discussed in this section:

- Find/Browse
- Parent/Child
- Fix/Inspect
- Header Detail
- Headerless Detail
- Search/Select
- Message Form

Processing for the following controls is also discussed in this section:

- Edit Control
- Grid Control



Process Flow for Find/Browse Form

Description

A Find/Browse form is used to query on a business view and select records from that business view for an operation. The following sections describe the processing flow of a Find/Browse form type.

Default Flags

There are no form option flags checked by default on Find/Browse forms. The only form option flag that can have an impact for this form type is the “No Fetch on Grid Business View” flag. Checking any of the other form option flags does not have any impact on the form processing.

The Find/Browse forms along with Parent/Child Browse forms are the only forms that have the “Entry Point” property checked by default. Find/Browse forms are typically the entry point into applications. The entry point property can be unchecked by the application developer.

Dialog Initialization

1. Initialize Thread Handling
2. Initialize Error Handling Process
3. Initialize Business View Columns
4. Initialize Form Controls
5. Initialize Grid Fields
6. Initialize Static Text
7. Initialize Helps
8. Initialize Event Rules Structures
9. Create Toolbar
10. Load Form Interconnection data into corresponding business view columns and filter fields, if any
11. *Perform Event Rules*: Dialog is Initialized
12. *Perform Event Rules*: Post Dialog is Initialized
13. If the grid option “Automatically Find On Entry” is checked:
 - Begin Detail Data Selection and Sequencing

Header Data Retrieval

There is no header record for Find/Browse type forms.

Detail Data Selection and Sequencing

An internal structure is created representing the data selection and data sequencing requirements specified by the user. This is then passed to the database engine to do the actual database select and sequencing. The data is then held until the data is retrieved.

The data used for selection is pulled from filter fields and QBE Columns.

1. If the form option flag “No Fetch On Grid Business View” is unchecked then the following two actions occur:
 - Select and Sequence
 - Begin Data Retrieval
2. If the form option flag “No Fetch On Grid Business View” is checked no data retrieval is performed.

Data Retrieval

A request is issued to the JDEKRNL, which performs the actual fetch of the data from the database. It will read one record at a time and for each record, perform the following processing:

1. Attempt to fetch a record from the database
2. If a record is fetched
 - Copy the data into the business view data structures
 - *Perform Event Rules: Grid Record is Fetched*
 - If the application developer has not chosen to suppress the writing of this grid record
 - Copy the business view data into the grid data structures
 - Perform Event Rules: *Write Grid Line - Before*
 - Add the row to the grid. The row now exists in the grid control
 - Perform Event Rules: *Write Grid Line - After*
 - Clear the grid data structures for reading the next record
 - Remove the suppress grid line flag

The previous steps occur for each record read from the database. After all records have been read:

3. Perform Event Rules: *Last Grid Record Has Been Read*

Closing Form

1. Perform Event Rules: *End Dialog*
2. Load Form Interconnection data from corresponding business view columns, if any
3. Terminate Error Handling
4. Terminate Thread Handling
5. Terminate Helps
6. Free all structures for form, including structures for Business View Columns, Form Controls, Grid Columns, and Event Rules
7. Destroy the window

Menu/Toolbar Items

Select

Select is a standard item that is automatically placed on Find/Browse forms. There is no default processing for Select on Find/Browse forms. It will behave as a user defined item.

Close

Close is a standard item that is automatically placed on Find/Browse forms. It closes the form.

1. Perform Event Rules: *Button Clicked*
2. Perform Event Rules: *Post Button Clicked*
3. Begin Closing Form

Find

Find is a standard item that is automatically placed on Find/Browse forms. When clicked by the user, it is the signal to the runtime engine to call the database and reload the grid based on the selections in the filter fields.

1. Perform Event Rules: *Button Clicked*
2. If there no errors in any filter fields:
 - Begin Data Selection and Sequencing
 - Perform Event Rules: *Post Button Clicked*

Delete

Delete is a standard item that can be added to Find/Browse forms. It deletes a record in the grid from the database.

1. Perform Event Rules: *Button Clicked*
2. For each selected grid row that is deletable
 - Copy the grid row data into the Business View structures
 - Remove Suppress Delete flag
 - *Perform Event Rules: Delete Grid Rec Verify - Before*
 - If the Suppress Delete flag is not set
 - Display Delete Confirmation dialog. If the user clicks NO or CANCEL the rest of this processing is skipped.
 - *Perform Event Rules: Delete Grid Rec Verify - After*
 - If the Suppress Delete flag is not set
 - Perform Event Rules: Delete Grid Rec from DB - Before*
 - If the Suppress Delete flag is not set:
 - Delete the record in the Business View from the database
 - Delete the grid row from the grid control
 - Perform Event Rules: *Delete Grid Rec from DB - After*
3. Perform Event Rules: *All Grid Recs Deleted*
4. Perform Event Rules: *Post Button Clicked*

Copy

Copy is a standard item that can be added to Find/Browse forms. The copy function allows users to duplicate and rename an object. There is no default processing for copy on Find/Browse forms. They will behave as user defined items. If a form Interconnection is placed on the Button Clicked event then the called form will open in copy mode.

Add

Add is a standard item that can be added to Find/Browse forms. There is no default processing for add on Find/Browse forms. They will behave as user defined items. If a form Interconnection is placed on the *Button Clicked* event then the called form will open in add mode.

User Defined Items

User-defined items are nonstandard items you can add to Find/Browse forms to perform specialized processing not handled by the standard items.

1. Perform Event Rules: *Button Clicked*
2. Perform Event Rules: *Post Button Clicked*

Process Flow for Parent/Child Form

Description

A Parent/Child form is used to query on a business view and represent the data in a hierarchical manner. Records can also be selected from that business view for an operation. The following sections describe the processing flow of a Parent/Child form type.

Default Flags

There are no form options flags checked by default on Parent/Child forms. The only form option flag that can have an impact for this form type is the “No Fetch on Grid Business View” flag. Checking any of the other form option flags does not have any impact on the form processing.

The Parent/Child Browse forms along with Find/Browse forms are the only forms that have the “Entry Point” property checked by default. Parent/Child forms are typically the entry points into applications. The entry point property can be unchecked by the application developer.

Dialog Initialization

1. Initialize Thread Handling
2. Initialize Error Handling Process
3. Initialize Business View Columns
4. Initialize Form Controls
5. Initialize Grid/Tree Fields
6. Initialize Static Text
7. Initialize Helps
8. Initialize Event Rules Structures
9. Create Toolbar
10. Load Form Interconnection data into corresponding business view columns and filter fields, if any
11. Perform Event Rules: *Dialog is Initialized*
12. Perform Event Rules: *Post Dialog is Initialized*
13. If the grid option flag “Automatically Find On Entry” is checked

- Begin [Detail Data Selection and Sequencing](#)

Header Data Retrieval

There is no header record for Parent/Child forms.

Detail Data Selection and Sequencing

An internal structure is now created representing the data selection and data sequencing requirements specified by the user. This is then passed to the database engine to do the actual database select and sequencing. The data is then held until the data is retrieved in the next step.

The data used for selection is pulled from filter fields and QBE Columns.

1. If the form option flag “No Fetch On Grid Business View” is unchecked
 - Select and Sequence
 - Begin [Data Retrieval](#)

Data Retrieval

A request is issued to the JDEKRNL, which performs the actual fetch of the data from the database. It will read one record at a time and for each record, perform the required processing. The data retrieval is performed in two different ways in the Parent/Child browse form. The fetch performed to get the first level nodes in the tree is similar to the one performed in the Find/Browse form:

Data Retrieval for the first level node in the tree

1. Attempt to fetch a record from the database
2. If a record is fetched
 - Copy the data into the business view data structures
 - Perform Event Rules: *Grid Record is Fetched*
 - If the application developer has not chosen to suppress the writing of this record
 - Copy the business view data into the grid data structures
 - Perform Event Rules: *Write Grid Line - Before*
 - Add the row to the grid and the corresponding column in Tree. The row and the corresponding tree node now exist in the control.
 - Perform Event Rules: *Write Grid Line - After*
 - Clear the data structures for reading the next record.

The previous steps occur for each record read from the database. After all records have been read:

1. If the Parent/Child Browse form has the grid permanently hidden, then expand the header node and move the selection to the first child node under the header. This initiates the events: 'Tree - Node Level Changed' and 'Tree - Node Selection Changed.'
2. Perform Event Rules: *Last Grid Record Has Been Read*

The Parent/Child form also performs data retrieval whenever a particular node on the tree is expanded. However, this fetch is performed only once for each node that is expanded. If a particular node is collapsed and expanded again, then the tree and the grid are replenished from internal structures. The processing performed for node expand event is given below:

Data Retrieval for Tree Node Expand

1. Perform Event Rules: *Tree - Node is expanding*
2. Check for internal flags to see if the fetch is not suppressed. If the fetch is suppressed through the system function, *Suppress Fetch on Node Expand*, the processing stops here.
3. Perform the key substitution (copying child key into parent key), that was setup in the design of the form.
4. Attempt to fetch a record from the database
5. If a record is fetched
 - Copy the data into the business view data structures
 - Perform Event Rules: *Grid Record is Fetched*
 - If the application developer has not chosen to suppress the writing of this record
 - Copy the business view data into the grid data structures
 - Perform Event Rules: *Write Grid Line - Before*
 - Add the row to the grid and the corresponding column in Tree. The row and the corresponding tree node now exist in the control.
 - Perform Event Rules: *Write Grid Line - After*
 - Clear the data structures for reading the next record.
6. Perform Event Rules: *Last Grid Record Has Been Read*
7. Move the tree selection to the first child node under the expanding node. This initiates the Events: *Tree - Node Level Changed* and *Tree - Node Selection Changed*.

Closing Form

1. Perform Event Rules: *End Dialog*
2. Load Form Interconnection data from corresponding business view columns, if any
3. Terminate Error Handling
4. Terminate Thread Handling
5. Terminate Helps
6. Free all structures for form, including structures for Business View Columns, Form Controls, Grid Columns, and Event Rules.
7. Destroy the window

Menu/Toolbar Items

Select

Select is a standard item that is automatically placed on Parent/Child Browse forms. There is no default processing for Select on Parent/Child Browse forms. It will behave as a user-defined item.

Close

Close is a standard item that is automatically placed on Parent/Child Browse forms. It closes the form.

1. Perform Event Rules: *Button Clicked*
2. Perform Event Rules: *Post Button Clicked*
3. Begin Closing Form

Delete

Delete is a standard item that can be added to Parent/Child Browse forms. Similar to the Find/Browse form, the Parent/Child Browse form does the following.

1. Perform Event Rules: *Button Clicked*
2. For the selected tree node
 - Copy the grid row data into the Business View structures
 - Remove Suppress Delete flag
 - Perform Event Rules: *Delete Grid Rec Verify - Before*
 - If the Suppress Delete flag is not set

Display Delete Confirmation dialog. If the user clicks No or Cancel the rest of this processing is skipped.

Perform Event Rules: *Delete Grid Rec Verify - After*

If the Suppress Delete flag is not set

- Perform Event Rules: *Delete Grid Rec from DB - Before*
- If the Suppress Delete flag is not set

Delete the record in the Business View from the database.

This will not delete the child records of the node, if any, from the database. It is your responsibility to delete them from the database.

Delete the grid row from the grid control

Perform Event Rules: *Delete Grid Rec from DB - After*

3. Perform Event Rules: *All Grid Recs Deleted*
4. Perform Event Rules: *Post Button Clicked*

Find

Find is a standard item that is automatically placed on Parent/Child Browse forms. When clicked by the user, it is the signal to the runtime engine to call the database and reload the grid based on the selections in the Filter Fields and QBE columns.

1. Perform Event Rules: *Button Clicked*
2. If there no errors in any filter fields
 - Begin Data Selection and Sequencing
 - Perform Event Rules: *Post Button Clicked*

Copy

Copy is a standard item that can be added to Parent/Child Browse forms. There is no default processing for copy on Parent/Child Browse forms. They will behave as user defined items. If a form interconnection is placed on the Button Clicked event then the called form will open in copy mode.

Add

Add is a standard item that can be added to Parent/Child forms. There is no default processing for add on Parent/Child forms. They will behave as user

defined items. If a form interconnection is placed on the *Button Clicked* event then the called form will open in add mode.

User-Defined Items

User-defined items are nonstandard items that you can add to Parent/Child forms to perform specialized processing not handled by the standard items.

1. Perform Event Rules: *Button Clicked*
2. Perform Event Rules: *Post Button Clicked*

Process Flow for Fix/Inspect Form

Description

A Fix/Inspect form is used to update and insert single data base records. It is also referred to as a Single Record Maintenance form. This form type can also be used to display static information to the user as well as prompt the user for information, for example a Sign-on Screen. The following sections describe the processing flow of a Fix/Inspect form type.

Default Flags

None of the form option flags are set by default.

Dialog Initialization

1. Initialize Thread Handling
2. Initialize Error Handling Process
3. Initialize Business View Columns
4. Initialize Form Controls
5. Initialize Static Text
6. Initialize Helps
7. Initialize Event Rules Structures
8. Create Toolbar
9. Load Form Interconnection data into corresponding business view columns. (Note: Form controls and business view columns are sharing internal memory so copying into the business view is effectively copying into the internal form control memory locations.)
10. Perform Event Rules: *Dialog is Initialized*
11. If the “No Fetch Form Business View” is checked
 - Perform Event Rules: *Post Dialog is Initialized*
12. If the form is in Add Mode
 - Begin Clearing Dialog
13. If the form is in update mode and “No Fetch Form Business View” is checked
 - Change mode to add mode

- Display the internal form control values to the screen
 - Begin Clearing Dialog
14. If the form is in update mode and “No Fetch Form Business View” is not checked
 - Begin Data Retrieval

Data Retrieval

A request is issued to the JDEKRNL, which performs the actual fetch of the data from the database. The fetch will be based on the information passed to the form through its form data structure.

1. Attempt to fetch a record from the database
2. If a record is fetched
 - Copy the data into the business view data structures (Note: remember that copying into the business view is effectively copying into the internal form control memory locations.)
3. If a record is found and the form is in Copy Mode
 - Switch to Add Mode
 - Display the internal form control values to the screen
 - Begin Clearing Dialog
4. If a record is found and not in Copy mode.
 - Perform Event Rules: *Post Dialog is Initialized*
5. If no records were fetched
 - Switch to Add Mode.
 - Display the internal form control values to the screen
 - Begin Clearing Dialog
6. Set the default focus based on the resulting data base mode

Clearing Dialog

1. If the form was called in Copy Mode
 - Clear the key (primary index) controls that have the “Do Not Clear After Add” flag unchecked
2. If the form was not called in Copy Mode
 - Clear all form controls that have the “Do Not Clear After Add” flag unchecked
3. Perform Event Rules: *Clear Screen Before Add*
4. Perform Event Rules: *Post Dialog is Initialized*

Closing Form

1. Perform Event Rules: *End Dialog*
2. Load Form Interconnection data from corresponding business view columns, if any
3. Terminate Error Handling
4. Terminate Thread Handling
5. Terminate Helps
6. Free all structures for form, including structures for Business View Columns, Form Controls, and Event Rules.
7. Destroy the window

Menu/Toolbar Items

OK

OK is a standard item that is automatically placed on Fix/Inspect forms. It validates the information on the form and updates or adds to the database through JDEKRNL function calls.

1. If there are any errors/warnings on the form, stop OK processing
2. *Perform Event Rules: Button Clicked*
3. For each control on the form
 - If the current control is a form control and it has not passed validation
 - *Perform Event Rules: Control is Exited*
 - *Perform Event Rules: Control is Exited and Changed - Inline*
 - *Perform Event Rules: Control is Exited and Changed - Async*
 - Perform Data Dictionary validation
4. If there are any errors on the form, stop OK processing
5. If the form is in Add mode
 - *Perform Event Rules: Add Record to Database - Before*
6. If the form is in Update mode
 - *Perform Event Rules: Update Record to Database - Before*
7. Perform Event Rules: Post Button Clicked
6. If form is in Add Mode
 - If form was called in Copy Mode or the flag 'End Form On Add' is checked

- Begin Closing Form
- Else
 - Begin Clearing Dialog
- 7. If the form is in Update Mode
 - If there were no errors attempting to update/add to the database
 - Begin Closing Form

Cancel

Cancel is a standard item that is automatically placed on Fix/Inspect forms.

1. *Perform Event Rules:* Button Clicked
2. *Perform Event Rules:* Post Button Clicked
3. Begin Closing Form

User-Defined Items

User-defined items are non-standard items that a developer can add to Fix/Inspect forms to perform specialized processing not handled by the standard items.

1. *Perform Event Rules:* Button Clicked
2. *Perform Event Rules:* Post Button Clicked

Process Flow for Header Detail Form

Description

A Header Detail form is used when a relationship exists between the information in the header and the information contained in the grid. The Header portion uses one business view, and the grid (detail) portion of the form uses another business view. This form type and the Headerless Detail form type are referred to as Transaction forms. The following sections describe the processing flow of a Header Detail form type.

Default Flags

No form option flags are automatically set for this form type. All form option flags can be checked or unchecked by the application developer.

Dialog Initialization

1. Initialize Thread Handling
2. Initialize Error Handling Process
3. Initialize Business View Columns
4. Initialize Form Controls
5. Initialize Grid Fields
6. Initialize Static Text
7. Initialize Helps
8. Initialize Event Rules Structures
9. Create Toolbar
10. Load Form Interconnection data into corresponding business view columns and filter fields, if any
11. Perform Event Rules: *Dialog is Initialized*
12. If the form option flag “No Fetch On Form Business View” is checked
 - If the form is not in Add mode and was not entered with a Copy button
 - Perform Event Rules: *Post Dialog is Initialized*
 - Copy the Form Control values to the screen

13. If the form is in Update mode (note that this includes forms entered with a Copy button)
 - If the form option flag “No Fetch On Form Business View” is checked
 - Begin Detail Data Selection and Sequencing
 - If the form option flag “No Fetch On Form Business View” is unchecked
 - Begin Header Data Retrieval
14. If the form is in Add Mode
 - Begin Clearing Dialog

Header Data Retrieval

A key structure is built for the business view of the header based on the header Business View Columns, or from the filter fields, if any exist. Then a call is made to the database attempting to retrieve the record specified.

1. If the database fetch is successful
 - Copy the fetched database record to the header Business View Columns
 - If the form was not opened with a Copy button
 - Perform Event Rules: *Post Dialog is Initialized* (note that where form controls and business view columns share memory, the form controls will have the values from the database during this event, which may mean blanks in the case of strings and characters)
 - Copy the Business View Columns to the Form Controls
 - If the grid option “Automatically Find on Entry” is checked
 - Begin Detail Data Selection and Sequencing
 - If the grid option “Automatically Find on Entry” is unchecked
 - Begin Add Entry to Row to Grid
2. If the database fetch failed
 - Begin Clearing Dialog.

Detail Data Selection and Sequencing

An internal structure is now created representing the data selection and data sequencing requirements specified by the user for the detail portion. This is then passed to the database engine to do the actual database select and sequencing. The data is retrieved in the next step.

The data used for selection is pulled from filter fields if any exist, or from the key business view columns if there are no filter fields.

1. If the form option flag “No Fetch On Grid Business View” is unchecked
 - Select and Sequence
 - Begin Data Retrieval
2. If the form option flag ‘No Fetch On Grid Business View’ is checked
3. Begin Add Entry Row To Grid.

Data Retrieval

A request is issued to the JDEKRNL, which performs the actual fetch of the data from the database. It will read one record at a time and for each record, perform the following processing:

1. Attempt to fetch a detail record from the database
2. If a record is fetched
 - Copy the data into the detail business view data structures
 - Perform Event Rules: *Grid Record is Fetched*
 - If the application developer has not chosen to suppress the writing of this grid record
 - Copy the business view data into the grid data structures
 - Perform Event Rules: *Write Everest Grid Line - Before*
 - Add the row to the grid. The row now exists in the grid control.
 - Perform Event Rules: *Write Everest Grid Line - After*
 - Clear the grid data structures for reading the next record.
 - Remove the suppress grid line flag.

The previous steps occur for each record read from the database. After all records have been read, the runtime engine will then:

1. If the form was opened with a Copy button
 - Switch to Add Mode
 - Begin Clearing Dialog once Detail Data Retrieval processing is complete
2. If no records were fetched
 - Switch to Add Mode
3. Perform Event Rules: *Last Grid Record Has Been Read* (note that this event is run regardless if records were actually fetched)
4. Begin Add Entry Row to Grid.

Clearing Dialog

1. If the form was called in Copy Mode
 - Clear the key controls that have the “Do Not Clear After Add“ flag unchecked
2. If the form was not called in Copy Mode
 - Clear all form controls that have the “Do Not Clear After Add“ flag unchecked
3. Perform Event Rules: *Clear Screen Before Add*
4. Delete any existing rows
5. Perform Event Rules: *Post Dialog is Initialized*
6. Begin Add Entry Row to Grid

Add Entry Row to Grid

1. Clear grid data structures
2. Perform Event Rules: *Add Last Entry Row to Grid*
3. Add the row to the grid control

Closing Form

1. Perform Event Rules: *End Dialog*
2. Load Form Interconnection data from corresponding business view columns, if any
3. Terminate Error Handling
4. Terminate Thread Handling
5. Terminate Helps
6. Free all structures for form, including structures for Business View Columns, Form Controls, Grid Columns, and Event Rules.
7. Destroy the window

Menu/Toolbar Items

OK

OK is a standard item that is automatically placed on header detail forms. It validates the information on the form and updates or adds to the database through JDEKRNL function calls.

1. If there are any errors/warnings on the form, stop OK processing.
2. Perform Event Rules: *Button Clicked*

-
3. For each control on the form
 - If the current control is a form control and it has not passed validation
 - Perform Event Rules: *Control is Exited*
 - Perform Event Rules: *Control is Exited and Changed - Inline*
 - Perform Event Rules: *Control is Exited and Changed - Asynch*
 - Perform Data Dictionary validation
 - If the current control is a grid control
 - For each grid row

If the current grid row needs to have Leave Row processing run and the row is updatable

Perform Event Rules: *Row is Exited and Changed - Inline* for current row

Perform Event Rules: *Row is Exited and Changed - Asynch* for current row
 4. If there are any errors on the form, stop OK processing
 5. Delete from the database any grid rows that are in the delete stack. See Delete for details. For each grid row in the delete stack:
 - Copy the information from the delete stack into the grid data structures
 - Copy the grid data structures into the Business View structures
 - Perform Event Rules: *Delete Grid Record from DB - Before*
 - If the database delete has not been suppressed
 - Delete the record in the Business View from the database
 - Delete the grid row from the grid control
 - Perform Event Rules: *Delete Grid Record from DB - After*
 6. Perform Event Rules: *All Grid Recs Deleted from DB*
 7. If the form option flag 'No Update On Form Business View' is unchecked
 - If the form is in Add mode
 - *Perform Event Rules: Add Record to Database - Before*
 - If the database add has not been suppressed

Add the header business view record to the database
 - Perform Event Rules: *Add Record to Database - After*
 - If the form is in Update mode

- Perform Event Rules: *Update Record to Database - Before*
 - If the database Update has not been suppressed

Update the record to the database

Perform Event Rules: *Update Record to Database - After*
8. If the form option flag 'No Update On Grid Business View' is unchecked
- For each grid row that was changed or added
 - Clear the business view data structures
 - Reset the original key values for this row in the business view data structures
 - Copy grid data structures to the business view data structures
 - Copy all non-filter database form controls to the business view data structures
 - Copy all equal filters to the business view data structures
 - If form is in Add Mode

Perform Event Rules: *Add Grid Rec to DB - Before*

Add the record in the business view data structure to the database

Perform Event Rules: *Add Grid Rec to DB - After*
 - If form is in Update Mode

Perform Event Rules: *Update Grid Rec to DB - Before*

Update the record in the business view data structure to the database

Perform Event Rules: *Update Grid Rec to DB - After*
 - If form is in Add Mode
 - Perform Event Rules: *All Grid Recs Added to DB*
 - If form is in Update Mode
 - Perform Event Rules: *All Grid Recs Updated to DB*
9. Perform Event Rules: *Post Button Clicked*
10. If form is in Add Mode
- If form was called in Copy Mode or the flag 'End Form On Add' is checked
 - Begin Closing Form

-
- Else
 - Begin Clearing Dialog
11. If the form is in Update Mode
 - If there were no errors attempting to update/add to the database
 - Begin Closing Form

Cancel

Cancel is a standard item that is automatically placed on header detail forms.

1. Perform Event Rules: *Button Clicked*
2. Perform Event Rules: *Post Button Clicked*
3. Begin Closing Form

Delete

Delete is a standard item that can be added to Header Detail forms. The delete applies to the grid record(s) selected. The actual delete from the database does not happen at this point, so that if the user clicks cancel, the records are not deleted from the database. The delete verification happens when the delete is pressed, and the actual database delete happens when OK is pressed.

1. Perform Event Rules: *Button Clicked*
2. For each selected grid row that is deletable
 - Remove Suppress Delete flag
 - Perform Event Rules: *Delete Grid Rec Verify - Before*
 - If the Suppress Delete flag is not set

Display Delete Confirmation dialog. If the user clicks NO or CANCEL the rest of this processing is skipped

Perform Event Rules: *Delete Grid Rec Verify - After*

If the Suppress Delete flag is not set

- If the record was read from the database
 - Add this record to the delete stack (records to be deleted if the user presses OK)
- Delete the grid row from the grid control

Find

Find is a standard item that can be added to header detail forms. When clicked by the user, it is the signal to the runtime engine to call the database and reload the grid based on the selections in the filter fields.

1. Perform Event Rules: *Button Clicked*
2. If there no errors in any filter fields
 - Switch to Update Mode
 - Begin Detail Data Selection and Sequencing
 - Perform Event Rules: *Post Button Clicked*

User-Defined Items

User-defined items are non-standard items that a developer can add to Header Detail forms to perform specialized processing not handled by the standard items.

1. Perform Event Rules: *Button Clicked*
2. Perform Event Rules: *Post Button Clicked*

Process Flow for Headerless Detail Form

Description

A Headerless Detail form is used to update and enter records that have information that is common to all records in a selected group. This form type and the Header Detail form type are referred to as Transaction forms. The following sections describe the processing flow of a Headerless Detail form type.

Default Flags

The following form option flags are automatically checked for Headerless Detail Forms and should not be unchecked: “No Update On Form Business View”, “No Fetch On Form Business View”. Other form option flags can be checked or unchecked by the application developer.

Dialog Initialization

1. Initialize Thread Handling
2. Initialize Error Handling Process
3. Initialize Business View Columns
4. Initialize Form Controls
5. Initialize Grid Fields
6. Initialize Static Text
7. Initialize Helps
8. Initialize Event Rules Structures
9. Create Toolbar
10. Load Form Interconnection data into corresponding business view columns and filter fields, if any
11. Perform Event Rules: *Dialog is Initialized*
12. If the form is not in Add Mode
 - If the form is not in Copy Mode
 - Perform Event Rules: *Post Dialog is Initialized* (note that this event is run immediately after Dialog is Initialized when in Update mode, so the FC values are still at their NULL or zero value)

- If the grid option “Automatically Find on Entry” is checked
 - Begin Data Selection and Sequencing
- If the grid option “Automatically Find on Entry” is unchecked
 - Begin Add Entry Row to Grid
- If the form is in Add Mode
 - Begin Clearing Dialog

Data Selection and Sequencing

An internal structure is now created representing the data selection and data sequencing requirements specified by the user. This is then passed to the database engine to do the actual database select and sequencing. The data is then held until the data is retrieved in the next step.

The data used for selection is pulled from filter fields if any exist, or from the key business view columns if there are no filter fields.

1. If the form option flag “No Fetch On Grid Business View” is unchecked
 - Select and Sequence
 - Begin Data Retrieval
2. If the form option flag “No Fetch On Grid Business View” is checked
 - Begin Add Entry Row To Grid

Data Retrieval

A request is issued to the JDEKRNL, which performs the actual fetch of the data from the database. It will read one record at a time and for each record, perform the following processing:

1. Attempt to fetch a record from the database
2. If a record is fetched
 - Copy the data into the business view data structures
 - Perform Event Rules: *Grid Record is Fetched*
 - If the application developer has not chosen to suppress the writing of this grid record
 - Copy the business view data into the grid data structures
 - Perform Event Rules: *Write Grid Line - Before*
 - Add the row to the grid. The row now exists in the grid control.
 - Perform Event Rules: *Write Grid Line - After*
 - Clear the grid data structures for reading the next record.
 - Remove the suppress grid line flag.

The previous steps occur for each record read from the database. After all records have been read, the runtime engine will then:

1. If the form is in Copy Mode
 - Switch to Add Mode
 - Begin Clearing Dialog once Data Retrieval processing is complete
2. If no records were fetched
 - Switch to Add Mode
3. Perform Event Rules: *Last Grid Record Has Been Read* (note that this event is run regardless of whether records were actually fetched)
4. Begin Add Entry Row to Grid.

Clearing Dialog

1. If the form was called in Copy Mode
 - Clear the key controls that have the “Do Not Clear After Add” flag unchecked
2. If the form was not called in Copy Mode
 - Clear all form controls that have the “Do Not Clear After Add” flag unchecked
3. Perform Event Rules: *Clear Screen Before Add*
 - Delete any existing grid rows
4. Perform Event Rules: *Post Dialog is Initialized*
5. Begin Add Entry Row to Grid

Add Entry Row to Grid

1. Clear grid data structures
2. Perform Event Rules: *Add Last Entry Row to Grid*
3. Add the row to the grid control

Closing Form

1. Perform Event Rules: *End Dialog*
2. Load Form Interconnection data from corresponding business view columns, if any
3. Terminate Error Handling
4. Terminate Thread Handling
5. Terminate Helps

6. Free all structures for form, including structures for Business View Columns, Form Controls, Grid Columns, and Event Rules
7. Destroy the window

Menu/Toolbar Items

OK

OK is a standard item that is automatically placed on Headerless Detail forms. It validates the information on the form and updates or adds to the database through JDEKRNL function calls.

1. If there are any errors/warnings on the form, stop OK processing
2. *Perform Event Rules: Button Clicked*
3. For each control on the form
 - If the current control is a form control and it has not passed validation
 - Perform Event Rules: *Control is Exited*
 - Perform Event Rules: *Control is Exited and Changed - Inline*
 - Perform Event Rules: *Control is Exited and Changed - Asynch*
 - Perform Data Dictionary validation
 - If the current control is a grid control, perform the following for each grid row:
 - Perform Event Rules: *Row is Exited and Changed - Inline* for current row
 - Perform Event Rules: *Row is Exited and Changed - Asynch* for current row
4. If there are any errors on the form, stop OK processing
5. Delete from the database any grid rows that are in the delete stack. See Delete for details. For each grid row in the delete stack:
 - Copy the grid row data into the Business View structures
 - Copy the grid data structures into the business view data structures
 - Perform Event Rules: *Delete Grid Record from DB - Before*
 - If the database delete has not been suppressed
 - Delete the record in the Business View from the database
 - Delete the grid row from the grid control
 - Perform Event Rules: *Delete Grid Record from DB - After*
6. Perform Event Rules: *All Grid Recs Deleted from DB*
7. If the form option flag 'No Update On Grid Business View' is unchecked

-
- For each grid row that was changed or added
 - Clear the business view data structures
 - Reset the original key values for this row in the business view data structures
 - Copy grid data structures to the business view data structures
 - Copy all non-filter database form controls to the business view data structures
 - Copy all equal filters to the business view data structures
 - If form is in Add Mode

Perform Event Rules: *Add Grid Rec to DB - Before*

Add the record in the business view data structure to the database

Perform Event Rules: *Add Grid Rec to DB - After*
 - If form is in Update Mode

Perform Event Rules: *Update Grid Rec to DB - Before*

Update the record in the business view data structure to the database

Perform Event Rules: *Update Grid Rec to DB - After*
 - If form is in Add Mode
 - Perform Event Rules: *All Grid Recs Added to DB*
 - If form is in Update Mode
 - Perform Event Rules: *All Grid Recs Updated to DB*
8. Perform Event Rules: *Post Button Clicked*
9. If form is in Add Mode
- If form was called in Copy Mode or the flag 'End Form On Add' is checked
 - Begin Closing Form
 - Else
 - Begin Clearing Dialog
10. If the form is in Update Mode
- If there were no errors attempting to update/add to the database
 - Begin Closing Form

Cancel

Cancel is a standard item that is automatically placed on Headerless Detail forms.

1. Perform Event Rules: *Button Clicked*
2. Perform Event Rules: *Post Button Clicked*
3. Begin Closing Form

Delete

Delete is a standard item that can be added to Headerless Detail forms. The actual delete from the database does not happen at this point, so that if the user clicks cancel, the records are not deleted from the database. The delete verification happens when delete is pressed, and the actual database delete happens when OK is pressed.

1. Perform Event Rules: *Button Clicked*
2. For each selected grid row that is deletable
 - Remove Suppress Delete flag
 - Perform Event Rules: *Delete Grid Rec Verify - Before*
 - If the Suppress Delete flag is not set
 - Display Delete Confirmation dialog. If the user clicks NO or CANCEL the rest of this processing is skipped.
 - Perform Event Rules: *Delete Grid Rec Verify - After*
 - If the Suppress Delete flag is not set
 - If the record was read from the database
 - Add this record to the delete stack (records to be deleted if the user presses OK)
 - Delete the grid row from the grid control

Find

Find is a standard item that can be added to Headerless Detail forms. When clicked by the user, it is the signal to the runtime engine to call the database and reload the grid based on the selections in the Filter Fields.

1. Perform Event Rules: *Button Clicked*
2. If there no errors in any filter fields
 - Switch to Update Mode
 - Begin Data Selection and Sequencing
 - Perform Event Rules: *Post Button Clicked*

User-Defined Items

User-defined items are non-standard items that a developer can add to Headerless Detail forms to perform specialized processing not handled by the standard items.

1. Perform Event Rules: *Button Clicked*
2. Perform Event Rules: *Post Button Clicked*

Process Flow for Search/Select Form

Description

A Search/Select form is used to select a single predetermined field from a record in a predetermined file. It is extremely important to note that Search/Select forms return only one value to the calling form, based on the dictionary alias. If there is no data dictionary alias match, the first value from the data structure will be returned. The following sections describe the processing flow of a Search/Select form type. A Search/Select form should only be attached as a visual assist to data items that have the same data type as the form data structure element.

Default Flags

No form option flags are automatically set for this form type, but the Grid option flag 'Automatically Find on Entry' is often checked by application developers. The only form option flag that impacts this form type is the "No Fetch on Grid Business View" flag. Checking any of the other form option flags does not have any impact on the form processing.

Dialog Initialization

1. Initialize Thread Handling
2. Initialize Error Handling Process
3. Initialize Business View Columns
4. Initialize Form Controls
5. Initialize Grid Fields
6. Initialize Static Text
7. Initialize Helps
8. Initialize Event Rules Structures
9. Create Toolbar
10. Load Form Interconnection data into corresponding business view columns and filter fields, if any
11. Perform Event Rules: *Dialog is Initialized*
12. Perform Event Rules: *Post Dialog is Initialized*
13. If the grid option "Automatically Find On Entry" is checked

- Begin Detail Data Selection and Sequencing

Header Data Retrieval

There is no header record on Search/Select Forms.

Detail Data Selection and Sequencing

An internal structure is now created representing the data selection and data sequencing requirements specified by the user. This is then passed to the database engine to do the actual database select and sequencing. The data is then held until the data is retrieved in the next step.

The data used for selection is pulled from filter fields.

1. If the form option flag “No Fetch On Grid Business View” is unchecked
 - Select and Sequence
 - Begin Data Retrieval

Data Retrieval

A request is issued to the JDEKRNL, which performs the actual fetch of the data from the database. It will read one record at a time and for each record, perform the following processing:

1. Attempt to fetch a record from the database
2. If a record is fetched
 - Copy the data into the business view data structures
 - Perform Event Rules: *Grid Record is Fetched*
 - If the application developer has not chosen to suppress the writing of this grid record
 - Copy the business view data into the grid data structures
 - Perform Event Rules: *Write Grid Line - Before*
 - Add the row to the grid. The row now exists in the grid control
 - Perform Event Rules: *Write Grid Line - After*
 - Clear the grid data structures for reading the next record
 - Remove the suppress grid line flag

The previous steps occur for each record read from the database. The following occurs only once.

- Perform Event Rules: *Last Grid Record Has Been Read*

Clearing Dialog

Not done for this form type.

Closing Form

1. Perform Event Rules: *End Dialog*
2. Load Form Interconnection data from corresponding business view columns, if any
3. Terminate Error Handling
4. Terminate Helps
5. Free all structures for form, including structures for Business View Columns, Form Controls, Grid Columns, and Event Rules
6. Destroy the window

Menu/Toolbar Items

Select

Select is a standard item that is automatically placed on Search/Select forms. It returns a value to the form interconnection and closes the form.

1. Copy the selected grid row into the business view column
2. Perform Event Rules: *Button Clicked*
3. Perform Event Rules: *Post Button Clicked*
4. Begin Closing Form

Close

Close is a standard item that is automatically placed on Search/Select forms.

1. Perform Event Rules: *Button Clicked*
2. Perform Event Rules: *Post Button Clicked*
3. Begin Closing Form

Find

Find is a standard item that can be added to Search/Select forms. When clicked by the user, it is the signal to the runtime engine to call the database and reload the grid based on the selections in the Filter Fields.

1. Perform Event Rules: *Button Clicked*

2. Begin Data Selection and Sequencing
3. Perform Event Rules: *Post Button Clicked*

User-Defined Items

User-defined items are non-standard items that a developer can add to Search/Select forms to perform specialized processing not handled by the standard buttons.

1. Perform Event Rules: *Button Clicked*
2. Perform Event Rules: *Post Button Clicked*

Process Flow for Message Form

Description

The Message form type is a form that comes up as a secondary window to inform the user of something or to ask a question. It parallels the behavior of a Windows message box. The form does not have a toolbar or a status bar and can only contain static text and buttons. It is the only form type that can have standard buttons appearing on the form instead of the toolbar.

Dialog Initialization

1. Initialize Form Controls
2. Initialize Static Text
3. Initialize Helps
4. Initialize Event Rules Structures
5. Perform Event Rules: *Dialog is Initialized*

Closing Form

1. Terminate Helps
2. Free all structures for Controls and Event Rules.
3. Destroy the window

Buttons

The form has an OK button by default. Other options include Cancel, Yes, and No. All of these buttons will cause the dialog to close, and there is no other default processing.

J.D. Edwards Standards

These forms may only have static text and buttons. Event rules are limited to only *Dialog is Initialized* and any button events. This event rule may not contain any Form Interconnection calls.

Process Flow for Edit Control

Description

An edit control is a text field on a form. All form types can contain edit controls except Message forms. There are two types of edit controls. The first type is commonly referred to as a database field. It is associated with an item in the business view and through that connection to a dictionary item. Database fields represent a field in a database record. The second type of edit control is commonly referred to as a data dictionary field. Although only one type is called a data dictionary field, both types have a connection to a specific data dictionary item.

Within the realm of database fields there is an added distinction between filter fields and non-filter fields. Database fields are nonfilter fields by default. A filter field is used to alter the selection criteria of a database fetch. A filter can have a comparison type of equal, not equal, less than, less than or equal to, greater than, or greater than or equal to. A filter can be marked so that a wildcard (*) displays when the filter is not included in the selection.

The actual storage for the edit control's value is based on the type of the associated dictionary item (for example math numeric, string, character) and is distinct from the screen representation of the value. An edit control is affected by the properties of the associated dictionary item. For example, if an edit control is associated with a dictionary item of type string with a length of thirty, that edit control will not allow more than thirty characters to be typed into the field.

Properties and Options

The following properties and options are available for edit controls. These properties are available on all form types and for both database fields and data dictionary fields. Only those properties that affect processing are discussed here. Those properties that are exclusively interface-related are either not discussed, or are only discussed as they relate to control processing.

Disable

Controls with this property appear gray and cannot be changed by the user. You can use event rules to change the value of disabled fields. When a control is disabled, the associated text (static text to the left) is also disabled. Controls do not have this property by default.

Visible	Controls with this property can be seen. Without this property the control cannot be seen. You can use event rules to change the value of hidden controls.
Do not clear after add	This option applies only when a form is in add mode and the form is being cleared. Controls with this option retain their value. Controls do not have this option by default.
Required entry field	This option requires that a value is entered in the control before the record is saved (OK processing). This option does not accept either Null or blank. Controls do not have this option by default.
Default cursor on add/update mode	This option specifies where the cursor will be placed in add/update mode. If multiple controls have this property on, you cannot predict which control will actually have the cursor. Controls do not have this option by default.
No display if currency is Off	This option causes the control to be hidden when the Multicurrency Conversion is off. Controls do not have this option by default.

Control is Entered

1. Perform event rules: *Control is Entered*

On a Windows client, when a user presses the tab key or otherwise causes focus to move from one control to another, the control receiving focus actually gets the “Control is Entered” message before the control losing focus gets the “Control is Exited” message. Because this is a direct result of the sequence of Windows messages, it cannot be altered, although it is somewhat counterintuitive.

Control is Exited

1. If the focus is going to another window, stop processing (focus will return to this control when this window again receives focus)
2. Copy the text from the screen to the internal storage (conversion from string to the appropriate type happens during this step)
3. If this control is marked as a Required Entry Field and does not contain a value, then set the field in error.
4. Perform event rules: *Control is Exited*
5. If this is the first time this control has been exited, or if this control has changed in value since the last time the control was exited, then:

-
- Begin Control is Exited and Changed (*Inline*)

Control is Exited and Changed (Inline portion)

1. Perform event rules: *Control is Exited and Changed Inline*
2. If the form is not closing:
 - Attempt to execute *Control is Exited and Changed (Asynchronous)* on a thread.
3. If the form is closing or if the attempt to execute asynchronously was unsuccessful:
 - Begin *Control is Excited and Changed (Asynchronous)* as an inline function.

Control is Exited and Changed (Asynchronous portion)

1. Perform event rules: *Control is Exited and Changed Asynch*
2. If no errors were set during event rules processing
 - If this is not a filter field, or if it is an equal filter field (validation takes place only on filter fields with a comparison type of equal):
 - Perform Data Dictionary Validation
 - If there is an associated description for this control, populate it (if there were errors, this clears the associated description)
 - If there were no errors during validation, then copy the value to the control

J.D. Edwards Standards

All filter fields are marked with “Display Wildcard.” Only index fields are marked as filter fields (for performance). If reasonable, event rule should be put on *Control is Exited and Changed Asynch* instead of on one of the inline events (for performance).

Process Flow for Grid Control

Description

A grid control is similar to a spreadsheet on a form. Find/Browse, Search and Select, Header Detail, and Headerless Detail form types can contain grid controls. Grid controls contain columns. The columns are specified at design time. There are two types of columns. The first type is commonly referred to as a database column. It is associated with an item in the business view and through that connection to a dictionary item. Database columns represent a field in a database record. The second type of column is commonly referred to as a data dictionary column. Although only one type is called a data dictionary column, both types have a connection to a specific data dictionary item. The difference is that a database column has the additional connection to a business view field.

A grid can either be a browse grid or an update grid. You can use a browse grid is for viewing only and cells cannot be selected individually. The Find/Browse form and the Search and Select form have browse grids. You can use an update grid to add or update records. Cells in an update grid can be selected individually. The Header Detail form and the Headerless Detail form have update grids.

Grid controls can also have a Query By Example (QBE) line. The QBE columns have a one to one correspondence with the grid columns. You use a QBE value to change the selection criteria of a database fetch. Only database columns allow entry in the QBE columns because the purpose of the QBE is to affect the select and only database columns are in the business view. A QBE column can have one of the following comparison types:

- equal
- not equal
- less than
- less than or equal to
- greater than
- greater than or equal to

The comparison type is equal unless you specify otherwise. You can specify the comparison type in the QBE column or by using system functions. You can use wildcards (* or %) for an inexact search on a string field.

The values contained on a grid row act as a logical unit. You must validate a grid row prior to accepting a record, but validating the individual columns is not required. There is a parallel between edit control validation and grid row validation. There is not a parallel between edit control validation and grid column validation. The *Column is Exited* events are executed only if the user physically exits the cell. The Data Dictionary Validation for a cell executes when the cell is exited after a change or the first time the row is exited after it has been added. If you change a cell programmatically, then the *Row is Exited* events execute prior to accepting a record, but the column events and validation do not execute unless focus is physically set on the column.

The vendor spreadsheet stores the grid cell values as a tab delimited string (one per row). The values can be retrieved on a cell by cell basis or on a row by row basis. There is also internal storage for the grid columns in the interactive engine. The actual storage for the grid columns value is based on the type of the associated dictionary item (for example, math numeric, string, character) and is distinct from the screen representation of the value. Only one row of data can be acted upon at any given time. Each event executes in the context of a specific row.

A grid column is affected by the properties of the associated dictionary item. For example, if a grid column is associated with a dictionary item of type string with a length of 30 that grid column will not allow more than 30 characters to be typed into the cell.

Grid Columns, and Query By Example (QBE) lines are discussed in separate flow documents, but are an integrated portion of the grid. [Grid Column link](#), [QBE link](#).

Properties and Options

The following properties and options are available for grid controls. Except where noted these properties are available on all grids. Only those properties that affect processing are listed here. Those properties that are exclusively interface related are either not discussed, or are only discussed as they relate to grid processing. For a more detailed discussion of the interface properties reference the Forms Design Aid documentation.

Disable

Grids with this property appear grey and cannot be changed by the user. The application developer can change the value of disabled fields through ER. Grids do not have this property by default.

Visible

Grids with this property can be seen. Without this property the grid cannot be seen. The application developer can cause hidden grids to change value through ER.

Hide Query By Example	Grids with this property do not have a QBE line. QBE is neither available to the user nor the ER. Browse grids do not have this property by default. Update grids have this property by default.
Update Mode	This property is only available on update grids, but does not have any effect on the grid during runtime.
Multiple Select	this property allows for multiple grid rows to be selected at once. this can impact ER execution and deleting. Grids do not have this option by default.
Automatically Find On Entry	This option determines if grid records will be fetched when the form is opened. Grids without this option will open with no grid rows. Grids do not have this property by default.
Auto Find On Changes	This option determines if grid records will be fetched after a child form that has changed records closes. this option should only be used on forms where there are no modeless form interconnects. Grids do not have this property by default.
No Adds on Update Grid	This property applies to updates only. It determines if there will be an entry row in the grid. Without an entry row records cannot be added. With this property checked the only records that can be altered are existing records (records can be updated only). Grids do not have this option by default.
Disable Page-At-A-Time Processing	This option causes all available grid records to be fetched with find is pressed. Without this option only the first page of grid records is fetched until the user scrolls down to see additional records. Each time more records are requested, a page of data is returned. This provides a substantial performance benefit for large files and it is not recommended that this option be checked without careful consideration. Grids do not have this option by default.
Clear Grid After Add	There is currently no reference to this field during runtime. Presence or absence of this flag has no effect on the grid
Refresh Grid After Update	There is currently no reference to this field during runtime. Presence or absence of this flag has no effect on the grid

Process All Rows In Grid

This option causes each row in the grid to perform the three Row is Exited events on all the grid rows at least once prior to updating or adding the database records.

Set Focus on Grid

1. Perform event rules: *Set Focus on Grid*

Kill Focus on Grid

1. Perform event rules: *Kill Focus on Grid*

Row is Entered

1. Update the status bar with the current row number
2. If this row is not the entry row (last row on update grids):
 - Perform event rules: *Row is entered*

Row is Exited

1. If the form is not losing focus:
 - Perform event rules: *Row is exited*
 - If this is an update grid and the row has been changed since the last time the row was exited
 - Perform event rules: *Row is Exited and changed - Inline*
 - If you are not leaving the grid (exiting one row, entering another):
 - Attempt to execute Row is Exited and Changed (Asynchronous portion) on a thread.
 - If we are leaving the grid or the attempt to execute asynchronously was unsuccessful:
 - Begin Row is Exited and Changed (Asynchronous portion) as an inline function.

Row is Exited and Changed (Asynchronous portion)

1. Set a bitmap on the grid row header to indicate that this row is being processed.
2. Perform event rules: *Row is Exited and Changed - Asynch*
3. If this is the first time this row has been exited since it was added:

-
- Begin Row is Exited Validation

Row is Exited Validation

For each grid cell in this row that has not already performed data dictionary validation:

1. If this data base item is also in the header portion of the form the grid column will be populated from there, so skip this validation and go to the next grid cell. Note: Filter fields are only populated to the grid column if they are equal filters.
2. If the text contained in the cell can be stored (no alphas in numerics, no out or range data parameters):
 - Begin Data Dictionary Validation for this cell.
 - If there is an associated description column for this cell, populate it with the information returned from Data Dictionary Validation.

Cell is Exited after a change

On update grids when the contents of a cell have changed (via user keying or visual assist) and the cell has been exited:

1. Clear any errors set on this cell previous calls to Row is Exited Validation and Column is Exited events.
2. If the text contained in the cell can be stored (no alphas in numerics, no our of range date parameters):
 - Perform Data Dictionary Validation
 - If there is an associated description column for this cell, populate it with the information returned from Data Dictionary Validation.

Double-Click a Grid Row

On browse grids, double clicking on the grid row will cause the Select button to be pressed.

Key Pressed

On update grids when a key is pressed on the entry row (last row in the grid), it causes a new row to be added to the grid.

J.D. Edwards Standards

The event Row is Exited and Changed - Asynch is equivalent to validating the rows contents. it is often used for the "Edit Line" master business function. If

reasonable, ER should be put on Row is Exited and Changed - Asynch instead of on one of the inline events (performance).



Appendix C: Date Reference Scan

You can run date scan programs to search for date references in event rules or business functions. This may be helpful if you wish to verify possible Year 2000 issues.



Working with the Date Reference Scan

J.D. Edwards provides two date reference scans. You can use the

- Business Function Date Reference Scan
- Event Rule Date Reference Scan

Business Function Date Reference Scan

You can run a business function date reference scan to check business functions for date references. This scan also reports references to system date functions. You can then check the reports that are created during this process. J.D. Edwards provides the following OneWorld date reference scans.

- Business function date references scan
- Event rule date references scan

To use the business function date reference scan

1. On Object Management Workbench, check out the following objects.
 - B9000085
 - D9000085
 - R9000085

This step brings the specifications you need to your workstation.

2. Choose B9000085 and click the Design button.
3. Click the Design Tools tab, then click Build Business Function.
4. Choose the version you wish to run and click Select.
5. On Data Selection and Sequencing choose which options you wish to use and click Submit.
6. Check your PDF report and the text log file in the default print directory, which is usually the B7\PrintQueue directory.

When you run R9000085, it creates two reports. One report has the number of date references the scan program detected. The other report shows the date reference details.

Event Rule Date Reference Scan

You can use an event rule date scan to check event rules for date references. You can then check the reports that are created during this process.

The process to do this is the same as the business function scan except you use the following objects.

- B9000080
- D9000080
- R9000080

When you run R9000080 it creates two reports. One report has the number of date references the scan program detected.

The following illustration is an example of an event rule date reference report.

Syst Code	Obj Type	Object Name	Scan program Member Description	No of date references	Date Scan Result
00	APPL	P0000	System Setup	0	Date references found
00	APPL	P0001	Company/Business Unit Tree Structure	0	Date references found
00	APPL	P0002	Next Numbers	32	Date references found
H95	APPL	P00022	Next Unique Key Number	0	Date references found
H95	APPL	P0004A	User Defined Codes	6	Date references found
H95	APPL	P0004D	User Defined Code Alternate Descriptions	0	Date references found
88	APPL	P00050	Workstation Configuration	0	Date references found
H95	APPL	P00053	Host Configuration	0	Date references found
H95	APPL	P00055	User Defined Code Search and Select	0	Date references found
00	APPL	P0006	Business Units	4	Date references found
00	APPL	P00095	Translate Business Units	0	Date references found
09	APPL	P0009A	Organizational Structure	0	Date references found
00	APPL	P00095	Business Unit Search	0	Date references found
00	APPL	P00071	Work Day Calendar	77	Date references found
00	APPL	P00071W	Work Day Calendar Search & Select	0	Date references found
30	APPL	P000721	Work Day Calendar Transaction Revisions	6	Date references found
00	APPL	P0008	Fiscal Date Patterns	9	Date references found
00	APPL	P00095	Daylight Savings Rules	2	Date references found
00	APPL	P00098	Set 52 Period Dates	11	Date references found
00	APPL	P00091	Supplemental Data Setup	7	Date references found
00	APPL	P00092	Supplemental Data	36	Date references found
00	APPL	P00093	Flash Messages	3	Date references found
00	APPL	P0009M	CTI Interface	2	Date references found
00	APPL	P0010	Companies	106	Date references found

The other report shows the date reference details.

The following illustration is an example of an event rule date reference detail report.

R90000805ScanDetailReport(D98_09_01T1301) - Notepad

File Edit Search Help

EVENT RULES DATE REFERENCE SCAN REPORT(DETAILED)

Type	Object Name	Version	Form Name	CtrlId	Event	Line#
1	P0000		W00000	43	Selection Changed	1
1	P0000		W00000	1	Selection Changed	18
1	P0000		W00000	1	Selection Changed	21
1	P0000		W00000	0	Dialog is Initialized	6
1	P0000		W00000	1	Selection Changed	37
1	P0002		W0002H	0	Write Grid Line-Before	2
1	P0002		W0002H	0	Write Grid Line-Before	3
1	P0002		W0002H	0	Write Grid Line-Before	4
1	P0002		W0002H	0	Write Grid Line-Before	6
1	P0002		W0002H	0	Write Grid Line-Before	7
1	P0002		W0002H	0	Write Grid Line-Before	11
1	P0002		W0002H	0	Write Grid Line-Before	12
1	P0002		W0002H	0	Write Grid Line-Before	14
1	P0002		W0002H	1	Row Exit & Changed - Asynch	12
1	P0002		W0002H	1	Row Exit & Changed - Asynch	19
1	P0002		W0002H	1	Row Exit & Changed - Asynch	44
1	P0002		W0002H	1	Row Exit & Changed - Asynch	53
1	P0002		W0002H	42	Col Exited & Changed - Asynch	1
1	P0002		W0002H	42	Col Exited & Changed - Asynch	2
1	P0002		W0002H	42	Col Exited & Changed - Asynch	4
1	P0002		W0002H	42	Col Exited & Changed - Asynch	5
1	P0002		W0002H	42	Col Exited & Changed - Asynch	7
1	P0002		W0002H	42	Col Exited & Changed - Asynch	8
1	P0002		W0002H	42	Col Exited & Changed - Asynch	8
1	P0002		W0002H	42	Col Exited & Changed - Asynch	9
1	P0002		W0002H	42	Col Exited & Changed - Asynch	10
1	P0002		W0002H	42	Col Exited & Changed - Asynch	12
1	P0002		W0002H	42	Col Exited & Changed - Asynch	13
1	P0004A		W0004AH	1	Update Grid Rec to DB-Before	3
1	P0004A		W0004AH	12	Selection Changed	1
1	P0004A		W0004AH	1	Add Grid Rec to DB - Before	3

Glossary

Glossary

AAI. See automatic accounting instruction.

action message. With OneWorld, users can receive messages (system-generated or user-generated) that have shortcuts to OneWorld forms, applications, and appropriate data. For example, if the general ledger post sends an action error message to a user, that user can access the journal entry (or entries) in error directly from the message. This is a central feature of the OneWorld workflow strategy. Action messages can originate either from OneWorld or from a third-party e-mail system.

activator. In the Solution Explorer, a parent task with sequentially-arranged child tasks that are automated with a director.

ActiveX. A computing technology, based on object linking and embedding, that enables Java applet-style functionality for Web browsers as well as other applications. (Java is limited to Web browsers at this time.) The ActiveX equivalent of a Java applet is an ActiveX control. These controls bring computational, communications, and data manipulation power to programs that can “contain” them. For example, certain Web browsers, Microsoft Office programs, and anything developed with Visual Basic or Visual C++.

advance. A change in the status of a project in the Object Management Workbench. When you advance a project, the status change might trigger other actions and conditions such as moving objects from one server to another or preventing check-out of project objects.

alphanumeric character. A combination of letters, numbers, and symbols used to represent data. Contrast with numeric character and special character.

API. See application programming interface.

APPL. See application.

applet. A small application, such as a utility program or a limited-function spreadsheet. It is generally associated with the programming language Java, and in this context refers to

Internet-enabled applications that can be passed from a Web browser residing on a workstation.

application. In the computer industry, the same as an executable file. In OneWorld, an interactive or batch application is a DLL that contains programming for a set of related forms that can be run from a menu to perform a business task such as Accounts Payable and Sales Order Processing. Also known as system.

application developer. A programmer who develops OneWorld applications using the OneWorld toolset.

application programming interface (API). A software function call that can be made from a program to access functionality provided by another program.

application workspace. The area on a workstation display in which all related forms within an application appear.

audit trail. The detailed, verifiable history of a processed transaction. The history consists of the original documents, transaction entries, and posting of records, and usually concludes with a report.

automatic accounting instruction (AAI). A code that refers to an account in the chart of accounts. AAIs define rules for programs that automatically generate journal entries, including interfaces between Accounts Payable, Accounts Receivable, Financial Reporting, General Accounting systems. Each system that interfaces with the General Accounting system has AAIs. For example, AAIs can direct the General Ledger Post program to post a debit to a specific expense account and a credit to a specific accounts payable account.

batch header. The information that identifies and controls a batch of transactions or records.

batch job. A task or group of tasks you submit for processing that the system treats as a single unit during processing, for example, printing reports and purging files. The computer system

performs a batch job with little or no user interaction.

batch processing. A method by which the system selects jobs from the job queue, processes them, and sends output to the outqueue. Contrast with interactive processing.

batch server. A server on which OneWorld batch processing requests (also called UBEs) are run instead of on a client, an application server, or an enterprise server. A batch server typically does not contain a database nor does it run interactive applications.

batch type. A code assigned to a batch job that designates to which J.D. Edwards system the associated transactions pertain, thus controlling which records are selected for processing. For example, the Post General Journal program selects for posting only unposted transaction batches with a batch type of O.

batch-of-one immediate. A transaction method that allows a client application to perform work on a client workstation, then submit the work all at once to a server application for further processing. As a batch process is running on the server, the client application can continue performing other tasks. See also direct connect, store and forward.

BDA. See Business View Design Aid.

binary string (BSTR). A length prefixed string used by OLE automation data manipulation functions. Binary Strings are wide, double-byte (Unicode) strings on 32-bit Windows platforms.

Boolean Logic Operand. In J.D. Edwards reporting programs, the parameter of the Relationship field. The Boolean logic operand instructs the system to compare certain records or parameters. Available options are:

EQ	Equal To.
LT	Less Than.
LE	Less Than or Equal To.
GT	Greater Than.
GE	Greater Than or Equal To.
NE	Not Equal To.
NL	Not Less Than.
NG	Not Greater Than.

browser. A client application that translates information sent by the World Wide Web. A client must use a browser to receive, manipulate, and display World Wide Web

information on the desktop. Also known as a Web browser.

BSFN. See business function.

BSTR. See binary string.

BSVW. See business view.

business function. An encapsulated set of business rules and logic that can normally be reused by multiple applications. Business functions can execute a transaction or a subset of a transaction (check inventory, issue work orders, and so on). Business functions also contain the APIs that allow them to be called from a form, a database trigger, or a non-OneWorld application. Business functions can be combined with other business functions, forms, event rules, and other components to make up an application. Business functions can be created through event rules or third-generation languages, such as C. Examples of business functions include Credit Check and Item Availability.

business function event rule. See named event rule.

business view. Used by OneWorld applications to access data from database tables. A business view is a means for selecting specific columns from one or more tables whose data will be used in an application or report. It does not select specific rows and does not contain any physical data. It is strictly a view through which data can be handled.

Business View Design Aid (BDA). A OneWorld GUI tool for creating, modifying, copying, and printing business views. The tool uses a graphical user interface.

category code. In user defined codes, a temporary title for an undefined category. For example, if you are adding a code that designates different sales regions, you could change category code 4 to Sales Region, and define E (East), W (West), N (North), and S (South) as the valid codes. Sometimes referred to as reporting codes.

central objects. Objects that reside in a central location and consist of two parts: the central objects data source and central C components. The central objects data source contains OneWorld specifications, which are stored in a relational database. Central C components

contain business function source, header, object, library, and DLL files and are usually stored in directories on the deployment server. Together they make up central objects.

check-in location. The directory structure location for the package and its set of replicated objects. This is usually \\deploymentserver\release\path_code\package\packagename. The sub-directories under this path are where the central C components (source, include, object, library, and DLL file) for business functions are stored.

child. See parent/child form.

client/server. A relationship between processes running on separate machines. The server process is a provider of software services. The client is a consumer of those services. In essence, client/server provides a clean separation of function based on the idea of service. A server can service many clients at the same time and regulate their access to shared resources. There is a many-to-one relationship between clients and a server, respectively. Clients always initiate the dialog by requesting a service. Servers passively wait for requests from clients.

CNC. See configurable network computing.

component. In the ActivEra Portal, an encapsulated object that appears inside a workspace. Portal components

configurable client engine. Allows user flexibility at the interface level. Users can easily move columns, set tabs for different data views, and size grids according to their needs. The configurable client engine also enables the incorporation of Web browsers in addition to the Windows 95- and Windows NT-based interfaces.

configurable network computing. An application architecture that allows interactive and batch applications, composed of a single code base, to run across a TCP/IP network of multiple server platforms and SQL databases. The applications consist of reusable business functions and associated data that can be configured across the network dynamically. The overall objective for businesses is to provide a future-proof environment that enables them to change organizational structures, business

processes, and technologies independently of each other.

constants. Parameters or codes that you set and the system uses to standardize information processing by associated programs. Some examples of constants are: validating bills of material online and including fixed labor overhead in costing.

control. Any data entry point allowing the user to interact with an application. For example, check boxes, pull-down lists, hyper-buttons, entry fields, and similar features are controls.

core. The central and foundation systems of J.D. Edwards software, including General Accounting, Accounts Payable, Accounts Receivable, Address Book, Financial Reporting, Financial Modeling and Allocations, and Back Office.

CRP. Conference Room Pilot.

custom gridlines. A grid row that does not come from the database, for example, totals. To display a total in a grid, sum the values and insert a custom gridline to display the total. Use the system function Insert Grid Row Buffer to accomplish this.

data dictionary. The OneWorld method for storing and managing data item definitions and specifications. J.D. Edwards has an active data dictionary, which means it is accessed at runtime.

data mart. Department-level decision support databases. They usually draw their data from an enterprise data warehouse that serves as a source of consolidated and reconciled data from around the organization. Data marts can be either relational or multidimensional databases.

data replication. In a replicated environment, multiple copies of data are maintained on multiple machines. There must be a single source that "owns" the data. This ensures that the latest copy of data can be applied to a primary place and then replicated as appropriate. This is in contrast to a simple copying of data, where the copy is not maintained from a central location, but exists independently of the source.

data source. A specific instance of a database management system running on a computer.

Data source management is accomplished through Object Configuration Manager (OCM) and Object Map (OM).

data structure. A group of data items that can be used for passing information between objects, for example, between two forms, between forms and business functions, or between reports and business functions.

data warehouse. A database used for reconciling and consolidating data from multiple databases before it is distributed to data marts for department-level decision support queries and reports. The data warehouse is generally a large relational database residing on a dedicated server between operational databases and the data marts.

data warehousing. Essentially, data warehousing involves off-loading operational data sources to target databases that will be used exclusively for decision support (reports and queries). There are a range of decision support environments, including duplicated database, enhanced analysis databases, and enterprise data warehouses.

database. A continuously updated collection of all information a system uses and stores. Databases make it possible to create, store, index, and cross-reference information online.

database driver. Software that connects an application to a specific database management system.

database server. A server that stores data. A database server does not have OneWorld logic.

DCE. See distributed computing environment.

DD. See data dictionary.

default. A code, number, or parameter value that is assumed when none is specified.

detail. The specific pieces of information and data that make up a record or transaction. Contrast with summary.

detail area. A control that is found in OneWorld applications and functions similarly to a spreadsheet grid for viewing, adding, or updating many rows of data at one time.

direct connect. A transaction method in which a client application communicates interactively

and directly with a server application. See also batch-of-one immediate, store and forward.

director. An interactive utility that guides a user through the steps of a process to complete a task.

distributed computing environment (DCE). A set of integrated software services that allows software running on multiple computers to perform in a manner that is seamless and transparent to the end-users. DCE provides security, directory, time, remote procedure calls, and files across computers running on a network.

DLL. See dynamic link library.

DS. See data structure.

DSTR. See data structure.

duplicated database. A decision support database that contains a straightforward copy of operational data. The advantages involve improved performance for both operational and reporting environments. See also enhanced analysis database, enterprise data warehouse.

dynamic link library (DLL). A set of program modules that are designed to be invoked from executable files when the executable files are run, without having to be linked to the executable files. They typically contain commonly used functions.

dynamic partitioning. The ability to dynamically distribute logic or data to multiple tiers in a client/server architecture.

embedded event rule. An event rule that is specific to a particular table or application. Examples include form-to-form calls, hiding a field based on a processing option value, and calling a business function. Contrast with business function event rule. See also event rule.

employee work center. This is a central location for sending and receiving all OneWorld 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. With respect to workflow, the Message Center is MAPI compliant and supports drag and drop work reassignment, escalation, forward and reply, and workflow monitoring. All messages

from the message center can be viewed through OneWorld messages or Microsoft Exchange.

encapsulation. The ability to confine access to and manipulation of data within an object to the procedures that contribute to the definition of that object.

enhanced analysis database. A database containing a subset of operational data. The data on the enhanced analysis database performs calculations and provides summary data to speed generation of reports and query response times. This solution is appropriate when external data must be added to source data, or when historical data is necessary for trend analysis or regulatory reporting. See also duplicated database, enterprise data warehouse.

enterprise data warehouse. A complex solution that involves data from many areas of the enterprise. This environment requires a large relational database (the data warehouse) that is a central repository of enterprise data, which is clean, reconciled, and consolidated. From this repository, data marts retrieve data to provide department-level decisions. See also duplicated database, enhanced analysis database.

enterprise server. A database server and logic server. See database server. Also referred to as host.

ER. See event rule.

ERP. See enterprise resource planning.

event. An action that occurs when an interactive or batch application is running. Example events are tabbing out of an edit control, clicking a push button, initializing a form, or performing a page break on a report. The GUI operating system uses miniprograms to manage user activities within a form. Additional logic can be attached to these miniprograms and used to give greater functionality to any event within a OneWorld application or report using event rules.

event rule. Used to create complex business logic without the difficult syntax that comes with many programming languages. These logic statements can be attached to applications or database events and are executed when the defined event occurs, such as entering a form, selecting a menu bar option, page breaking on

a report, or selecting a record. An event rule can validate data, send a message to a user, call a business function, as well as many other actions. There are two types of event rules:

- 1 Embedded event rules.
- 2 Named event rules.

executable file. A computer program that can be run from the computer's operating system. Equivalent terms are "application" and "program."

exit. 1) To interrupt or leave a computer program by pressing a specific key or a sequence of keys. 2) An option or function key displayed on a form that allows you to access another form.

facility. 1) A separate 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. Sometimes referred to as a business unit. 2) In Home Builder and ECS, a facility is a collection of computer language statements or programs that provide a specialized function throughout a system or throughout all integrated systems. For example, DREAM Writer and FASTR are facilities.

FDA. See Form Design Aid.

find/browse. A type of form used to:

- 1 Search, view, and select multiple records in a detail area.
- 2 Delete records.
- 3 Exit to another form.
- 4 Serve as an entry point for most applications.

firewall. A set of technologies that allows an enterprise to test, filter, and route all incoming messages. Firewalls are used to keep an enterprise secure.

fix/inspect. A type of form used to view, add, or modify existing records. A fix/inspect form has no detail area.

form. An element of OneWorld's graphical user interface that contains controls by which a user can interact with an application. Forms allow the user to input, select, and view information. A OneWorld application might contain multiple forms. In Microsoft Windows terminology, a form is known as a dialog box.

Form Design Aid (FDA). The OneWorld GUI development tool for building interactive applications and forms.

form interconnection. Allows one form to access and pass data to another form. Form interconnections can be attached to any event; however, they are normally used when a button is clicked.

form type. The following form types are available in OneWorld:

- 1 Find/browse.
- 2 Fix/inspect.
- 3 Header detail.
- 4 Headerless detail.
- 5 Message.
- 6 Parent/child.
- 7 Search/select.

fourth generation language (4GL). A programming language that focuses on what you need to do and then determines how to do it. Structured Query Language is an example of a 4GL.

graphical user interface (GUI). A computer interface that is graphically based as opposed to being character-based. An example of a character-based interface is that of the AS/400. An example of a GUI is Microsoft Windows. Graphically based interfaces allow pictures and other graphic images to be used in order to give people clues on how to operate the computer.

grid. See detail area.

GUI. See graphical user interface.

header. Information at the beginning of a table or form. This information is used to identify or provide control information for the group of records that follows.

header/detail. A type of form used to add, modify, or delete records from two different tables. The tables usually have a parent/child relationship.

headerless detail. A type of form used to work with multiple records in a detail area. The detail area is capable of receiving input.

hidden selections. Menu selections you cannot see until you enter HS in a menu's Selection field. Although you cannot see these selections, they are available from any menu. They include such items as Display Submitted Jobs (33), Display User Job Queue (42), and

Display User Print Queue (43). The Hidden Selections window displays three categories of selections: user tools, operator tools, and programmer tools.

host. In the centralized computer model, a large timesharing computer system that terminals communicate with and rely on for processing. It contrasts with client/server in that those users work at computers that perform much of their own processing and access servers that provide services such as file management, security, and printer management.

HTML. See hypertext markup language.

hypertext markup language. A markup language used to specify the logical structure of a document rather than the physical layout. Specifying logical structure makes any HTML document platform independent. You can view an HTML document on any desktop capable of supporting a browser. HTML can include active links to other HTML documents anywhere on the Internet or on intranet sites.

index. Represents both an ordering of values and a uniqueness of values that provide efficient access to data in rows of a table. An index is made up of one or more columns in the table.

inheritance. The ability of a class to receive all or parts of the data and procedure definitions from a parent class. Inheritance enhances development through the reuse of classes and their related code.

install system code. See system code.

integrated toolset. Unique to OneWorld is an industrial-strength toolset embedded in the already comprehensive business applications. This toolset is the same toolset used by J.D. Edwards to build OneWorld interactive and batch applications. Much more than a development environment, however, the OneWorld integrated toolset handles reporting and other batch processes, change management, and basic data warehousing facilities.

interactive processing. Processing actions that occur in response to commands you enter directly into the system. During interactive processing, you are in direct communication with the system, and it might prompt you for additional information while processing your

request. See also online. Contrast with batch processing.

interface. A link between two or more computer systems that allows these systems to send information to and receive information from one another.

Internet. The worldwide constellation of servers, applications, and information available to a desktop client through a phone line or other type of remote access.

interoperability. The ability of different computer systems, networks, operating systems, and applications to work together and share information.

intranet. A small version of the Internet usually confined to one company or organization. An intranet uses the functionality of the Internet and places it at the disposal of a single enterprise.

IP. A connection-less communication protocol that by itself provides a datagram service. Datagrams are self-contained packets of information that are forwarded by routers based on their address and the routing table information contained in the routers. Every node on a TCP/IP network requires an address that identifies both a network and a local host or node on the network. In most cases the network administrator sets up these addresses when installing new workstations. In some cases, however, it is possible for a workstation, when booting up, to query a server for a dynamically assigned address.

IServer Service. Developed by J.D. Edwards, 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.

ISO 9000. A series of standards established by the International Organization for Standardization, designed as a measure of product and service quality.

J.D. Edwards Database. See JDEBASE Database Middleware.

Java. An Internet executable language that, like C, is designed to be highly portable across platforms. This programming language was developed by Sun Microsystems. Applets, or Java applications, can be accessed from a web browser and executed at the client, provided

that the operating system or browser is Java-enabled. (Java is often described as a scaled-down C++). Java applications are platform independent.

Java Database Connectivity (JDBC). The standard way to access Java databases, set by Sun Microsystems. This standard allows you to use any JDBC driver database.

JavaScript. A scripting language related to Java. Unlike Java, however, JavaScript is not an object-oriented language and it is not compiled.

jde.ini. J.D. Edwards file (or member for AS/400) that provides the runtime settings required for OneWorld initialization. Specific versions of the file/member must reside on every machine running OneWorld. This includes workstations and servers.

JDEBASE Database Middleware. J.D. Edwards proprietary database middleware package that provides two primary benefits:

1. Platform-independent APIs for multidatabase access. These APIs are used in two ways:
 - a. By the interactive and batch engines to dynamically generate platform-specific SQL, depending on the datasource request.
 - b. As open APIs for advanced C business function writing. These APIs are then used by the engines to dynamically generate platform-specific SQL.
2. Client-to-server and server-to-server database access. To accomplish this OneWorld is integrated with a variety of third-party database drivers, such as Client Access 400 and open database connectivity (ODBC).

JDECallObject. An application programming interface used by business functions to invoke other business functions.

JDENET. J.D. Edwards proprietary middleware software. JDENET is a messaging software package.

JDENET communications middleware. J.D. Edwards proprietary communications middleware package for OneWorld. It 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 OneWorld supported platforms.

job queue. A group of jobs waiting to be batch processed. See also batch processing.

just in time installation (JITI). OneWorld's method of dynamically replicating objects from the central object location to a workstation.

just in time replication (JITR). OneWorld's method of replicating data to individual workstations. OneWorld replicates new records (inserts) only at the time the user needs the data. Changes, deletes, and updates must be replicated using Pull Replication.

KEY. A column or combination of columns that identify one or more records in a database table.

leading zeros. A series of zeros that certain facilities in J.D. Edwards systems place in front of a value you enter. This normally occurs when you enter a value that is smaller than the specified length of the field. For example, if you enter 4567 in a field that accommodates eight numbers, the facility places four zeros in front of the four numbers you enter. The result appears as: 00004567.

level of detail. 1) The degree of difficulty of a menu in J.D. Edwards software. The levels of detail for menus are as follows:

- A Major Product Directories.
- B Product Groups.
- 1 Basic Operations.
- 2 Intermediate Operations.
- 3 Advanced Operations.
- 4 Computer Operations.
- 5 Programmers.
- 6 Advanced Programmers Also known as menu levels.

2) The degree to which account information in the General Accounting system is summarized. The highest level of detail is 1 (least detailed) and the lowest level of detail is 9 (most detailed).

MAPI. See Messaging Application Programming Interface.

master table. A database table used to store data and information that is permanent and necessary to the system's operation. Master tables might contain data such as paid tax

amounts, supplier names, addresses, employee information, and job information.

menu. A menu that displays numbered selections. Each of these selections represents a program or another menu. To access a selection from a menu, type the selection number and then press Enter.

menu levels. See level of detail.

menu masking. A security feature of J.D. Edwards systems that lets you prevent individual users from accessing specified menus or menu selections. The system does not display the menus or menu selections to unauthorized users.

Messaging Application Programming Interface (MAPI). An architecture that defines the components of a messaging system and how they behave. It also defines the interface between the messaging system and the components.

middleware. A general term that covers all the distributed software needed to support interactions between clients and servers. Think of it as the software that's in the middle of the client/server system or the "glue" that lets the client obtain a service from a server.

modal. A restrictive or limiting interaction created by a given condition of operation. Modal often describes a secondary window that restricts a user's interaction with other windows. A secondary window can be modal with respect to its primary window or to the entire system. A modal dialog box must be closed by the user before the application continues.

mode. In reference to forms in OneWorld, mode has two meanings:

- An operational qualifier that governs how the form interacts with tables and business views. OneWorld form modes are: add, copy, and update.
- An arbitrary setting that aids in organizing form generation for different environments. For example, you might set forms generated for a Windows environment to mode 1 and forms generated for a Web environment to mode 2.

modeless. Not restricting or limiting interaction. Modeless often describes a secondary window that does not restrict a user's interaction with

other windows. A modeless dialog box stays on the screen and is available for use at any time but also permits other user activities.

multitier architecture. A client/server architecture that allows multiple levels of processing. A tier defines the number of computers that can be used to complete some defined task.

named event rule. Encapsulated, reusable business logic created using through event rules rather than C programming. Contrast with embedded event rule. See also event rule.

NER. See named event rule.

network computer. As opposed to the personal computer, the network computer offers (in theory) lower cost of purchase and ownership and less complexity. Basically, it is a scaled-down PC (very little memory or disk space) that can be used to access network-based applications (Java applets, ActiveX controls) via a network browser.

network computing. Often referred to as the next phase of computing after client/server. While its exact definition remains obscure, it generally encompasses issues such as transparent access to computing resources, browser-style front-ends, platform independence, and other similar concepts.

next numbers. A feature you use to control the automatic numbering of such items as new G/L accounts, vouchers, and addresses. It lets you specify a numbering system and provides a method to increment numbers to reduce transposition and typing errors.

non-object librarian object. An object that is not managed by the object librarian.

numeric character. Digits 0 through 9 that are used to represent data. Contrast with alphanumeric characters.

object. A self-sufficient entity that contains data as well as the structures and functions used to manipulate the data. For OneWorld purposes, an object is a reusable entity that is based on software specifications created by the OneWorld toolset. See also object librarian.

object configuration manager (OCM). OneWorld's Object Request Broker and the control center for the runtime environment. It keeps track of the runtime locations for

business functions, data, and batch applications. When one of these objects is called, the Object Configuration Manager directs access to it using defaults and overrides for a given environment and user.

object embedding. When an object is embedded in another document, an association is maintained between the object and the application that created it; however, any changes made to the object are also only kept in the compound document. See also object linking.

object librarian. A repository of all versions, applications, and business functions reusable in building applications. You access these objects with the Object Management Workbench.

object librarian object. An object managed by the object librarian.

object linking. When an object is linked to another document, a reference is created with the file the object is stored in, as well as with the application that created it. When the object is modified, either from the compound document or directly through the file it is saved in, the change is reflected in that application as well as anywhere it has been linked. See also object embedding.

object linking and embedding (OLE). A way to integrate objects from diverse applications, such as graphics, charts, spreadsheets, text, or an audio clip from a sound program. See also object embedding, object linking.

object management workbench (OMW). An application that provides check-out and check-in capabilities for developers, and aids in the creation, modification, and use of OneWorld Objects. The OMW supports multiple environments (such as production and development).

object-based technology (OBT). A technology that supports some of the main principles of object-oriented technology: classes, polymorphism, inheritance, or encapsulation.

object-oriented technology (OOT). Brings software development past procedural programming into a world of reusable programming that simplifies development of applications. Object orientation is based on the following principles: classes, polymorphism, inheritance, and encapsulation.

OCM. See object configuration manager.

ODBC. See open database connectivity.

OLE. See object linking and embedding.

OMW. Object Management Workbench.

OneWorld. A combined suite of comprehensive, mission-critical business applications and an embedded toolset for configuring those applications to unique business and technology requirements. OneWorld is built on the Configurable Network Computing technology- J.D. Edwards' own application architecture, which extends client/server functionality to new levels of configurability, adaptability, and stability.

OneWorld application. Interactive or batch processes that execute the business functionality of OneWorld. They consist of reusable business functions and associated data that are platform independent and can be dynamically configured across a TCP/IP network.

OneWorld object. A reusable piece of code that is used to build applications. Object types include tables, forms, business functions, data dictionary items, batch processes, business views, event rules, versions, data structures, and media objects. See also object.

OneWorld process. Allows OneWorld clients and servers to handle processing requests and execute transactions. A client runs one process, and servers can have multiple instances. OneWorld 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.

OneWorld Web development computer. A standard OneWorld Windows developer computer with the additional components installed:

- JFC (0.5.1).
- Generator Package with Generator.Java and JDECOM.dll.
- R2 with interpretive and application controls/form.

online. Computer functions over which the system has continuous control. Users are online with the system when working with J.D. Edwards system provided forms.

open database connectivity (ODBC). Defines a standard interface for different technologies to process data between applications and different data sources. The ODBC interface is made up of a set of function calls, methods of connectivity, and representation of data types that define access to data sources.

open systems interconnection (OSI). The OSI model was developed by the International Standards Organization (ISO) in the early 1980s. It defines protocols and standards for the interconnection of computers and network equipment.

operand. See Boolean Logic Operand.

output. Information that the computer transfers from internal storage to an external device, such as a printer or a computer form.

output queue. See print queue.

package. OneWorld 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 install program can find them. It is a point-in-time "snap shot" of the central objects on the deployment server.

package location. The directory structure location for the package and its set of replicated objects. This is usually \\deployment server\release\path_code\package\ package name. The sub-directories under this path are where the replicated objects for the package will be placed. This is also referred to as where the package is built or stored.

parameter. A number, code, or character string you specify in association with a command or program. The computer uses parameters as additional input or to control the actions of the command or program.

parent/child form. A type of form that presents parent/child relationships in an application on one form. The left portion of the form presents a tree view that displays a visual representation of a parent/child relationship. The right portion of the form displays a detail area in browse mode. The detail area displays the records for the child item in the tree. The parent/child form supports drag and drop functionality.

partitioning. A technique for distributing data to local and remote sites to place data closer to the users who access. Portions of data can be copied to different database management systems.

path code. A pointer to a specific set of objects. A path code is used to locate:

1. Central Objects.
2. Replicated Objects.

platform independence. A benefit of open systems and Configurable Network Computing. Applications that are composed of a single code base can be run across a TCP/IP network consisting of various server platforms and SQL databases.

polymorphism. A principle of object-oriented technology in which a single mnemonic name can be used to perform similar operations on software objects of different types.

portability. Allows the same application to run on different operating systems and hardware platforms.

portal. A configurable Web object that provides information and links to the Web. Portals can be used as home pages and are typically used in conjunction with a Web browser.

primary key. A column or combination of columns that uniquely identifies each row in a table.

print queue. A list of tables, such as reports, that you have submitted to be written to an output device, such as a printer. The computer spools the tables until it writes them. After the computer writes the table, the system removes the table identifier from the list.

processing option. A feature of the J.D. Edwards reporting system that allows you to supply parameters to direct the functions of a program. For example, processing options allow you to specify defaults for certain form displays, control the format in which information prints on reports, change how a form displays information, and enter beginning dates.

program temporary fix (PTF). A representation of changes to J.D. Edwards software that your organization receives on magnetic tapes or diskettes.

project. An Object Management Workbench object used to organize objects in development.

published table. Also called a “Master” table, this is the central copy to be replicated to other machines. Resides on the “Publisher” machine. the Data Replication Publisher Table (F98DRPUB) identifies all of the Published Tables and their associated Publishers in the enterprise.

publisher. The server that is responsible for the Published Table. The Data Replication Publisher Table (F98DRPUB) identifies all of the Published Tables and their associated Publishers in the enterprise.

pull replication. One of the OneWorld methods for replicating data to individual workstations. Such machines are set up as Pull Subscribers using OneWorld’s data replication tools. The only time 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 Data Replication Pending Change Notification table (F98DRPCN).

purge. The process of removing records or data from a system table.

QBE. See query by example.

query by example (QBE). Located at the top of a detail area, it is used to search for data to be displayed in the detail area.

redundancy. Storing exact copies of data in multiple databases.

regenerable. Source code for OneWorld business functions can be regenerated from specifications (business function names). Regeneration occurs whenever an application is recompiled, either for a new platform or when new functionality is added.

relationship. Links tables together and facilitates joining business views for use in an application or report. Relationships are created based on indexes.

release/release update. A “release” contains major new functionality, and a “release update” contains an accumulation of fixes and performance enhancements, but no new functionality.

replicated object. A copy or replicated set of the central objects must reside on each client

and server that run OneWorld. The path code indicates the directory the directory where these objects are located.

run. To cause the computer system to perform a routine, process a batch of transactions, or carry out computer program instructions.

SAR. See software action request.

scalability. Allows software, architecture, network, or hardware growth that will support software as it grows in size or resource requirements. The ability to reach higher levels of performance by adding microprocessors.

search/select. A type of form used to search for a value and return it to the calling field.

selection. Found on J.D. Edwards menus, selections represent functions that you can access from a menu. To make a selection, type the associated number in the Selection field and press Enter.

server. Provides the essential functions for furnishings services to network users (or clients) and provides management functions for network administrators. Some of these functions are storage of user programs and data and management functions for the file systems. It may not be possible for one server to support all users with the required services. Some examples of dedicated servers that handle specific tasks are backup and archive servers, application and database servers.

servlet. Servlets provide a Java-based solution used to address the problems currently associated with doing server-side programming, including inextensible scripting solutions. Servlets are objects that conform to a specific interface that can be plugged into a Java-based server. Servlets are to the server-side what applets are to the client-side.

software. The operating system and application programs that tell the computer how and what tasks to perform.

software action request (SAR). An entry in the AS/400 database used for requesting modifications to J.D. Edwards software.

special character. A symbol used to represent data. Some examples are *, &, #, and /. Contrast with alphanumeric character and numeric character.

specifications. A complete description of a OneWorld object. Each object has its own specification, or name, which is used to build applications.

Specs. See specifications.

spool. The function by which the system stores generated output to await printing and processing.

spooled table. A holding file for output data waiting to be printed or input data waiting to be processed.

SQL. See structured query language.

static text. Short, descriptive text that appears next to a control variable or field. When the variable or field is enabled, the static text is black; when the variable or field is disabled, the static text is gray.

store and forward. A transaction method that allows a client application to perform work and, at a later time, complete that work by connecting to a server application. This often involves uploading data residing on a client to a server.

structured query language (SQL). A fourth generation language used as an industry standard for relational database access. It can be used to create databases and to retrieve, add, modify, or delete data from databases. SQL is not a complete programming language because it does not contain control flow logic.

subfile. See detail.

submit. See run.

subscriber. The server that is responsible for the replicated copy of a Published Table. Such servers are identified in the Subscriber Table.

subscriber table. The Subscriber Table (F98DRSUB), which is stored on the Publisher Server with the Data Replication Publisher Table (F98DRPUB) identifies all of the Subscriber machines for each Published Table.

subsystem job. Within OneWorld, subsystem jobs are batch processes that continually run independently of, but asynchronously with, OneWorld applications.

summary. The presentation of data or information in a cumulative or totaled manner in which most of the details have been

removed. Many of the J.D. Edwards systems offer forms and reports that are summaries of the information stored in certain tables. Contrast with detail.

system. See application.

System Code. System codes are a numerical representation of J.D. Edwards and customer systems. For example, 01 is the system code for Address Book. System codes 55 through 59 are reserved for customer development by customers. Use system codes to categorize within OneWorld. For example, when establishing user defined codes (UDCs), you must include the system code the best categorizes it. When naming objects such as applications, tables, and menus, the second and third characters in the object's name is the system code for that object. For example, G04 is the main menu for Accounts Payable, and 04 is its system code.

system function. A program module, provided by OneWorld, available to applications and reports for further processing.

table. A two-dimensional entity made up of rows and columns. All physical data in a database are stored in tables. A row in a table contains a record of related information. An example would be a record in an Employee table containing the Name, Address, Phone Number, Age, and Salary of an employee. Name is an example of a column in the employee table.

table design aid (TDA). A OneWorld GUI tool for creating, modifying, copying, and printing database tables.

table event rules. Use table event rules to attach database triggers (or programs) that automatically run whenever an action occurs against the table. An action against a table is referred to as an event. When you create a OneWorld database trigger, you must first determine which event will activate the trigger. Then, use Event Rules Design to create the trigger. Although OneWorld allows event rules to be attached to application events, this functionality is application specific. Table event rules provide embedded logic at the table level.

TAM. Table Access Management.

TBLE. See table.

TC. Table conversion.

TCP/IP. Transmission Control Protocol/Internet Protocol. The original TCP protocol was developed as a way to interconnect networks using many different types of transmission methods. TCP provides a way to establish a connection between end systems for the reliable delivery of messages and data.

TCP/IP services port. Used by a particular server application to provide whatever service the server is designed to provide. The port number must be readily known so that an application programmer can request it by name.

TDA. See table design aid.

TER. See table event rules.

Terminal Identification. The workstation ID number. Terminal number of a specific terminal or IBM user ID of a particular person for whom this is a valid profile. Header Field: Use the Skip to Terminal/User ID field in the upper portion of the form as an inquiry field in which you can enter the number of a terminal or the IBM user ID of a specific person whose profile you want the system to display at the top of the list. When you first access this form, the system automatically enters the user ID of the person signed on to the system. Detail Field: The Terminal/User ID field in the lower portion of the form contains the user ID of the person whose profile appears on the same line. A code identifying the user or terminal for which you accessed this window.

third generation language (3GL). A programming language that requires detailed information about how to complete a task. Examples of 3GLs are COBOL, C, Pascal and FORTRAN.

token. A referent to an object used to determine ownership of that object and to prevent non-owners from checking the object out in Object Management Workbench. An object holds its own token until the object is checked out, at which time the object passes its token to the project in which the object is placed.

trigger. Allow you to attach default processing to a data item in the data dictionary. When that data item is used on an application or report, the trigger is invoked by an event associated with the data item. OneWorld also has three

visual assist triggers: calculator, calendar and search form.

UBE. Universal batch engine.

UDC Edit Control. Use a User-Defined Code (UDC) Edit Control for a field that accepts only specific values defined in a UDC table.

Associate a UDC edit control with a database item or dictionary item. The visual assist Flashlight automatically appears adjacent to the UDC edit control field. When you click on the visual assist Flashlight, the attached search and select form displays valid values for the field.

To create a UDC Edit Control, you must:

- Associate the data item with a specific UDC table in the Data Dictionary.
- Create a search and select form for displaying valid values from the UDC table.

uniform resource identifier (URI). A character string that references an internet object by name or location. A URL is a type of URI.

uniform resource locator (URL). Names the address (location) of a document on the Internet or an intranet. A URL includes the document's protocol and server name. The path to the document might be included as well. The following is an example of a URL:
<http://www.jdedwards.com>. This is J.D. Edwards Internet address.

URI. See uniform resource identifier.

URL. See uniform resource locator.

user defined code (type). The identifier for a table of codes with a meaning you define for the system, such as ST for the Search Type codes table in Address Book. J.D. Edwards systems provide a number of these tables and allow you to create and define tables of your own. User defined codes were formerly known as descriptive titles.

user defined codes (UDC). Codes within software that users can define, relate to code descriptions, and assign valid values. Sometimes user defined codes are referred to as a generic code table. Examples of such codes are unit-of-measure codes, state names, and employee type codes.

UTB. Universal Table Browser.

valid codes. The allowed codes, amounts, or types of data that you can enter in a field. The system verifies the information you enter against the list of valid codes.

visual assist. Forms that can be invoked from a control to assist the user in determining what data belongs in the control.

vocabulary overrides. A feature you can use to override field, row, or column title text on forms and reports.

wchar_t. Internal type of a wide character. Used for writing portable programs for international markets.

web client. Any workstation that contains an internet browser. The web client communicates with the web server for OneWorld data.

web server. Any workstation that contains the IServer service, SQL server, Java menus and applications, and Internet middleware. The web server receives data from the web client, and passes the request to the enterprise server. When the enterprise server processes the information, it sends it back to the web server, and the web server sends it back to the web client.

WF. See workflow.

window. See form.

workflow. According to the Workflow Management Coalition, workflow means "the automation of a business process, in whole or part, during which documents, information, or tasks are passed from one participant to another for action, according to a set of procedural rules."

workgroup server. A remote database server usually containing subsets of data replicated from a master database server. This server does not performance an application or batch processing. It may or may not have OneWorld running (in order to replicate data).

workspace. In the ActivEra Portal, the main section of the Portal. A user might have access to several workspaces, each one configured differently and containing its own components.

worldwide web. A part of the Internet that can transmit text, graphics, audio, and video. The

World Wide Web allows clients to launch local or remote applications.

z file. For store and forward (network disconnected) user, OneWorld store and forward applications perform edits on static data and other critical information that must be valid to process an order. After the initial edits are complete, OneWorld stores the transactions in work tables on the workstation. These work table are called Z files. When a network connection is established, Z files are uploaded to the enterprise server and the transactions are edited again by a master business function. The master business function will then update the records in your transaction files.

Index

Index

A

Access32, 25–6

Action messages, 22–6

Add

 menu/toolbar items

 Find/Browse form, B–6

 Parent/Child form, B–13

 projects to a project, 3–21

 user to project, 3–43

Add a processing option template with translated text, 15–15

Add button, processing, 12–25

Add entry row to grid

 Header Detail form, B–22

 Headerless Detail form, B–29

Add Object form, 3–24

Adding

 bitmap strip, 10–36

 business function, object codes, 13–32

 business functions

 codes, 13–32

 related functions, 13–33

 related tables and functions, 13–32

 comment lines to event rules, 12–36

 language specific media object

 attachment, 10–120

Adding a business view, 6–5

 J.D. Edwards naming standards, 6–6

 joined views, 6–7

Adding a table, 5–3

 indices, 5–5

Adding an application to a menu, 19–14

Adding an interactive message data item, 21–19

Adding an interactive version, 9–6

Adding an object to an OMW project, 3–27

Adding glossary text for languages, 4–41

Adding multiple objects to an OMW project, 3–28

Adding or changing a menu selection, 19–11

Adding or changing web address on OneWorld Explorer Help, 19–21

Adding same-named objects to a project, 3–35

Adding tips of the day to an object, 20–6

Additional features, record locking, 17–1

Advance, project, project status, 3–20

Advanced functions, expression manager, 12–89

 date functions, 12–89

 general functions, 12–89

 text functions, 12–90

 trig functions, 12–90

Advanced get, 3–31

Advantages, currency advantages, 18–1

Alias, used in an RPG program, 4–13

Alias for an external data dictionary item, 4–14

Aligning

 alignment grid to align controls, 10–30

 controls, 10–29

 group, 10–29

All grid recs deleted from DB, runtime processing, 12–28

Allowed (User) Actions, 3–4

APIs

 calling JDECACHE APIs, 14–7

 JDECACHE cursors, 14–17

 cursor-advancing APIs, 14–18

 non-cursor-advancing APIs, 14–18

 JDECACHE manipulation, 14–6

 jdeCacheFetchPosition, 14–19

 jdeCacheFetchPositionByRef, 14–20

Application

 currency, 18–4

 debugging an application, 23–9

 inspecting or modifying a variable, 23–10

 just in time debugging, 23–11

 event rules, 12–4

 JDECACHE, 14–3

Application design, 9–3

 batch applications, 9–4

 interactive applications, 9–3

- web applications, 9–4
- Application event rules, 1–11
- Application Programming Interfaces, 13–21
- Application specific parameters
 - begin document, 13–60
 - edit document, 13–63
 - end document, 13–64
- Application text modification rules, A–10
- Applications, 1–4
 - add a OneWorld application to a menu, 19–15
 - add a Windows application to a menu, 19–19
 - add a WorldVision application to a menu, 19–18
 - adding an application to a menu, 19–14
 - batch applications, 9–4
 - building an application, 1–5
 - change row and column text, 4–46
 - interactive applications, 9–3
 - messaging, 21–1
 - web applications, 9–4
- Assign, value, 12–85
- Assignment, creating, 12–85
 - expression, 12–87
- Assignment display, expression manager, 12–88
- Associating
 - an item or description with a control, 10–90
 - database items, 10–90
 - database items, dictionary items, or descriptions, 10–89
 - description or item with a control, 10–90
 - descriptions, 10–90
 - dictionary items, 10–90
- Asynchronous, business functions, 12–114
- Asynchronous events, how threaded events work, 12–112
- Asynchronous processing, 12–111
- Attaching
 - business function to an event, 12–65
 - event rules, Menu/Toolbar Exit, 10–39
 - functions, 12–57
 - system function to an event, 12–57
- Attaching a data structure template to a message, 21–22
- Attaching a default value trigger, 4–23
- Attaching a display rule trigger, 4–29

- Attaching a next number trigger, 4–35
 - next numbers, 4–35
- Attaching a smart field trigger, 4–36
- Attaching a user defined codes trigger, 4–31
- Attaching a visual assist trigger, 4–24
- Attaching an edit rule trigger, 4–26
- Attaching an interactive message data item, 21–23
- Attaching default triggers, 4–21, 4–22
- Attachment, adding, language specific media object, 10–120
- Automatic error settings, 21–7
- Automatic Line Numbering, 12–55
- Available business functions, business function builder form, 13–38
- Available information, expression manager, 12–88
- Available objects, processing, 12–10
- Available objects and runtime data structures, 12–7
- Available objects tab, 7–4
- Available operations, Table I/O, 12–91

B

- Balance table opens and closes, business function performance, 25–15
- Batch application event rules variables, 12–50
- Batch application performance, 25–12
 - setting up level breaks, 25–12
 - slow performance, 25–12
- Batch applications, 9–4
 - debugging business functions, 23–16
 - searching for batch applications, 8–6
- Batch error messages, 22–1, 22–3
 - action messages, 22–6
 - creating a level-break message, 22–9
 - first-level messages, 22–3
 - level break messages, 22–6
 - level-break messages, 22–3
 - second-level messages, 22–4
 - text substitution error messages, 22–5
 - third-level messages, 22–5
 - work center API, 22–7
- BC assigned database values, runtime processing, 12–17

- Begin DLL section, build output, 13-43
- Begin document
 - application specific parameters, 13-60
 - common parameters, 13-59, 13-61
 - function name, 13-58, 13-60
 - hook up tips, 13-58
 - naming, transaction master business function modules, 13-58
 - special logic/processing required, 13-58, 13-61
 - typical users and hook up, 13-61
 - what does it do, 13-58
- Bitmap controls, creating, 10-65
- Bitmap strip, adding, 10-36
- Bitmap Strips, Adding, 10-36
- Breakpoint manager, 23-7
- Breakpoints, setting breakpoints, 23-11, 23-20
- BrowsER, 12-117
 - expanded node, 12-121
 - filter ER records, 12-122
 - options, working with, 12-120
 - search, 12-122
 - show object ID, 12-121
 - working with, 12-117
- Build all, performing, 13-45
- Build an application, 1-5
- Build options, setting, 13-40
- Build output
 - begin DLL section, 13-43
 - business function builder form, 13-38
 - compile section, 13-43
 - link section, 13-44
 - makefile section, 13-43
 - reading, 13-42
 - rebase section, 13-44
 - summary section, 13-45
- Build triggers option, 18-3
- Building
 - single business function, 13-39
 - transaction master business functions, 13-68
- Business function
 - cache business function description, 14-35
 - cache business function source description, 14-35
 - cache business function source name, 14-35
 - creating, documentation, 13-77, 13-78
 - documentation, 13-77
 - generating, 13-81
 - template, 13-79
 - viewing, 13-85
 - event, attaching, 12-65
 - event rules, 12-3
 - generating, documentation, 13-81
 - level-break messages, 22-13
 - modification rules, A-18
 - viewing, documentation, 13-85
 - Business function - values to pass, documentation, viewing, 13-86
 - Business Function Builder, DLLs, 13-4
 - Business function builder
 - resolving errors, 13-52
 - working with, 13-37
 - Business function builder form
 - available business functions, 13-38
 - Build output, 13-38
 - functions list, 13-38
 - understanding, 13-37
 - Business function data reference scan, C-3
 - Business function data structure, 7-10
 - Business function data structures, 7-2
 - interconnection data structures, 7-3
 - Business function design, 1-10
 - Business Function Design form, 13-36
 - Business function document viewer, documentation, viewing, 13-87
 - Business function event rules, 1-11
 - creating, 13-7
 - creating or editing event rules, 13-8
 - Business Function Object Codes, adding, 13-32
 - Business function objects, linking, 13-40
 - Business function performance, 25-14
 - balance table opens and closes, 25-15
 - memory allocation, 25-15
 - Business function search, documentation, viewing, 13-86
 - Business function source files, structure, 13-18
 - Business Functions, building all, 13-45
 - Business functions
 - adding
 - codes, 13-32
 - object codes, 13-32
 - related functions, 13-33
 - related tables and functions, 13-32
 - asynchronous, 12-114

- building a single, 13–39
- changing, parent DLL, 13–33
- components, 13–3
- creating a level-break business function, 22–15
- debugging, 13–33
- extending a transaction boundary using business functions, 16–13
- features, 13–2
- OneWorld tools, 13–1
- pessimistic record locking, 17–5
- searching for business functions, 8–7
- transaction processing, 16–5
- transaction processing in remote business functions, 16–6
- using the same cache, 14–14
- working with, 13–31

Business view

- adding a business view, 6–5
 - J.D. Edwards naming standards, 6–6
 - joined views, 6–7
- choosing a table, 6–11
- choosing data items, 6–12
- creating a table join, 6–17
- creating a table union, 6–19
- searching for business views, 8–8
- using select distinct, 6–14

Business view design, 1–9, 6–1

- indices, 6–3
- select distinct, 6–2
- table join, 6–1
 - left outer joins, 6–2
 - right outer joins, 6–2
 - simple joins, 6–2
- table union, 6–2

Business view performance, 25–9

- performance tips
 - joined business view versus table I/O, 25–10
 - restrict the number of fields, 25–9
 - single-table or multiple-table, 25–9
 - table I/O versus joined business view, 25–10
 - unions, 25–9

Business views

- modification rules, A–13
- selecting, 10–21

Button clicked, runtime processing, 12–24

Buttons, Message form, B–39

C

C business functions, understanding, 13–13

Cache

- structure, 13–66
 - detail, 13–67
 - header, 13–66
- using the same cache in multiple business functions or forms, 14–14

Cache action code standards, 14–37

- CACHE_ADD, 14–37
- CACHE_ADD_UPDATE, 14–38
- CACHE_CLOSE_CURSOR, 14–39
- CACHE_DELETE, 14–38
- CACHE_GET, 14–37
- CACHE_GET_NEXT, 14–39
- CACHE_TERMINATE, 14–39
- CACHE_TERMINATE_ALL, 14–39
- CACHE_UPDATE, 14–38

Cache business function description, 14–35

Cache business function source name, 14–35

Cache data structure

- group cache function, 14–37
- individual cache function, 14–37

Cache name, 14–36

Cache programming standards, 14–36

- cache action code standards, 14–37
- cache name, 14–36
- data structure standard data items, 14–37
- defining the cache data structure, 14–37
- general standards, 14–36
- group cache business function header file, 14–39
- terminating versus clearing cache, 14–36

- CACHE_ADD, 14–37
- CACHE_ADD_UPDATE, 14–38
- CACHE_CLOSE_CURSOR, 14–39
- CACHE_DELETE, 14–38
- CACHE_GET, 14–37
- CACHE_GET_NEXT, 14–39
- CACHE_TERMINATE, 14–39
- CACHE_TERMINATE_ALL, 14–39
- CACHE_UPDATE, 14–38

Caching, 14–1

- calling JDECACHE APIs, 14–7
- initializing the cache, 14–10

- JDECACHE, 14–7
 - performance considerations, 14–4
- JDECACHE API set, 14–4
- JDECACHE APIs, 14–1
- JDECACHE cursors, 14–15
- JDECACHE errors, 14–23
- JDECACHE example program, 14–25
- opening JDECACHE cursors, 14–15
- tables, 14–1
- Calling the processing work center API, 22–21
- Calling the work center initialization API, 22–18
- cAllowUserIdToChange parameter, 22–20
- Cancel, menu/toolbar items
 - Fix/Inspect form, B–18
 - Header Detail form, B–25
 - Headerless Detail form, B–32
- Cancel document (optional)
 - common parameters, 13–65
 - function name, 13–65
 - naming, transaction master business function modules, 13–65
 - special logic/processing required, 13–65
 - what does it do, 13–65
- Cell is exited after change, process flow for grid control, B–49
- Central object data dictionary, storing data dictionary and data dictionary items, 4–4
- Changing
 - business function, parent DLL, 13–33
 - size, forms, grids, and controls, 10–26
 - tab sequence, 10–91
- Changing menu selection text, 19–30
- Changing menu text for languages, 19–28
- Changing object properties, 3–34
- Changing the release level of an object, 3–35
- Check boxes, creating, 10–46
- Checking objects in, 3–33
- Checking objects out, 3–33
- Child nodes, 10–83
- Choosing a data item for a business view, 6–12
- Choosing a table for a business view, 6–11
- Clear cache
 - function name, 13–64
 - naming, transaction master business function modules, 13–64
 - special logic/processing required, 13–65
 - what does it do, 13–65
- Clearing dialog
 - Fix/Inspect form, B–16
 - Header Detail form, B–22
 - Headerless Detail form, B–29
 - Search/Select form, B–37
- Close, menu/toolbar items
 - Find/Browse form, B–5
 - Parent/Child form, B–12
 - Search/Select form, B–37
- Closing a cursor, 14–20
- Closing form
 - Fix/Inspect form, B–17, B–37
 - Form/Browse form, B–5
 - Header Detail form, B–22
 - Headerless Detail form, B–29
 - Message form, B–39
 - Parent/Child form, B–12
- Code, interpretive and compiled code, 23–3
- Coexistence
 - data item used in RPG program, 4–13
 - indices and logicals, 5–5, 5–12
- Column, exited and changed - asynch, 12–113
- Column properties, 5–24
- Combo boxes, creating, 10–67
- Comment, 15–10
 - entering a comment for a processing option, 15–10
- Comment lines, event rules, adding, 12–36
- Commit, 16–2
 - two-phase commit (manual commit mode), 16–2
- Commits and rollbacks, 16–1
 - commit, 16–2
 - rollback, 16–2
- Common parameters
 - begin document, 13–59, 13–61
 - cancel document (optional), 13–65
 - edit document, 13–63
 - end document, 13–64
- Compile section, build output, 13–43
- Components, business functions, 13–3
- Concepts, allowed (user) actions, 3–4
- Connecting, database, 13–26
- Considerations for coexisting with World software, 25–6
- Control
 - associating an item or description, 10–90
 - exited and changed - asynch, 12–113

- exited processing, 12-10
- Control and static text, moving, 10-26
- Control is entered, process flow for edit control, B-42
- Control is exited, process flow for edit control, B-42
- Control is exited and changed - asynchronous, process flow for edit control, B-43
- Control is exited and changed - inline, process flow for edit control, B-43
- Control is exited and changed asynchronous, Edit control, B-43
- Control Modes, 10-95
- Control tables modification rules, A-12
- Controls
 - aligning, 10-29
 - group, 10-29
 - alignment grid to align, 10-30
 - bitmap, creating, 10-65
 - changing size, 10-26
 - creating, UDC edit, 10-58
 - cut, copy, and paste, 10-29
 - disconnecting, static text, 10-59
 - media object, creating, 10-60, 10-62
 - moving, 10-25
 - paste, cut, and copy, 10-29
 - selecting, 10-25
 - size and place using the size command, 10-27
 - size using a mouse, 10-27
 - tree, 10-67
 - understanding, 12-2
 - working with, 10-43
- Copy, menu/toolbar items
 - Find/Browse form, B-6
 - Parent/Child form, B-13
- Copying, event rules, 12-34
- Copying a menu selection, 19-27
- Copying tables, 5-16
- Create, project, 3-17
- Create multi-level error messaging, 21-11
- Creating
 - assignment, 12-85
 - bitmap controls, 10-65
 - business function documentation, 13-77, 13-78
 - business function event rules, 13-7
 - check boxes, 10-46
 - combo boxes, 10-67
 - data structure documentation, 13-79
 - edit controls, 10-53
 - event rule variables, 12-50
 - expression for an assignment, 12-87
 - form interconnections, 12-71
 - forms, 10-17
 - group boxes, 10-69
 - if and while statements, 12-44
 - master business function, processing options, 13-56
 - media object controls, 10-60, 10-62
 - Menu/Toolbar Exits, 10-33
 - modal form interconnect, 12-71
 - modeless form interconnect, 12-75
 - parameter documentation, 13-81
 - push buttons, 10-45
 - radio buttons, 10-48
 - report interconnections, 12-81
 - static text controls, 10-52
 - subcategories in menus, 10-40
 - table event rules, 12-104
 - Table I/O event rule, 12-92
 - text boxes, 10-68
 - UDC edit controls, 10-58
- Creating a business function data structure, 7-10
- Creating a currency conversion trigger, 18-8
- Creating a data dictionary item for a level-break message, 22-9
- Creating a data structure for the data dictionary item, 22-12
- Creating a level-break business function, 22-15
- Creating a level-break message, 22-9
- Creating a level-break message business function structure, 22-13
- Creating a media object data structure, 7-7
- Creating a processing options data structure, 7-9
- Creating a simple error message, 21-16
- Creating a table join, 6-17
- Creating a table union, 6-19
- Creating a text substitution error message, 21-18
- Creating a web view subheading on a menu selection, 19-22
- Creating and saving, event rules, 12-29
- Creating data structures, 7-7
- Creating fast path selections, 19-23
- Creating OMW objects, 3-24

- Creating or editing, event rules for a business function event rules, 13–8
 - Cross reference facility, 8–1, 8–3
 - rebuilding cross reference information, 8–15
 - searching for batch applications, 8–6
 - searching for business functions, 8–7
 - searching for business views, 8–8
 - searching for data items, 8–4
 - searching for data structures, 8–9
 - searching for event rules, 8–12
 - searching for forms, 8–11
 - searching for interactive applications, 8–5
 - searching for objects, 8–3
 - searching for tables, 8–10
 - viewing field relationships, 8–14
 - Cross reference information, rebuilding cross reference information, 8–15
 - Currency
 - advantages, 18–1
 - application, 18–4
 - build trigger options, 18–3
 - creating a currency conversion trigger, 18–8
 - currency implementation, 18–1
 - setting up currency conversion, 18–6
 - showing currency-sensitive controls, 18–7
 - Currency conversion, setting up currency conversion, 18–6
 - Currency conversion trigger, 18–8
 - Currency process, table event rules, 18–4
 - Cursor-advancing APIs, 14–18
 - Cursors
 - JDECACHE cursors, 14–15
 - opening JDECACHE cursors, 14–15
 - Cut, copy, and paste, controls, 10–29
 - Cutting, event rules, 12–34
- D**
- Data dictionary, 1–8, 4–1
 - alternate language terms, 4–44
 - data structure, 22–12
 - default triggers, 4–5
 - error messages, 4–5
 - glossary items, 4–5
 - jargon, 4–44
 - level-break message, 22–9
 - naming data items, 4–4
 - runtime, 4–3
 - storing data dictionary and data dictionary items, 4–4
 - using the data dictionary, 4–7
 - Data dictionary and data dictionary items, storing Data Dictionary and Data Dictionary items, 4–4
 - Data dictionary item naming conventions, 4–12
 - data item alias, 4–13
 - data item name for an external data dictionary item, 4–13
 - Data dictionary items, naming conventions, 4–13
 - Data Dictionary menu - GH951, 4–7
 - Data dictionary performance, 25–4
 - overrides, 25–4
 - triggers, 25–4
 - validation, 25–5
 - Data Dictionary triggers, override, design time, 10–99
 - Data interdependence, 16–3
 - Data Item, alias, 4–13
 - Data item, 15–11
 - adding an interactive message data item, 21–19
 - attaching and interactive message data item, 21–23
 - attaching default triggers, 4–21, 4–22
 - change row and column text for all applications, 4–46
 - choosing a data item for a business view, 6–12
 - data dictionary item naming conventions, 4–12
 - defining a data item, 4–11
 - general information, 4–16
 - naming a data item, 4–12
 - naming data items, 4–4
 - searching for data items, 8–4
 - selecting a data item for a processing option, 15–11
 - updating the glossary, 4–40
 - Data item alias, 4–13
 - alias for an external data dictionary item, 4–14
 - Data item name, 4–13
 - Data item name for an external data dictionary item, 4–13

- Data item specifications, defining general information, 4-16
- Data items, choosing data items for the table, 5-10
- Data Items form, 4-46
- Data Reference Scan, C-1
- Data reference scan
 - business function data reference scan, C-3
 - event rule data reference scan, C-4
- Data Retrieval, Fix/Inspect form, B-16
- Data retrieval
 - Find/Browse form, B-4
 - first level node in the tree, Parent/Child form, B-10
 - Header Detail form, B-21
 - Headerless Detail form, B-28
 - Parent/Child forms, B-10
 - Search/Select form, B-36
 - tree node expand, Parent/Child form, B-11
- Data selection and sequencing, 25-10
- Data structure
 - creating, documentation, 13-79
 - documentation, template, 13-80
- Data structure objects, data structure performance, performance tips, 25-10
- Data structure performance, 25-10
 - performance tips
 - data structure objects, 25-10
 - restrict the number of fields, 25-10
- Data structure standard data items, 14-37
- Data structure template, attaching to a message, 21-22
- Data structures, 7-1
 - business function data structures, 7-2
 - create a media object data structure, 7-7
 - creating a business function data structure, 7-10
 - creating a media object data structure, 7-7
 - creating a processing options data structure, 7-9
 - creating data structures, 7-7
 - displaying type definitions, 7-5
 - interconnection data structures, 7-3
 - modifying form data structures, 7-3
 - available objects tab, 7-4
 - modifying report data structures, 7-4
 - searching for data structures, 8-9
 - system generated, 7-1
 - system generated data structures
 - form data structures, 7-1
 - report data structures, 7-1
 - type definition, 7-15
 - user generated data structures, 7-1
 - media object data structures, 7-1
 - processing options data structures, 7-1
- Data structures modification rules, A-16
- Data type
 - JDEDATE, 13-22
 - MATH_NUMERIC, 13-22
 - OneWorld, specific, 13-21
- Database
 - connecting, 13-26
 - index limitations, 25-6
 - Access 32, 25-6
 - DB2 for OS/400, 25-7
 - key column violation, 25-7
 - Oracle, 25-7
 - specification file corruption, 25-8
 - SQLSERVER, 25-6
- Database APIs, using, 13-23
- Database communications steps, understanding, 13-26
- Database items, 10-49
 - associating, 10-90
 - filtering, 10-57
 - inserting, form, 10-50
- Database locking, 17-3
- Date functions, advanced functions, expression manager, 12-89
- DB2 for OS/400, 25-7
- Debug logs, 23-25
- Debug tracing
 - system function tracing, 23-30
 - turning on debug tracing, 23-29
- Debugger tools, Microsoft Visual C ++, 23-1
- Debugging, 23-1
 - business functions, 13-33
 - batch applications, 23-16
 - interactive applications, 23-13
 - debug tracing
 - system function tracing, 23-30
 - turning on debug tracing, 23-29
 - debugging an application, 23-9
 - inspecting or modifying a variable, 23-10
 - just in time debugging, 23-11
 - debugging features, 23-2

- debugging process, 23-1
- debugging strategies
 - application encountering errors, 23-24
 - debug logs, 23-25
 - function being called as expected, 23-24
 - output of the program incorrect, 23-24
 - program ending unexpectedly, 23-23
 - source of the problem, 23-24
 - SQL log tracing, 23-26
- debugging tools, 23-1
- event rules debugger, 23-5
 - breakpoint manager, 23-7
 - event rules window, 23-6
 - object browse window, 23-6
 - search combo box, 23-8
 - variable selection and display window, 23-7
- features of Visual C++ debugger, 23-19
 - go command, 23-19
 - local window, 23-20
 - setting breakpoints, 23-20
 - step command, 23-19
 - step into command, 23-19
 - using watch, 23-20
- interpretive and compiled code, 23-3
- modify the Visual C++ debugging environment, 23-21
- setting breakpoints, 23-11
- Debugging an application, 23-9
 - inspecting or modifying a variable, 23-10
 - just in time debugging, 23-11
- Debugging business functions attached to batch applications, 23-16
- Debugging business functions attached to interactive applications, 23-13
- Debugging features, 23-2
- Debugging strategies
 - application encountering errors, 23-24
 - debug logs, 23-25
 - function being called as expected, 23-24
 - output of the program incorrect, 23-24
 - program ending unexpectedly, 23-23
 - source of the problem, 23-24
 - SQL log tracing, 23-26
- Debugging tools, OneWorld Event Rules, 23-1
- Decimal trigger, overriding, 10-108
- Default flags
 - Find/Browse form, B-3
 - Fix/Inspect form, B-15
 - Header Detail form, B-19
 - Headerless Detail form, B-27
 - Parent/Child form, B-9
 - Search/Select form, B-35
- Default Project, managing non-object librarian objects with, 3-4
- Default project, 3-9
- Default triggers, 4-5
 - attaching default triggers, 4-21, 4-22
 - default value trigger, 4-23
 - display rule trigger, 4-29
 - edit rule trigger, 4-26
 - next number trigger, 4-35
 - smart field trigger, 4-36
 - user defined codes trigger, 4-31
 - visual assist trigger, 4-24
- Default value trigger, 4-23
 - overriding, 10-101
- Defining
 - form properties, 10-19
 - grid column properties, 10-74
 - grid properties, 10-71
 - literal values in event rule logic, 12-47
 - options
 - edit or UDC edit control, 10-88
 - form, 10-86
 - forms, grids, edit controls, 10-86
 - grid, 10-87
 - properties, parent/child control, 10-76
- Defining a data item, 4-11
- Defining a new menu, 19-5
- Defining an active message, 21-31
- Defining general information, 4-16
 - updating data items for languages, 4-46
- Defining indices, 5-11
 - J.D. Edwards design standards, 5-12
- Defining jargon, 4-45
- Defining the cache data structure, 14-37
- Defining the glossary text for a runtime message, 21-21
- Defining transaction processing for a form, 16-8
- Defining transaction processing for a report, 16-18
- Delete
 - menu/toolbar items
 - Find/Browse form, B-6
 - Header Detail form, B-25
 - Headerless Detail form, B-32

- Parent/Child form, B-12
 - project, 3-22
 - user from project, 3-44
 - Delete button, processing, 12-26
 - Delete grid rec from DB after, runtime processing, 12-28
 - Delete grid rec from DB before, runtime processing, 12-27
 - Delete grid rec verify after, runtime processing, 12-27
 - Delete grid rec verify before, runtime processing, 12-26
 - Deleting objects from projects, 3-31
 - Deleting records, 14-19
 - Description
 - process flow for edit control, B-41
 - process flow for Find/Browse form, B-3
 - process flow for Fix/Inspect form, B-15
 - process flow for grid control, B-45
 - process flow for Header Detail form, B-19
 - process flow for Headerless Detail form, B-27
 - process flow for Message form, B-39
 - process flow for Parent/Child form, B-9
 - process flow for Search/Select form, B-35
 - Descriptions, associating, 10-90
 - Design standards, J.D. Edwards, 10-5, 10-26, 12-68, 12-75
 - Design time, override, data dictionary triggers, 10-99
 - Designing, form layout, 10-25
 - Detail, cache structure, 13-67
 - Detail data selection and sequencing
 - Find/Browse form, B-4
 - Header Detail form, B-20
 - Headerless Detail form, B-28
 - Parent/Child form, B-10
 - Search/Select form, B-36
 - Development cycle, 1-6
 - Dialog, initialized, 12-14
 - Dialog initialization
 - Find/Browse form, B-3
 - Fix/Inspect form, B-15
 - Header Detail form, B-19
 - Headerless Detail form, B-27
 - Message form, B-39
 - Parent/Child form, B-9
 - Search/Select form, B-35
 - Dictionary item naming conventions, data item name, 4-13
 - Dictionary items, 10-49
 - associating, 10-90
 - inserting, form, 10-51
 - Disconnecting, static text from controls, 10-59
 - Display rule trigger, 4-29
 - Displaying type definitions, 7-5
 - Distributed business functions, 13-5
 - DLLs, 13-4
 - Do in while loop, runtime processing, 12-16
 - Document processing, transaction master business function, implementing, 13-70
 - Documentation
 - business function, 13-77
 - creating, 13-77, 13-78
 - template, 13-79
 - business function - values to pass, viewing, 13-86
 - business function document viewer, viewing, 13-87
 - business function search, viewing, 13-86
 - data structure
 - creating, 13-79
 - template, 13-80
 - generating, business function, 13-81
 - parameter, creating, 13-81
 - viewing, business function, 13-85
 - Double-click on grid row, process flow for grid control, B-49
 - Dragging and dropping, 10-85
 - Dynamic Overrides, 12-109
- ## E
- Edit control, control exited and change, asynchronous, B-43
 - Edit controls
 - creating, 10-53
 - defining options, 10-88
 - Edit document
 - application specific parameters, 13-63
 - common parameters, 13-63
 - function name, 13-62

- hook up tips, 13–63
- naming, transaction master business function modules, 13–62
- special logic/processing required, 13–62
- what does it do, 13–62
- Edit line
 - naming, transaction master business function modules, 13–60
 - what does it do, 13–60
- Edit rule trigger, 4–26
 - overriding, 10–104
- Editing keypad, expression manager, 12–89
- Editing or creating
 - event rules, business function event rules, 13–8
 - event rules for a business function event rules, 13–8
- Elements, forms, 10–3
- Embedded, event rules, 12–3
- Embedded event rules, 1–11
 - application event rules, 1–11
 - table event rules, 1–11
- End document
 - application specific parameters, 13–64
 - common parameters, 13–64
 - function name, 13–63
 - hook up tips, 13–64
 - naming, transaction master business function modules, 13–63
 - special logic/processing required, 13–64
 - what does it do, 13–63
- Enterprise server, jde.ini, setting, 16–24, 16–28
- Environment, customizing the Visual C++ debugging environment, 23–21
- Error handling, 21–5
 - error setting, 21–7
 - error settings
 - automatic error settings, 21–7
 - manual error settings, 21–7
 - event-driven model, 21–5
 - multilevel error messaging for C business functions, 21–11
 - resetting errors, 21–9
- Error messages, 4–5, 21–2, 21–15
 - adding an interactive message data item, 21–19
 - attaching an interactive message data item, 21–23
 - batch error messages, 22–1
 - creating a level-break message, 22–9
 - creating a simple error message, 21–16
 - creating a text substitution error message, 21–18
 - locating an existing error message, 21–15
- Error messaging performance, 25–15
- Error output, reviewing, 13–52
- Error Setting, 21–7
- Error settings
 - automatic error settings, 21–7
 - manual error settings, 21–7
- Errors, resolving, 13–52
 - business function builder tools, 13–52
- Event
 - attaching a business function, 12–65
 - attaching a system function, 12–57
- Event flow, typical, Find/Browse form, 12–12
- Event rule
 - form flow, 12–12
 - manipulation, 12–10
 - modification rules, A–14
 - Table I/O, creating, 12–92
- Event rule data reference scan, C–4
- Event Rule Design, working with, 12–29
- Event rule logic
 - if statements, 12–43
 - literal values, defining, 12–47
 - printing, 12–37
 - while statements, 12–44
- Event rule variables
 - batch application, 12–50
 - creating, 12–50
 - interactive, 12–49
 - working with, 12–49
- Event rules, 1–10
 - adding, comment lines, 12–36
 - application, 12–4
 - attaching, Menu/Toolbar Exit, 10–39
 - business function, 12–3
 - business function event rules, 1–11
 - creating or editing, 13–8
 - copying, 12–34
 - creating and saving, 12–29
 - currency, build triggers options, 18–3
 - currency processing, 18–4
 - cutting, 12–34
 - embedded, 12–3
 - embedded event rules, 1–11
 - paste, 12–34

- searching for event rules, 8–12
 - table, 12–4
 - understanding, 12–3
 - validating, 12–39
 - Event rules buttons, understanding, 12–4
 - Event rules debugger
 - debugging an application, 23–9
 - inspecting or modifying a variable, 23–10
 - just in time debugging, 23–11
 - main controls, 23–5
 - search combo box, 23–8
 - Event Rules Design
 - OneWorld tools, 12–1
 - Table I/O, 12–91
 - Event rules performance, 25–12
 - performance tips
 - I/O commands in event rule code, 25–14
 - JDB API calls in a business function, 25–14
 - JDBFetch statements, 25–14
 - JDBFetchNext, 25–14
 - jdeCache instead of work tables, 25–13
 - Event rules window, 23–6
 - Event-driven model, 21–5
 - Event-level, interactive event rule variables, 12–50
 - Events, understanding, 12–2
 - Exited and changed - asynch
 - Column, 12–113
 - control, 12–113
 - row, 12–113
 - Exited processing, control, 12–10
 - Expanded node, BrowsER, 12–121
 - Explorer, adding or changing web addresses, 19–21
 - Expression, creating for an assignment, 12–87
 - Expression editing field, expression manager, 12–88
 - Expression manager, 12–87
 - advanced functions, 12–89
 - date functions, 12–89
 - general functions, 12–89
 - text functions, 12–90
 - trig functions, 12–90
 - assignment display, 12–88
 - available information, 12–88
 - Editing keypad, 12–89
 - expression editing field, 12–88
 - Extending a transaction boundary, 16–9
 - Extending a transaction boundary between forms, 16–9
 - Extending a transaction boundary using business functions, 16–13
 - Extending a transaction boundary using table I/O, 16–16
 - External Business Function, calling an API from, 13–27
 - External data dictionary Item, alias for an external data dictionary item, 4–14
 - External data dictionary item, 4–13
 - External developer considerations, J.D.
 - Edwards naming standards, guidelines when adding a business view, 6–7
- ## F
- F0002 - Next Number table, 4–35
 - F00165 - Media object storage, 7–7
 - Fast path, creating fast path selections, 19–23
 - Features, business functions, 13–2
 - Field relationships, viewing field relationships, 8–14
 - Filter ER Records, BrowsER, 12–122
 - Filtering
 - database items, 10–57
 - menu filtering, 19–3
 - Filtering projects, 3–13
 - Find, menu/toolbar items
 - Find/Browse form, B–5
 - Header Detail form, B–26
 - Headerless Detail form, B–32
 - Parent/Child form, B–13
 - Search/Select form, B–37
 - Find/Browse
 - form design performance tips, 25–11
 - forms, 10–6
 - Find/Browse form
 - closing form, B–5
 - data retrieval, B–4
 - default flags, B–3
 - dialog initialization, B–3
 - header data retrieval form, B–4
 - menu/toolbar items, B–5
 - add, B–6

- close, B-5
- copy, B-6
- delete, B-6
- find, B-5
- select, B-5
- user defined items, B-7
- process flow, description, B-3
- typical event flow, 12-12
- Find/browse form, detail data selection, B-4
- First level node in the tree, data retrieval, Parent/Child form, B-10
- First-level messages, 22-3
- Fix/Inspect, forms, 10-7
- Fix/Inspect form
 - clearing dialog, B-16
 - closing form, B-17
 - data retrieval, B-16
 - default flags, B-15
 - dialog initialization, B-15
 - Find/Browse form, OK, B-17
 - menu/toolbar items, B-17
 - cancel, B-18
 - OK, B-17
 - user defined items, B-18
 - process flow, B-15
 - description, B-15
- Form
 - defining transaction processing for a form, 16-8
 - testing, 10-125
- Form controls, understanding, 10-44
- Form data structures, 7-1
- Form design, 1-9, 25-11
 - find/browse, 25-11
 - header detail and headerless detail, 25-12
- Form flow, event rule, 12-12
- Form Interconnect calls, J.D. Edwards, standards, B-39
- Form interconnections, creating, 12-71
- Form interconnects
 - modal processing, 12-71
 - modeless processing, 12-71
- Form processing, B-1
 - process flow for Find/Browse form, B-3
 - understanding, 12-2
- Form types, understanding, 10-5
- Format tables, performance impact, 13-75
- Formatted data, 5-23
- Formatting trigger, overriding, 10-103
- Form-level, interactive event rule variables, 12-49
- Forms
 - business function builder, understanding, 13-37
 - Business Function Design, 13-36
 - changing size, 10-26
 - controls, understanding, 12-2
 - creating, 10-17
 - Data Items, 4-46
 - defined, 10-2
 - defining options, 10-86
 - defining properties, 10-19
 - designing layout, 10-25
 - elements, 10-3
 - events, understanding, 12-2
 - extending a transaction boundary, 16-9
 - extending a transaction boundary between forms, 16-9
 - Find/Browse, 10-6
 - Fix/Inspect, 10-7
 - form processing, understanding, 12-2
 - Header Detail, 10-9
 - Headerless Detail, 10-10
 - inserting
 - database items, 10-50
 - dictionary items, 10-51
 - Menu Execution, 19-22
 - Menu Header Revisions, 19-6, 19-17, 19-29
 - Menu Selection Revisions, 19-12, 19-14, 19-15, 19-18, 19-19, 19-21, 19-22, 19-27, 19-30
 - Menu Selection Search, 19-28
 - Menu Text Overrides, 19-28, 19-30
 - Message, 10-13
 - OneWorld Report, 19-17
 - Parent/Child, 10-14
 - parent/child control, 10-76
 - ReNUMBER Selection, 19-30
 - Search and Select, 10-12
 - searching for forms, 8-11
 - select type, 10-18
 - testing, 10-125
 - types, understanding, 10-5
 - User Defined Codes, 19-24
 - using the same cache, 14-14
 - Windows Application, 19-20

- Work With Menu Selections, 19–12, 19–24, 19–27, 19–30
- Work With Menus, 19–28
- Work With User Defined Codes, 19–24
- WorldVision, 19–18
- Forms Design, OneWorld tools, 10–1
- Function name
 - begin document, 13–58, 13–60
 - cancel document (optional), 13–65
 - clear cache, 13–64
 - edit document, 13–62
 - end document, 13–63
- Functions
 - attaching, 12–57
 - distributed business, 13–5
- Functions list, business function builder form, 13–38
- Fundamentals
 - business view design, 6–1
 - cross reference facility, 8–1, 8–3
 - data dictionary, 4–1
 - data structures, 7–1

G

- General Functions, advanced functions, expression manager, 12–89
- General standards, cache APIs, 14–36
- Generating a form, 24–7
- Generating header files, 5–16
- Generating indexes, 5–16
- Generating tables, 5–15
- Getting object specifications, 3–31
- GH951 - Data Dictionary menu, 4–7
- Glossary, updating the glossary, 4–40
- Glossary items, 4–5
- Glossary text, defining for a runtime message, 21–21
- Go command, 23–19
- Grid
 - defining options, 10–87
 - processing, 12–11
 - set lower limits, 12–11
- Grid column properties, defining, 10–74
- Grid properties, defining, 10–71
- Grid rec is fetched, runtime processing, 12–18
- Grids, changing size, 10–26

- Group, controls, aligning, 10–29
- Group boxes, creating, 10–69
- Group cache business function header file, 14–39
- Group cache function, 14–37
- Group programming standards, individual cache business function header file, 14–40

H

- Handle request, event rule variable, 12–51
- Header, cache structure, 13–66
- Header data retrieval
 - Find/Browse forms, B–4
 - Header Detail form, B–20
 - Parent/Child form, B–10
 - Search/Select form, B–36
- Header Detail, forms, 10–9
- Header detail and headerless detail, form design performance tips, 25–12
- Header Detail form
 - Add entry row to grid, B–22
 - clearing dialog, B–22
 - closing form, B–22
 - data retrieval, B–21
 - default flags, B–19
 - detail data selection and sequencing, B–20
 - dialog initialization, B–19
 - header data retrieval form, B–20
 - menu/toolbar items, B–22
 - cancel, B–25
 - delete, B–25
 - find, B–26
 - OK, B–22
 - user defined items, B–26
 - process flow, B–19
 - description, B–19
- Header file sections, 13–14
- Headerless Detail, forms, 10–10
- Headerless Detail form
 - add entry row to grid, B–29
 - clearing dialog, B–29
 - closing form, B–29
 - data retrieval, B–28
 - default flags, B–27
 - detail data selection and sequencing, B–28

- dialog initialization, B-27
 - menu/toolbar items, B-30
 - cancel, B-32
 - delete, B-32
 - find, B-32
 - OK, B-30
 - user defined items, B-33
 - process flow, B-27
 - description, B-27
 - HJDECURSOR, 14-15
 - Hook up tips
 - begin document, 13-58
 - edit document, 13-63
 - end document, 13-64
 - How JDECACHE partial key works, 14-22
 - How OneWorld works at design time, 1-13
 - How table event rules work with currency processing, 18-4
 - How the data dictionary is used at runtime, 4-3
 - Hungarian notation, example of, 12-51
- I**
- I/O commands in event rule code, event rules performance, performance tips, 25-14
 - Icon, object type, definitions, 3-5
 - If and while statements
 - creating, 12-44
 - working with, 12-43
 - If statements, event rule logic, 12-43
 - imaging, third-party software, 10-116
 - Implementing, transaction master business functions, 13-69
 - document processing, 13-70
 - single record processing, 13-69
 - Index
 - generate for tables, 5-16
 - using an index to access the cache, 14-11
 - Index considerations, 25-5
 - Index limitations for various databases, 25-6
 - Access 32, 25-6
 - DB2 for OS/400, 25-7
 - key column violation, 25-7
 - Oracle, 25-7
 - specification file corruption, 25-8
 - SQLSERVER, 25-6
 - Indices, 5-5, 6-3
 - defining indices, 5-11
 - J.D. Edwards design standards, 5-12
 - JDECACHE, 14-8
 - setting up indices, 14-8
 - Indices and logicals, coexistence, 5-5, 5-12
 - Individual cache business function header file, 14-40
 - Individual cache function, 14-37
 - Information Messages, 21-3
 - Information messages, glossary items, 4-5
 - Information structure, standard parameters, 13-74
 - Inherit, token, 3-38
 - Initialized
 - dialog, 12-14
 - post dialog, 12-15
 - pre dialog, 12-13
 - Initializing the cache, 14-10
 - Inner joins. *See* Simple joins
 - Inserting
 - database items, form, 10-50
 - dictionary items, form, 10-51
 - Inspecting or modifying a variable, 23-10
 - Interactive applications, 9-3
 - debugging business functions, 23-13
 - modification rules, A-4
 - searching for interactive applications, 8-5
 - Interactive event rule variables, 12-49
 - event-level, 12-50
 - form-level, 12-49
 - Interconnect data structure, 7-3
 - Interconnection data structures
 - displaying type definitions, 7-5
 - modifying form data structures, 7-3
 - modifying report data structures, 7-4
 - Interpretive and compiled code, 23-3
- J**
- J.D. Edwards
 - design standards, 10-5, 10-26, 12-68, 12-75
 - ODBC - Open database connectivity, 13-24
 - open database connectivity, 13-24
 - standards, from interconnect calls, B-39

- J.D. Edwards design standards, guidelines to name an index, 5–12
- J.D. Edwards naming standards, guidelines when adding a business view, 6–6
 - external developer considerations, 6–7
- J.D. Edwards processing option naming standards, 15–9
- J.D. Edwards standards
 - process flow for edit control, B–43
 - process flow for grid control, B–49
 - processing option naming standards, 15–9
 - comment, 15–10
 - data item, 15–11
 - processing option data structure, 15–9
 - tab title, 15–9
- Jargon, 4–44
 - defining, 4–45
- Java & HTML Generator, options, 24–6
- Java & HTML Generator, logging in, 24–6
- Java application, 24–3
- Java generator, 9–4
- JDB API calls in a business function, event rules performance, performance tips, 25–14
- JDBFetch statements, event rules performance, performance tips, 25–14
- JDBFetchNext, event rules performance, performance tips, 25–14
- jde.ini
 - enterprise server, setting, 16–24, 16–28
 - record change detection, 17–3
 - workstation, setting, 16–26
- jde.ini file, setting for transaction processing, 16–21
- JDE.LOG, 14–23
- JDEBASE API, standard categories, 13–25
- JDEBase APIs, record locking, 17–4
- JDECACHE, 14–7
 - calling JDECACHE APIs, 14–7
 - initializing the cache, 14–10
 - JDECACHE errors, 14–23
 - JDECACHE standards, 14–35
 - jdeCacheInit/jdeCache terminate rule, 14–14
 - multiple cursor support, 14–20
 - partial keys, 14–21
 - performance considerations, 14–4
 - setting up indices, 14–8
 - using an index to access the cache, 14–11
 - using in multiple business functions or forms, 14–14
 - when to use JDECACHE, 14–3
- JDECACHE API set, 14–4
 - JDECACHE management APIs, 14–4
 - JDECACHE manipulation APIs, 14–5
- JDECACHE cursor, how a JDECACHE partial key works, 14–22
- JDECACHE cursors, 14–15
 - closing a cursor, 14–20
 - deleting records, 14–19
 - HJDECURSOR, 14–15
 - JDECACHE dataset, 14–16
 - JDECACHE multiple cursor support, 14–20
 - JDECACHE partial keys, 14–21
 - jdeCacheFetchPosition, 14–19
 - jdeCacheFetchPositionByRef, 14–20
 - opening a JDECACHE cursor, 14–15
 - resetting the cursor, 14–20
 - types of JDECACHE cursors APIs, 14–17
 - cursor-advancing APIs, 14–18
 - non-cursor-advancing APIs, 14–18
 - updating records, 14–18
- JDECACHE dataset, 14–16
- JDECACHE errors, 14–23
- JDECACHE example program, 14–25
- JDECACHE management APIs, 14–4
- JDECACHE manipulation, APIs, 14–6
- JDECACHE manipulation APIs, 14–5
- JDECACHE multiple cursor support, 14–20
- JDECACHE partial keys, 14–21
- JDECACHE standards, 14–35
 - cache business function description, 14–35
 - cache business function source description, 14–35
 - cache business function source name, 14–35
- jdeCacheDelete, 14–19
- jdeCacheFetchPosition, 14–18
- jdeCacheInit, 14–10
- jdeCacheInit/jdeCacheTerminate rule, 14–14
- JDEDATE, data type, 13–22
- JDEDEBUG.LOG, 14–23
- Joined business view versus table I/O, business view performance, performance tips, 25–10
- Joined views, naming conventions, 6–7
- Just in time debugging, 23–11

K

Key column violation, 25–7
 Key pressed, process flow for grid control, B–49
 Kill focus on grid, process flow for grid control, B–48

L

Language
 alternate language terms for data items, 4–44
 changing menu text for languages, 19–28
 media objects, considerations, 10–120
 output for web generator, 24–6
 Language considerations, media objects, 10–120
 Language considerations for processing options, 15–12
 Language Preview, 10–125
 Language specific media object attachment, adding, 10–120
 Languages, updating data items, 4–46
 Last grid rec has been read, runtime processing, 12–22
 Left outer joins, 6–2
 Level break messages, 22–6
 Level messages, 21–2
 Level-break messages, 22–3
 creating a data dictionary item, 22–9
 data dictionary, creating a data structure, 22–12
 first-level messages, 22–3
 second-level messages, 22–4
 third-level messages, 22–5
 Link section, build output, 13–44
 Linking, business function objects, 13–40
 Linking menus, 19–22
 Literal values, event rule logic, defining, 12–47
 Loading, nodes to the tree, 10–82
 Local window, 23–20
 Locating an existing error message, 21–15
 Logic, event rule, printing, 12–37

M

Main Form
 components, 3–5
 news/status view, 3–5
 search view, 3–5
 Maintain, user status, 3–44
 Maintaining objects in multiple software releases, 3–34
 Makefile section, build output, 13–43
 Manipulation, event rule, 12–10
 Manual error settings, 21–7
 Master business functions
 creating, processing options, 13–56
 information structure, standard
 parameters, 13–74
 master file, 13–71
 naming convention, 13–55
 Master file, master business functions, 13–71
 MATH_Numeric, data type, 13–22
 Math_Numeric, currency, application, 18–4
 Media object controls, creating, 10–60, 10–62
 Media object data structures, 7–1
 creating a media object data structure, 7–7
 media object storage - F00165 table, 7–7
 Media object storage - F00165, 7–7
 Media objects
 media considerations, 10–120
 processing, 10–115
 standard processing, using, 10–116
 Memory allocation, business function performance, 25–15
 Menu, Data Dictionary menu - GH951, 4–7
 Menu design
 menu design tables, 19–4
 menu filtering, 19–3
 menu selection revisions
 changing menu selection text, 19–30
 changing menu text for languages, 19–28
 copying a menu selection, 19–27
 renumbering a menu selection, 19–30
 Menu design tables, 19–4
 F0082 (Menu Master), 19–4
 F00821 (Menu Selection), 19–4
 F0083 (Menu Text Override), 19–4

- F0084 (Menu Path), 19-4
- Menu filtering, 19-3
- Menu Master - F0082, 19-4
- Menu Path - F0084, 19-4
- Menu report, printing a menu report, 19-8
- Menu selection
 - adding an application to a menu, 19-14
 - adding or changing a menu selection, 19-11
 - define a menu selection, 19-14
 - name a menu selection, 19-12
- Menu Selection - F00821, 19-4
- Menu selection revisions
 - changing menu selection text, 19-30
 - changing menu text for languages, 19-28
 - copying a menu selection, 19-27
 - renumbering a menu selection, 19-30
- Menu selections
 - adding or changing web address on OneWorld Explorer Help, 19-21
 - creating a web view subheading on a menu, 19-22
 - creating fast path selections, 19-23
 - linking menus, 19-22
- Menu Text Override - F0083, 19-4
- Menu/Toolbar Exits
 - attaching, event rules, 10-39
 - creating, 10-33
- Menu/toolbar items
 - Find/Browse form, B-5
 - add, B-6
 - close, B-5
 - copy, B-6
 - delete, B-6
 - find, B-5
 - select, B-5
 - user defined items, B-7
 - Fix/Inspect form, B-17
 - cancel, B-18
 - user defined items, B-18
 - Header Detail form, B-22
 - cancel, B-25
 - delete, B-25
 - find, B-26
 - OK, B-22
 - user defined items, B-26
 - Headerless Detail form, B-30
 - cancel, B-32
 - delete, B-32
 - find, B-32
 - OK, B-30
 - user defined items, B-33
- Parent/Child form, B-12
 - add, B-13
 - close, B-12
 - copy, B-13
 - delete, B-12
 - find, B-13
 - select, B-12
 - user defined items, B-14
- Search/Select form, B-37
 - find, B-37
 - select, B-37
 - user defined items, B-38
- Menus, 1-12
 - add a OneWorld application to a menu, 19-15
 - add a OneWorld report section, 19-17
 - add a Windows application to a menu, 19-19
 - add a WorldVision application to a menu, 19-18
 - adding an application to a menu, 19-14
 - adding or changing a menu selection, 19-11
 - creating a web view subheading on a menu, 19-22
 - creating subcategories, 10-40
 - define a menu selection, 19-14
 - defining a new menu, 19-5
 - linking menus, 19-22
 - menu selection revisions
 - changing menu selection text, 19-30
 - changing menu text for languages, 19-28
 - copying a menu selection, 19-27
 - renumbering a menu selection, 19-30
 - name a menu selection, 19-12
 - printing a menu report, 19-8
 - reviewing selections for a menu, 19-8
- Message, forms, 10-13
- Message form
 - Buttons, B-39
 - closing form, B-39
 - dialog initialization, B-39
 - process flow, B-39
 - description, B-39
- Message types, 21-1
 - error messages, 21-2
 - information messages, 21-3

- level messages, 21–2
 - workflow messages, 21–2
 - Messages
 - action messages, 22–6
 - attaching a data structure template, 21–22
 - batch error messages, 22–1
 - defining an active message, 21–31
 - first-level messages, 22–3
 - level break messages, 22–6
 - level-break messages, 22–3
 - second-level messages, 22–4
 - text substitution error messages, 22–5
 - third-level messages, 22–5
 - Messaging, 21–1
 - batch error messages, 22–3
 - Microsoft Visual C++
 - customizing the environment, 23–21
 - features, 23–19
 - go command, 23–19
 - local window, 23–20
 - setting breakpoints, 23–20
 - step command, 23–19
 - step into command, 23–19
 - using watch, 23–20
 - Modal form interconnect, creating, 12–71
 - Modal processing, form interconnects, 12–71
 - Modeless form interconnect, creating, 12–75
 - Modeless processing, form interconnects, 12–71
 - Modification rules
 - application text, A–10
 - business functions, A–18
 - business views, A–13
 - control tables, A–12
 - data structures, A–16
 - event rules, A–14
 - interactive applications, A–4
 - OneWorld, A–4
 - report specifications, A–7
 - reports, A–7
 - table specifications, A–11
 - versions, A–19
 - Modify
 - project, 3–19
 - project description, 3–19
 - Modifying form data structures, 7–3
 - available objects tab, 7–4
 - Modifying report data structures, 7–4
 - Mouse, size controls, 10–27
 - Moving
 - control and static text, 10–26
 - controls, 10–25
 - Moving multiple objects in an OMW project, 3–29
 - Moving objects, 3–29
 - Moving objects in an OMW project, 3–29
 - Multilevel error messaging for C business functions, 21–11
 - create multi-level error messaging, 21–11
 - Multiple-table or single-table, business view performance, performance tips, 25–9
- ## N
- Naming, transaction master business function modules, 13–57
 - begin document, 13–58
 - Cancel document (optional), 13–65
 - clear cache, 13–64
 - Edit document, 13–62
 - edit line, 13–60
 - end document, 13–63
 - Naming a data item, 4–12
 - Naming convention, 13–74
 - master business functions, 13–55
 - Naming conventions
 - data dictionary item, 4–12
 - data dictionary items, 4–13
 - data item alias, 4–13
 - external data dictionary item, 4–14
 - data item name, 4–13
 - external data dictionary item, 4–13
 - forms, 10–19
 - joined views, 6–7
 - Naming data items, 4–4
 - Native locking strategy, 17–1
 - Next number trigger, 4–35
 - overriding, 10–106
 - Next Numbers table - F0002, 4–35
 - Node, expanded, BrowsER, 12–121
 - Nodes
 - child, 10–83
 - loading to the tree, 10–82
 - parent, 10–83
 - Non-Object Librarian Objects, examples, 3–6
 - Non-cursor-advancing APIs, 14–18

Non-Object Librarian Object

- managing using default project, 3-4
- object librarian object, distinguished from, 3-6

O

Object

- add object to project, 3-27
 - checking objects in and out, 3-32
 - create, 3-23
 - delete objects from project, 3-30
 - getting object specifications, 3-31
 - icon definitions, 3-5
 - maintain objects in multiple software releases, 3-34
 - move an object, 3-29
 - properties, 3-34
 - remove objects from project, 3-30
 - searching for objects with the OMW, 3-25
 - view status, 3-5
- Object browse window, 23-6
- Object ID, show, BrowsER, 12-121
- Object librarian, 1-8
- Object Librarian Object, non-object librarian object, distinguished from, 3-6
- Object Librarian Objects, examples, 3-6
- Object Management Workbench
- allowed (user) action, 3-4
 - concepts, allowed (user) actions, 3-4
 - creating objects, 3-23
 - non-object librarian objects, managing using default project, 3-4
- object
- add object to project, 3-27
 - checking objects in and out, 3-32
 - creating a new object, 3-23
 - delete objects from project, 3-30
 - getting object specifications, 3-31
 - icon definitions, 3-5
 - maintain objects in multiple software releases, 3-34
 - move an object, 3-29
 - properties, 3-34
 - remove objects from project, 3-30
 - searching for objects, 3-25

- view status, 3-5
- object librarian and non-object librarian objects, 3-6
- overview, 3-1
- project, 3-3
- add projects to, 3-21
 - advance project status of, 3-20
 - advanced search, 3-15
 - create, 3-17
 - default project, 3-4
 - delete, 3-22
 - modify, 3-19
 - modify description, 3-19
 - search for by associated objects, 3-16
 - view status, 3-5
- token, 3-4
- inherit, 3-38
 - release, 3-40
 - switch, 3-39
- token queue, 3-38
- user
- add to project, 3-43
 - delete from project, 3-44
 - search, 3-41
 - view and maintain status, 3-44
- user role, 3-4
- Objects, 1-3
- searching for objects, 8-3
 - storing objects, 1-7
- Objects and applications
- application, 1-4
 - object, 1-3
- ODBC - Open database connectivity, J.D. Edwards, 13-24
- OK, menu/toolbar items
- Header Detail form, B-22
 - Headerless Detail form, B-30
- Ok, menu/toolbar items, Fix/Inspect form, B-17
- OneWorld
- debugging tools, interpretive and compiled code, 23-3
 - modification rules, A-4
 - specific data type, 13-21
- OneWorld applications, add a OneWorld application to a menu, 19-15
- OneWorld currency implementation, 18-1
- OneWorld report section, add to a menu, 19-17

- OneWorld report selection, add a
 - OneWorld report selection to a menu, 19–17
 - OneWorld storing objects, 1–7
 - OneWorld Tool Set, overview, 1–8
 - OneWorld Tools, 1–1
 - business view design, 6–1
 - OneWorld tools
 - build an application, 1–5
 - business function design, 1–10
 - Business functions, 13–1
 - business view design, 1–9
 - data dictionary, 1–8, 4–1
 - development cycle, 1–6
 - event rules, 1–10
 - Event Rules Design, 12–1
 - form design, 1–9
 - Forms Design, 10–1
 - menus, 1–12
 - object librarian, 1–8
 - processing options, 1–12
 - report design, 1–10
 - table design, 1–9, 5–9
 - Open database connectivity, J.D. Edwards, 13–24
 - Opening a JDECACHE cursor, 14–15
 - Optimistic locking, 17–3
 - database locking, 17–3
 - Options
 - BrowsER, working with, 12–120
 - defining, 10–86
 - edit or UDC edit control, 10–88
 - form, 10–86
 - grid, 10–87
 - Oracle, 25–7
 - Override
 - data dictionary triggers, design time, 10–99
 - decimal trigger, 10–108
 - default value trigger, 10–101
 - edit rule trigger, 10–104
 - formatting trigger, 10–103
 - next number trigger, 10–106
 - previously defined, 10–99
 - styles trigger, 10–107
 - user defined codes trigger, 10–105
 - visuals assist trigger, 10–102
 - Overrides, 25–4
 - dynamic, 12–109
 - Overview of the debugging process, 23–1
- ## P
- Page-at-a-time processing, runtime processing, 12–16
 - Parameter, creating, documentation, 13–81
 - Parent DLL, business functions, changing, 13–33
 - Parent nodes, 10–83
 - Parent/Child, forms, 10–14
 - Parent/child control
 - defining properties, 10–76
 - form, 10–76
 - Parent/Child form
 - closing form, B–12
 - default flags, B–9
 - detail data selection, B–10
 - dialog initialization, B–9
 - header data retrieval form, B–10
 - menu/toolbar items, B–12
 - add, B–13
 - close, B–12
 - copy, B–13
 - delete, B–12
 - find, B–13
 - select, B–12
 - user defined items, B–14
 - process flow, B–9
 - description, B–9
 - Parent/Child forms, data retrieval, B–10
 - first level node in the tree, B–10
 - tree node expand, B–11
 - Paste, event rules, 12–34
 - Paste options, setting, 12–35
 - Paste, cut, copy, controls, 10–29
 - Performance considerations, JDECACHE guidelines, 14–4
 - Performance impact, format tables, 13–75
 - Performance tips, 25–3
 - batch application performance
 - setting up level breaks, 25–12
 - slow performance, 25–12
 - batch application performance, 25–12
 - business function performance, 25–14
 - balance table opens and closes, 25–15
 - memory allocation, 25–15

- business view performance, 25–9
 - joined business view versus table I/O, 25–10
 - restrict the number of fields, 25–9
 - single-table or multiple-table, 25–9
 - table I/O versus joined business view, 25–10
 - unions, 25–9
- considerations for coexisting with World software, 25–6
- data dictionary performance, 25–4
 - overrides, 25–4
 - triggers, 25–4
 - validation, 25–5
- data selection and sequencing, 25–10
- data structure performance, 25–10
 - data structure objects, 25–10
 - restrict the number of fields, 25–10
- error messaging performance, 25–15
- event rules performance, 25–12
 - I/O commands in event rule code, 25–14
 - JDB API calls in a business function, 25–14
 - JDBFetch statements, 25–14
 - JDBFetchNext, 25–14
 - jdeCache instead of work tables, 25–13
- form design, 25–11
 - find/browse, 25–11
 - header detail and headerless detail, 25–12
- index limitations for various databases, 25–6
 - Access 32, 25–6
 - DB2 for OS/400, 25–7
 - key column violation, 25–7
 - Oracle, 25–7
 - specification file corruption, 25–8
 - SQLSERVER, 25–6
- table design performance, 25–5
 - index considerations, 25–5
- table I/O objects, 25–8
- things to look for, 25–3
- transaction processing performance, 25–15
- Performing, build all, 13–45
- Pessimistic locking, 17–4
 - record locking, 17–4
- Pessimistic record locking
 - business functions, 17–5
 - transaction boundary, 17–5
- Portability and standards, understanding, 13–24
- Post dialog, initialized, 12–15
- Pre dialog, initialized, 12–13
- Previewing tables, 5–13
- Previously defined triggers, overriding, 10–99
- Printing, event rule logic, 12–37
- Process flow for edit control, B–41
 - control is entered, B–42
 - control is exited, B–42
 - asynchronous, B–43
 - inline, B–43
 - description, B–41
 - J.D. Edwards standards, B–43
 - properties and options, B–41
- Process flow for Find/Browse, B–3
- Process flow for Find/Browse form, description, B–3
- Process flow for Fix/Inspect form, B–15
 - description, B–15
- Process flow for grid control, B–45
 - cell is exited after change, B–49
 - description, B–45
 - double-click on grid row, B–49
 - J.D. Edwards standards, B–49
 - key pressed, B–49
 - kill focus on grid, B–48
 - properties and options, B–46
 - row is entered, B–48
 - row is exited, B–48
 - row is exited and changed, asynchronous, B–48
 - row is exited validation, B–49
 - set focus on grid, B–48
- Process flow for Header Detail form, B–19
 - description, B–19
- Process flow for Headerless Detail form, B–27
 - description, B–27
- Process flow for Message form, B–39
 - description, B–39
- Process flow for Parent/Child form, B–9
 - description, B–9
- Process flow for Search/Select form, B–35
 - description, B–35
- Processing
 - add button, 12–25
 - available objects, 12–10

- delete button, 12–26
 - media objects, 10–115
 - runtime, 12–7
 - select button, 12–23
 - Processing option data structure, 15–9
 - Processing option template, change a processing option template for text translation, 15–13
 - Processing options, 1–12
 - add a processing template with translated text, 15–15
 - J.D. Edwards processing option naming standards, 15–9
 - comment, 15–10
 - data item, 15–11
 - processing option data structure, 15–9
 - tab title, 15–9
 - language considerations, 15–12
 - master business functions, creating, 13–56
 - templates, 15–2
 - Processing options data structures, 7–1, 7–9
 - Processing work center APIs, calling the processing work center APIs, 22–21
 - Program
 - application encountering errors, 23–24
 - debug logs, 23–25
 - ending unexpectedly, 23–23
 - function being called as expected, 23–24
 - output of the program incorrect, 23–24
 - source of the problem, 23–24
 - SQL log tracing, 23–26
 - Programming standards, cache
 - programming standards, 14–36
 - cache action code standards, 14–37
 - cache name, 14–36
 - data structure standard data items, 14–37
 - defining the cache data structure, 14–37
 - general standards, 14–36
 - group cache business function header file, 14–39
 - terminating versus clearing cache, 14–36
 - Programs and IDs, P0082 (Menu Design)
 - Menu Execution, 19–22
 - Menu Header Revisions, 19–6, 19–17, 19–29
 - Menu Selection Revisions, 19–12, 19–14, 19–15, 19–18, 19–19, 19–21, 19–22, 19–27, 19–30
 - Menu Selection Search, 19–28
 - Menu Text Overrides, 19–28, 19–30
 - OneWorld Report, 19–17
 - Renumber Selection, 19–30
 - User Defined Codes, 19–24
 - Windows Application, 19–20
 - Work With Menu Selections, 19–12, 19–24, 19–27, 19–30
 - Work With Menus, 19–28
 - Work With User Defined Codes, 19–24
 - WorldVision, 19–18
 - Project
 - add projects to, 3–21
 - add user to, 3–43
 - advance project status of, 3–20
 - advanced search, 3–15
 - create, 3–17
 - delete, 3–22
 - delete user, 3–44
 - modify, 3–19
 - modify description of, 3–19
 - search for by associated objects, 3–16
 - view status, 3–5
 - viewing, 3–13
 - project, 3–3
 - Project Status, advance, 3–20
 - Project window, 3–13
 - Projects
 - default, 3–9
 - filtering, 3–13
 - life cycle, 3–9
 - searching, 3–15
 - Properties
 - defining, parent/child control, 10–76
 - grid, defining, 10–71
 - grid column, defining, 10–74
 - Properties and options
 - process flow for edit control, B–41
 - process flow for grid control, B–46
 - Push buttons, creating, 10–45
- ## Q
- Quick Form, using, 10–113
- ## R
- Radio buttons, creating, 10–48

- Reading, build output, 13–42
- Rebase section, build output, 13–44
- Rebuilding cross reference information, 8–15
- Record locking, 17–1
 - JDEBase APIs, 17–4
 - native locking strategy, 17–1
 - optimistic locking, 17–3
 - pessimistic locking, 17–4
- Records
 - deleting records, 14–19
 - jdeCacheDelete, 14–19
 - jdeCacheFetchPosition, 14–18
 - updating records, 14–18
- Related functions, adding, 13–33
- Related tables and functions, adding, 13–32
- Release, token, 3–40
- Removing objects from projects, 3–30
- Removing tables from the database, 5–17
- Renumbering a menu selection, 19–30
- Replicated data dictionary, storing data dictionary and data dictionary items, 4–4
- Report
 - add a OneWorld report selection to a menu, 19–17
 - defining transaction processing, 16–18
- Report data structures, 7–1
 - modifying report data structures, 7–4
- Report design, 1–10
- Report interconnections, creating, 12–81
- Report specification modification rules, A–7
- Reports, printing a menu report, 19–8
- Resetting errors, 21–9
- Resetting the cursor, 14–20
- Restrict the number of fields
 - business view performance, performance tips, 25–9
 - data structure performance, performance tips, 25–10
- Reviewing, error output, 13–52
- Reviewing selections for a menu, 19–8
- Right outer joins, 6–2
- Rollback, 16–2
- Row, exited and changed - asynch, 12–113
- Row and column text, change for all applications, 4–46
- Row is entered, process flow for grid control, B–48
- Row is exited, process flow for grid control, B–48

- Row is exited and changed - asynchronous, process flow for grid control, B–48
- Row is exited validation, process flow for grid control, B–49
- Runtime
 - data structures, 12–7
 - processing, 12–7
- Runtime data structures and available objects, 12–7
- Runtime message, defining the glossary text, 21–21
- Runtime processing
 - all grid recs deleted from DB, 12–28
 - BC assigned database values, 12–17
 - button clicked, 12–24
 - delete grid rec from DB after, 12–28
 - delete grid rec from DB before, 12–27
 - delete grid rec verity after, 12–27
 - delete grid rec verity before, 12–26
 - do in while loop, 12–16
 - grid rec is fetched, 12–18
 - last grid rec has been read, 12–22
 - page-at-a-time processing, 12–16
 - SQL select, 12–16
 - write grid line after, 12–21
 - write grid line before, 12–19

S

- Saving and creating, event rules, 12–29
- Search
 - advanced, projects, 3–15
 - BrowsER, 12–122
 - project, by associated objects, 3–16
 - user, 3–41
- Search and Select, forms, 10–12
- Search combo box, 23–8
- Search/Select form
 - clearing dialog, B–37
 - closing form, B–37
 - data retrieval, B–36
 - default flags, B–35
 - detail data selection and sequencing, B–36
 - dialog initialization, B–35
 - header data retrieval form, B–36
 - menu/toolbar items, B–37
 - close, B–37

- find, B-37
- select, B-37
- user defined items, B-38
- process flow, B-35
 - description, B-35
- Searching, for projects, 3-15
- Searching for batch applications, 8-6
- Searching for business functions, 8-7
- Searching for business views, 8-8
- Searching for data items, 8-4
- Searching for data structures, 8-9
- Searching for event rules, 8-12
- Searching for forms, 8-11
- Searching for interactive applications, 8-5
- Searching for objects, 3-25, 8-3
- Searching for tables, 8-10
- Second-level messages, 22-4
- Select, menu/toolbar items
 - Find/Browse form, B-5
 - Parent/Child form, B-12
 - Search/Select form, B-37
- Select button, processing, 12-23
- Select distinct, 6-2, 6-14
- Select/Search form, menu/toolbar items, close, B-37
- Selecting
 - business views, 10-21
 - controls, 10-25
 - form type, 10-18
- Send Message system function, 21-27
- Send message system functions
 - defining an active message, 21-31
 - use the send message system function, 21-28
- Set focus on grid, process flow for grid control, B-48
- Setting
 - build options, 13-40
 - enterprise server, jde.ini, 16-24, 16-28
 - paste options, 12-35
 - workstation, jde.ini, 16-26
- Setting breakpoints, 23-11, 23-20
- Setting up currency conversion, 18-6
- Setting up level breaks, 25-12
- Show, object ID, BrowsER, 12-121
- Showing currency-sensitive controls, 18-7
- Simple error messages, 21-16
 - adding an interactive message data item, 21-17
- Simple joins, 6-2
- Single record processing, transaction master business function, implementing, 13-69
- Single-table or multiple-table, business view performance, performance tips, 25-9
- Size, controls using the mouse, 10-27
- Size and place, control using the size command, 10-27
- Size command, size and place a control, 10-27
- Size of alias, for use in an RPG program, 4-13
- Slow performance, 25-12
- Smart field trigger, 4-36
- Special logic/processing required
 - begin document, 13-58, 13-61
 - cancel document (optional), 13-65
 - clear cache, 13-65
 - edit document, 13-62
 - end document, 13-64
- Specification file corruption, 25-8
- SQL log tracing, 23-26
- SQL select, Runtime processing, 12-16
- SQLSERVER, 25-6
- Standard, JDEBASE API categories, 13-25
- Standard parameters, 13-74
- Standard processing, media objects, using, 10-116
- Standards
 - cache programming standards, 14-36
 - cache action code standards, 14-37
 - cache name, 14-36
 - data structure standard data items, 14-37
 - defining the cache data structure, 14-37
 - general standards, 14-36
 - group cache business function header file, 14-39
 - terminating versus clearing cache, 14-36
 - JDECACHE standards, 14-35
 - cache business function description, 14-35
 - cache business function source description, 14-35
 - cache business function source name, 14-35
- Standards and portability, understanding, 13-24

- Static text, controls
 - creating, 10–52
 - disconnecting, 10–59
- Status, user, view and maintain, 3–44
- Step command, 23–19
- Step into command, 23–19
- Steps, database communication, understanding, 13–26
- Storing objects
 - central-storage server, 1–7
 - workstation or server, 1–7
- Structure
 - business function source files, 13–18
 - cache, 13–66
 - detail, 13–67
 - header, 13–66
- Structures, runtime, 12–7
- Styles trigger, overriding, 10–107
- Subcategories, creating in menus, 10–40
- Summary section, build output, 13–45
- Switch, token, 3–39
- System function, event, attaching, 12–57
- System function tracing, 23–30
- System Functions, send message system functions, 21–27
- System functions
 - system function tracing, 23–30
 - transaction processing, 16–6
- System generated data structures, 7–1
 - form data structures, 7–1
 - report data structures, 7–1

T

- Tab sequence, changing, 10–91
- Tab title, 15–9
 - defining a tab title for a processing option, 15–9
- Table
 - adding a table, 5–3
 - indices, 5–5
 - choosing a table for a business view, 6–11
 - choosing data items, 5–10
 - event rules, 12–4
 - Next Numbers - F0002, 4–35
- Table design, 1–9, 5–9
 - adding a table, 5–3

- choosing data items for the table, 5–10
- column properties, 5–24
- defining indices, 5–11
 - J.D. Edwards design standards, 5–12
- previewing tables, 5–13
- universal table browser
 - formatted data, 5–23
 - unformatted data, 5–22
- Table design performance, 25–5
 - index considerations, 25–5
- Table event rules, 1–11, 12–103
 - creating, 12–104
- Table I/O
 - available operations, 12–91
 - event rule, creating, 12–92
 - event rule variable, 12–51
 - Event Rules Design, 12–91
 - extending a transaction boundary, 16–16
 - valid mapping operators, 12–92
- Table I/O objects, 25–8
- Table I/O versus joined business view, business view performance, performance tips, 25–10
- Table join, 6–1
 - creating a table join, 6–17
 - left outer joins, 6–2
 - right outer joins, 6–2
 - simple joins, 6–2
- Table specifications modification rules, A–11
- Table union, 6–2
 - creating a table union, 6–19
- Tables
 - copying tables, 5–16
 - deleting, 5–17
 - generating header files, 5–16
 - generating indexes, 5–16
 - generating tables, 5–15
 - menu design tables, 19–4
 - previewing tables, 5–13
 - removing from database, 5–17
 - searching for tables, 8–10
- Template
 - business function documentation, 13–79
 - data structure documentation, 13–80
- Templates, processing options, 15–2
- Terminating the work center process, 22–23
- Terminating versus clearing cache, 14–36
- Testing, forms, 10–125
- Text Boxes, creating, 10–68

- Text functions, advanced functions, expression manager, 12–90
- Text substitution error messages, 22–5
- Text translation, change a processing option template, 15–13
- Text Variables, Creating, 10–109
- The JDECACHE API set, 14–4
- The jdeCacheFetchPosition API, 14–19
- The jdeCacheFetchPositionByRef API, 14–20
- Things to look for, 25–3
- Third-level messages, 22–5
- Third-party software, imaging, 10–116
- Threaded events, asynchronous events, 12–112
- To add a OneWorld application to a menu, 19–15
- To add a OneWorld report selection to a menu, 19–17
- To add a Windows application to a menu, 19–19
- To add a WorldVision application to a menu, 19–18
- To define a menu selection, 19–14
- To name a menu selection, 19–12
- Token
 - defined, 3–4
 - inherit, 3–38
 - release, 3–40
 - switch, 3–39
- Token queue, 3–38
 - viewing, 3–38
- Transaction boundary, 16–3
 - data interdependence, 16–3
 - extending a transaction boundary using table I/O, 16–16
 - extending between forms, 16–9
 - extending using business functions, 16–13
 - pessimistic recording locking within transaction boundary, 17–5
- Transaction master business function, Components, 13–57
- Transaction master business function modules, naming, 13–57
 - begin document, 13–58
 - cancel document (optional), 13–65
 - clear cache, 13–64
 - Edit document, 13–62
 - edit line, 13–60
 - end document, 13–63
- Transaction master business functions, 13–55
 - building, 13–68
 - implementing, 13–69
 - document processing, 13–70
 - single record processing, 13–69
- Transaction processing
 - commits and rollbacks, 16–1
 - data interdependence, 16–3
 - defining for a report, 16–18
 - defining transaction processing for a form, 16–8
 - extending a transaction boundary, 16–9
 - remote business functions, 16–6
 - system functions, 16–6
 - transaction boundaries, 16–3
- Transaction processing and business functions, 16–5
- Transaction processing in remote business functions, 16–6
- Transaction processing performance, 25–15
- Transaction processing scenarios, 16–4
- Transaction processing system functions, 16–6
- Translated text, adding a processing template with translated text, 15–15
- Tree
 - controls, 10–67
 - loading nodes, 10–82
 - node is selected, 10–85
- Tree node expand, data retrieval, Parent/Child form, B–11
- Trig functions, advanced functions, expression manager, 12–90
- Trigger
 - build triggers option, 18–3
 - creating a currency conversion trigger, 18–8
- Triggers, 25–4
 - attaching default triggers, 4–21, 4–22
 - default triggers, 4–5
 - default value trigger, 4–23
 - display rule trigger, 4–29
 - edit rule trigger, 4–26
 - next number trigger, 4–35
 - override
 - decimal, 10–108
 - default value, 10–101
 - edit rule, 10–104
 - formatting, 10–103

- next number, 10-106
 - previously defined, 10-99
 - styles, 10-107
 - user defined codes, 10-105
 - visual assist, 10-102
 - smart field trigger, 4-36
 - user defined codes trigger, 4-31
 - visual assist trigger, 4-24
 - Turning on debug tracing, 23-29
 - Two-phase commit (manual commit mode), 16-2
 - Type definitions, displaying type definitions, 7-5
 - Types of JDECACHE cursor APIs, 14-17
 - Typical, event flow, Find/Browse form, 12-12
 - Typical users and hook up, begin document, 13-61
- ## U
- UDC edit controls
 - creating, 10-58
 - defining options, 10-88
 - Understanding
 - business function builder form, 13-37
 - C business functions, 13-13
 - controls, 12-2
 - database communications steps, 13-26
 - event rules, 12-3
 - event rules buttons, 12-4
 - events, 12-2
 - form processing, 12-2
 - form types, 10-5
 - portability and standards, 13-24
 - standards and portability, 13-24
 - Understanding application design, 9-3
 - Understanding batch applications, 9-4
 - Understanding batch error messaging, 22-3
 - Understanding default triggers, 4-5
 - Understanding error handling, 21-5
 - Understanding how level break messages work, 22-6
 - Understanding how OneWorld stores objects, 1-7
 - Understanding how to build an application, 1-5
 - Understanding interactive applications, 9-3
 - Understanding objects and applications, 1-3
 - Understanding OneWorld modification rules, A-4
 - Understanding OneWorld tools, 1-8
 - Understanding the development cycle, 1-6
 - Understanding the event rules debugger, 23-5
 - Understanding web applications, 9-4
 - Unformatted data, 5-22
 - Unions, business view performance, performance tips, 25-9
 - Universal table browser
 - formatted data, 5-23
 - unformatted data, 5-22
 - Updating an object to match another object, 3-35
 - Updating data items for languages, 4-46
 - Updating different objects in different releases, 3-36
 - Updating records, 14-18
 - Updating the glossary, 4-40
 - adding glossary text for languages, 4-41
 - Useful features of the Visual C++ debugger, 23-19
 - User
 - add to project, 3-43
 - delete from project, 3-44
 - search, 3-41
 - view and maintain status, 3-44
 - User defined codes trigger, 4-31
 - overriding, 10-105
 - User defined items, menu/toolbar items
 - Find/Browse form, B-7
 - Fix/Inspect form, B-18
 - Header Detail form, B-26
 - Headerless Detail form, B-33
 - Parent/Child form, B-14
 - Search/Select form, B-38
 - User generated data structures, 7-1
 - media object data structures, 7-1
 - processing options data structures, 7-1
 - User Role, 3-4
 - Using
 - database APIs, 13-23
 - utility programs, 13-42
 - Using advanced get, 3-31
 - Using an index to access the cache, 14-11
 - Using pessimistic record locking within a transaction boundary, 17-5
 - Using Quick Form, 10-113

Using select distinct, 6–14
 Using standard processing, media objects, 10–116
 Using the data dictionary, 4–7
 Using the jdeCacheInit/jdeCacheTerminate rule, 14–14
 Using the same cache in multiple business functions or forms, 14–14
 Using watch, 23–20
 Utility programs, using, 13–42

V

Valid mapping operators, Table I/O, 12–92
 Validating, event rules, 12–39
 Validation, 25–5
 Value, assign, 12–85
 Variable selection and display window, 23–7
 Variables
 Grid, 12–49
 text, creating, 10–109
 Version, adding an interactive version, 9–6
 Versions modification rules, A–19
 View
 object status, 3–5
 project status, 3–5
 user status, 3–44
 Viewing, documentation
 business function - values to pass, 13–86
 business function document viewer, 13–87
 business function search, 13–86
 Viewing attachments in Design view, 3–45
 Viewing attachments in the Object Management Workbench, 3–46
 Viewing field relationships, 8–14
 Viewing projects, 3–13
 Visual assist trigger, 4–24
 overriding, 10–102

W

Web
 adding or changing web address on OneWorld Explorer Help, 19–21

 creating a web view subheading on a menu, 19–22
 Web applications, 9–4
 generating a form, 24–7
 Java generator, 9–4
 setting language, 24–6
 Web-based applications, required components, 24–1
 Java application, 24–3
 While statements, event rule logic, 12–44
 Windows application, add a Windows application to a menu, 19–19
 Work center API, 22–7
 Work center initialization API, calling the work center initialization API, 22–18
 Work center process, terminating the work center process, 22–23
 Workflow messages, 21–2
 Working with
 BrowsER, 12–117
 BrowsER options, 12–120
 business function builder, 13–37
 business functions, 13–31
 controls, 10–43
 Event Rule Design, 12–29
 event rule variables, 12–49
 if and while statements, 12–43
 Working with asynchronous processing, 12–111
 Working with error messages, 21–15
 Working with table design, 5–9
 Working with the cross reference facility, 8–3
 Working with the Data Reference Scan, C–3
 Working with the Send Message System Function, 21–27
 Working with the Tip of the Day utility, 20–3
 Workstation, jde.ini, setting, 16–26
 Workstation jde.ini, record change detection, 17–3
 WorldVision applications, add a WorldVision application to a menu, 19–18
 Write grid line after, runtime processing, 12–21
 Write grid line before, runtime processing, 12–19

