

Guide
September 2000



J D E D W A R D S

Development Tools

Release
OneWorld® Xe



J.D. Edwards World Source Company
One Technology Way
Denver, CO 80237

Copyright ©J.D. Edwards World Source Company, 1997–2000
All Rights Reserved

J.D. Edwards is a registered trademark of J.D. Edwards & Company. The names of all other products and services of J.D. Edwards used herein are trademarks or registered trademarks of J.D. Edwards World Source Company.

All other product names used are trademarks or registered trademarks of their respective owners.

The information in this guide is confidential and a proprietary trade secret of J.D. Edwards World Source Company. It may not be copied, distributed, or disclosed without prior written permission. This guide is subject to change without notice and does not represent a commitment on the part of J.D. Edwards & Company and/or its subsidiaries. The software described in this guide is furnished under a license agreement and may be used or copied only in accordance with the terms of the agreement. J.D. Edwards World Source Company uses automatic software disabling routines to monitor the license agreement. For more details about these routines, please refer to the technical product documentation.

This publication is not sponsored or endorsed by or affiliated with Microsoft Corporation. Screen examples are reprinted by permission from Microsoft Corporation.

OneWorld Development Tools Guide

The OneWorld *Development Tools* guide consists of several units, including.

- Fundamentals
- Application Design
- Event Rules
- Business Functions
- Caching
- Additional Features
- Messaging
- Debugging
- Web Applications
- Performance
- Appendices

Fundamentals

Fundamentals covers the basic concepts and tools you need to know to start developing applications. It includes the preliminary steps you must take before you actually design an application. Fundamentals includes the following sections:

- Object Management Workbench
- Data Dictionary
- Table Design
- Business View Design
- Data Structures
- Cross Reference
- Checkout Log

Application Design

Application Design includes the steps you follow to actually design an application. It describes different form types and controls. It includes the following sections:

- Application Design
- Forms Design

Event Rules

Event Rules discusses using event rules and logic. It also describes how to use BrowsER. It includes the following sections:

- Event Rules Design

Business Functions

Business Functions discusses both C business functions and named event rules. It also has chapters on master business functions, using Business Function Builder, and Business Function Documentation. It includes the following sections:

- Business Functions

Caching

Caching describes how you can use caching for better performance. It includes the following sections:

- Caching

Additional Features

Additional Features discusses additional features of the OneWorld development tools that you can use to enhance your applications. It includes the following sections:

- Processing Options
- Transaction Processing
- Record Locking
- Currency
- Menu Design

Messaging

Messaging discusses error handling, interactive messaging, and batch messaging. It includes the following sections:

- Messaging
- Batch Error Messages

Debugging

Debugging provides information to help you debug your applications. It includes different methods of debugging and strategies to help you. It includes the following sections:

- Debugging

Web Applicaitons

Web Applications discusses designing and generating web applications. It includes the following sections:

- Developing Web Applications
- Generating Web Applications

Performance

Performance discusses troubleshooting your application and various performance issues. It includes tips to help you get better performance from your applications. It includes the following sections:

- Performance

Appendices

The appendices include information about making software modifications and detailed information about the process flow for different form types. This unit includes the following appendices:

- OneWorld Modification Rules
- Form Processing
- Date Reference Scan

OneWorld Acronyms

Following are some acryonyms that are commonly used in OneWorld.

APPL	Application
BDA	Business View Design Aid
BSFN	Business Function
BSVW	Business View
CRP	Conference Room Pilot
DD	Data Dictionary
DLL	Dynamic Link Library
DS or DSTR	Data Structure
ER	Event Rules
FDA	Form Design Aid
NER	Named Event Rules
OCM	Object Configuration Manager
OL	Object Librarian
OMW	Object Management Workbench
QBE	Query by Example
RDA	Report Design Aid
SAR	Software Action Request
Specs	Specifications
TAM	Table Access Management

TBLE	Table
TC	Table Conversion
TDA	Table Design Aid
TER	Table Event Rules
UBE	Universal Batch Engine
UTB	Universal Table Browser
WF	Work Flow



Table of Contents

OneWorld Development Tools Guide	1
OneWorld Tools	1-1
Understanding Objects and Applications	1-3
Understanding an Object	1-3
Understanding an Application	1-4
Understanding How to Build an Application	1-5
Understanding the Development Cycle	1-6
Understanding How OneWorld Stores Objects	1-7
Understanding the OneWorld Toolset	1-8
Object Management Workbench	1-8
Data Dictionary	1-8
Table Design	1-9
Business View Design	1-9
Form Design	1-9
Report Design	1-10
Business Function Design	1-10
Event Rules	1-10
Processing Options	1-12
Menus	1-12
How OneWorld Works at Design Time	1-13

Fundamentals

Object Management Workbench	2-1
Understanding the Object Management Workbench	2-3
Object Management Workbench Concepts	2-3
Projects	2-3
Default Project	2-4
User Roles	2-4
Allowed Actions	2-4
Tokens	2-4
Object Management Workbench Interface	2-5
Object Librarian and Non-Object Librarian Objects	2-6
Working with the Object Management Workbench	2-9
To Use Your Default Project	2-9
Understanding Project Life Cycle	2-9
Working with Projects	2-13
Viewing Projects in the Project Window	2-13
Filtering Projects	2-13
Searching for Projects	2-15

Creating New Projects	2-17
Changing Project Properties	2-19
Advancing Projects	2-20
Adding Existing Projects to a Project	2-21
Deleting Projects	2-22
Working with Objects	2-23
Creating Objects	2-23
Searching for Objects	2-25
Adding Objects to Projects	2-27
Moving Objects	2-29
Removing Objects from Projects	2-30
Deleting Objects	2-30
Getting Object Specifications	2-31
Checking Objects In and Out	2-32
Changing Objects	2-34
Maintaining Objects in Multiple Software Releases	2-35
Working with Tokens	2-37
Understanding the Token Queue	2-38
Inheriting Tokens	2-38
Switching Tokens	2-39
Releasing Tokens Manually	2-40
Working with Users	2-41
Searching for Users	2-41
Adding Users to Projects	2-43
Removing Users from Projects	2-44
Changing User Properties	2-44
Working with Attachments	2-45
Data Dictionary	3-1
Understanding the Data Dictionary	3-3
How the Data Dictionary is Used at Runtime	3-3
Naming Data Items	3-4
Storing the Data Dictionary and Data Dictionary Items	3-4
Glossary Items	3-5
Error Messages	3-5
Understanding Default Triggers	3-5
Using the Data Dictionary	3-7
Defining a Data Item	3-11
Creating a Data Item	3-11
Naming a Data Item	3-12
Data Dictionary Item Naming Conventions	3-12
Defining General Information	3-16
Attaching Default Triggers	3-21
Attaching a Default Value Trigger	3-23
Attaching a Visual Assist Trigger	3-24
Attaching an Edit Rule Trigger	3-26
Attaching a Display Rule Trigger	3-29
Attaching a User Defined Code Trigger	3-31
Attaching a Next Number Trigger	3-35
Attaching a Smart Field Trigger	3-36
Updating the Glossary	3-40

Defining Jargon and Alternate Language Terms	3-44
Table Design	4-1
Adding a Table	4-3
Indices	4-5
Working with Table Design	4-9
Choosing Data Items for the Table	4-10
Defining Indices	4-10
Previewing Tables	4-12
Working with Tables	4-15
Generating Tables	4-15
Generating Indexes	4-16
Generating Header Files	4-16
Copying Tables	4-16
Removing Tables from the Database	4-17
Viewing the Data in Tables	4-19
Example: Universal Table Browser (Unformatted Data)	4-22
Example: Universal Table Browser (Formatted Data)	4-23
Example: Column Properties	4-24
Business View Design	5-1
Table Join	5-1
Table Union	5-2
Select Distinct	5-2
Indices	5-3
Adding a Business View	5-5
Joined Views	5-7
Working with Business View Design	5-9
Launching the Business View Design Aid	5-10
Choosing a Table for a Business View	5-11
Choosing Data Items for a Business View	5-12
Using Select Distinct	5-14
Creating a Table Join	5-17
Creating a Table Union	5-19
Data Structures	6-1
System Generated Data Structures	6-1
User Generated Data Structures	6-1
Working with Interconnection Data Structures	6-3
Changing Form Data Structures	6-3
Changing Report Data Structures	6-4
Example: Changing Interconnection Data Structures	6-5
Displaying Type Definitions	6-5
Creating a Data Structure	6-7
Creating a Media Object Data Structure	6-7
Creating a Processing Options Data Structure	6-9
Creating a Business Function Data Structure	6-10
Defining a Data Structure	6-13
Launching Data Structure Design	6-13
Designing a Data Structure	6-14
Creating a Type Definition	6-15

Cross Reference Facility	7-1
Working with the Cross Reference Facility	7-3
Searching for Objects	7-3
Searching for Data Items	7-4
Searching for Interactive Applications	7-5
Searching for Batch Applications	7-6
Searching for Business Functions	7-7
Searching for Business Views	7-8
Searching for Data Structures	7-9
Searching for Tables	7-10
Searching for Forms	7-11
Searching for Event Rules	7-12
Viewing Field Relationships	7-14
Rebuilding Cross Reference Information	7-15

Application Design

Application Design	8-1
Understanding Application Design	8-3
Understanding Interactive Applications	8-3
Understanding Batch Applications	8-4
Understanding Web Applications	8-4
Adding an Interactive Application	8-5
Create an Interactive Application Object	8-5
Creating an Interactive Version Object	8-6
Forms Design	9-1
What is a Form?	9-2
Elements of a Form	9-3
Understanding Form Types	9-5
About J.D. Edwards Design Standards	9-5
Find/Browse Forms	9-6
Fix/Inspect Forms	9-7
Header Detail Forms	9-9
Headerless Detail Forms	9-10
Search and Select Forms	9-12
Message Forms	9-13
Parent/Child Forms	9-14
Creating a Form	9-17
Select a Form Type	9-17
Define Form Properties	9-19
Select a Business View	9-22
Revise Information about a Form	9-23
Delete a Form	9-24
Designing a Form Layout	9-25
Selecting and Moving Controls	9-25
Changing the Size of Grids, and Controls	9-26
Using the Cut, Copy, and Paste Commands in Forms	9-28
Aligning Controls	9-29

Using Undo and Redo	9-31
Working with Menu/Toolbar Exits	9-33
Creating, Editing, or Deleting a Menu/Toolbar Exit	9-33
Adding Bitmaps	9-36
Attaching Event Rules to a Menu/Toolbar Exit	9-38
Creating Subcategories in Menus	9-39
Working with Controls	9-41
Understanding Form Controls	9-42
Creating Push Buttons	9-43
Creating Check Boxes	9-44
Creating Radio Buttons	9-46
Adding Data Items to a Form	9-47
Inserting Database Items Onto a Form	9-48
Inserting Data Dictionary Items Onto a Form	9-49
Creating Static Text Controls	9-50
Creating Edit Controls	9-51
Creating UDC Edit Controls	9-56
Creating a UDC Edit Control	9-56
Disconnecting Static Text from Controls	9-57
Creating Media Object Controls	9-58
Creating Bitmap Controls	9-63
Creating a Tree Control	9-65
Creating Combo Boxes	9-65
Creating Text Boxes	9-66
Creating Group Boxes	9-67
Defining Grid Properties	9-69
Defining Grid Column Properties	9-72
Defining Properties for Parent/Child Controls	9-74
Defining Options for Forms, Grid, and Edit Controls	9-84
Defining Options for a Form	9-84
Defining Options For a Grid	9-85
Defining Options For an Edit or UDC Edit Control	9-86
Associating Database Items, Dictionary Items, or Descriptions	9-87
Changing Tab Sequence	9-89
Creating Tab Controls	9-90
Designing Forms Using Multiple Modes	9-93
Viewing Forms in a Particular Mode	9-94
Setting Control Mode Properties	9-95
Overriding Data Dictionary Triggers at Design Time	9-97
Using Text Variables	9-107
Adding a Text Variable	9-107
Attaching or Assigning a Text Variable	9-108
Using Quick Form	9-111
Processing Media Objects	9-113
Imaging	9-114
Using Standard Processing for Media Objects	9-114
Language Considerations for Media Objects	9-118
Testing a Form	9-123
Language Preview	9-123

ActivEra Portal Design	10-1
Planning Portal Components	10-3
Building HTML Components	10-7
Creating an HTML Component	10-7
Example: Designing an HTML Component	10-12
Creating the Component	10-12
Adding JavaScript for the Personalization Mode	10-13
Building URI Components	10-15
Creating a URI Component	10-15
Calling URLs	10-20
Wrapping URIs	10-20
Obtaining User Information	10-22
Example: Designing a URI Component	10-22
Before You Begin	10-23
Creating the Component	10-23
Wrapping URLs in Personalization Mode	10-26
Building OneWorld Components	10-29
Creating a OneWorld Component	10-30
Building Java Applet Components	10-35
Building Java Servlet Components	10-43
Understanding the ComponentServlet Class	10-44
Extending the Component Servlet	10-45
Understanding the loadTemplate() Method	10-46
Understanding the OwpWrapForm Script	10-51
Understanding the OwpWrapPage Script	10-52
Understanding the OwpTargetedPage Script	10-53
Extending the OneWorldComponentServlet Class	10-54
Understanding the ComponentTemplateParser Class	10-56
Constructing a Template Parser	10-56
Understanding the toString Method	10-57
Understanding the setString Method	10-57
Understanding the retrieve Method	10-57
Using the encompass Method	10-58
Understanding the replace Method	10-58
Understanding the replaceTag Method	10-60
Understanding the parseKey Method	10-60
Understanding the ComponentDatabaseInterface Class	10-61
Constructing a Database Interface	10-61
Using the Public Fields	10-62
Understanding the getComponentName Method	10-62
Understanding the setComponentName Method	10-63
Understanding the getData Method	10-63
Understanding the setData Method	10-64
Understanding the getDataInputStream Method	10-64
Understanding the getDataOutputStream Method	10-65
Understanding the getDefaultUserId Method	10-65
Understanding the setDefaultUserId Method	10-65
Understanding the Link Center Component	10-67
Understanding Portal Design Standards	10-69
Understanding Portal Style Sheets	10-69

Event Rules

Event Rules Design	11-1
Understanding Controls	11-2
Understanding Events	11-2
Understanding Form Processing	11-2
Understanding Event Rules	11-3
Business Function Event Rules	11-3
Embedded Event Rules	11-4
Application Event Rules	11-4
Table Event Rules	11-4
Understanding the Event Rule Buttons	11-4
Event Rules Based on Form Type	11-5
Find/Browse	11-5
Form-Level Events	11-5
Grid-Level Events	11-5
Fix/Inspect	11-5
Header Detail and Headerless Detail	11-5
Form-Level Events	11-5
Grid-Level Events	11-5
Runtime Processing	11-7
Runtime Data Structures	11-7
Available Objects and Runtime Data Structures	11-7
Event Rule Manipulation	11-10
Processing Available Objects	11-10
Control is Exited Processing	11-10
Grid Processing	11-11
Form Flow	11-12
Typical Event Flow for a Find/Browse Form	11-12
Pre Dialog Is Initialized	11-13
Dialog Is Initialized	11-14
Post Dialog Is Initialized	11-15
SQL SELECT	11-16
Do in While Loop	11-16
Page-at-a-Time Processing	11-16
BC Assigned Database Values	11-17
Grid Rec Is Fetched	11-18
Write Grid Line Before	11-19
Write Grid Line After	11-21
Last Grid Rec Has Been Read	11-22
Select Button Processing	11-23
Button Clicked	11-24
Add Button Processing	11-25
Delete Button Processing	11-26
Delete Grid Rec Verify Before	11-26
Delete Grid Rec Verify After	11-27
Delete Grid Rec From DB Before	11-27
Delete Grid Rec From DB After	11-28
All Grid Recs Deleted From DB	11-28

Working with Event Rules Design	11-29
Creating and Saving Event Rules	11-30
Finding Event Rules	11-32
Cutting, Copying, and Pasting Event Rules	11-33
Adding Comment Lines to Event Rules	11-36
Printing Event Rule Logic	11-37
Validating Event Rules	11-39
Working with If and While Statements	11-43
Creating If and While Statements	11-44
Defining Literal Values in Event Rule Logic	11-47
Working with Event Rule Variables	11-49
Interactive Event Rule Variables	11-49
Batch Application Event Rule Variables	11-50
Creating an Event Rule Variable	11-50
Using Event Rule Variables for Automatic Line Numbering	11-55
Attaching Functions	11-57
Attaching a System Function to an Event	11-57
Common System Functions	11-59
Attaching a Business Function to an Event	11-65
Creating Form Interconnections	11-71
Creating a Modal Form Interconnection	11-71
Creating a Modeless Form Interconnection	11-75
Creating Report Interconnections	11-81
Creating Assignments	11-85
Expression Manager	11-87
Table I/O	11-91
Available Operations	11-91
Valid Mapping Operators	11-92
Creating a Table I/O Event Rule	11-92
Understanding Buffered Inserts	11-95
Buffered Insert Error Messaging	11-95
Using Buffered Inserts in Table I/O	11-96
Understanding Handles	11-97
Using a Handle	11-98
Table Event Rules	11-103
Creating Table Event Rules	11-104
Creating Dynamic Overrides	11-109
Working with Asynchronous Processing	11-111
Asynchronous Events	11-112
Control is Exited and Changed - Asynch	11-113
Row is Exited and Changed - Asynch	11-113
Column is Exited and Changed - Asynch	11-113
Asynchronous Business Functions	11-114
Example: Asynchronous Event Processing	11-115
Working with BrowsER	11-117
Working with BrowsER	11-117
Using Visual ER Compare	11-123
Using Visual ER Compare	11-123
Launching Visual ER Compare	11-123
Understanding the Visual ER Compare Interface	11-124

Working with Visual ER Compare	11-125
Changing Your Target ER	11-125
Printing a Visual ER Compare Report	11-126
Using AutoMerge	11-126

Business Functions

Business Functions	12-1
Business Function Features	12-2
What are the Components of a Business Function?	12-3
How Distributed Business Functions Work	12-5
Creating Business Function Event Rules	12-7
Understanding C Business Functions	12-13
Understanding Header File Sections	12-14
Business Function Header File Example	12-15
Understanding the Structure of a Business Function Source File ...	12-18
Example: Business Function Source File Check for In Add Mode	12-18
Using Application Programming Interfaces (APIs)	12-21
Using Common Library APIs	12-21
MATH_NUMERIC Data Type	12-22
JDEDATE Data Type	12-22
Using Database APIs	12-23
Understanding Standards and Portability	12-24
J.D. Edwards Open Database Connectivity (ODBC)	12-24
Standard JDEBASE API Categories	12-25
Connecting to a Database	12-26
Understanding Database Communication Steps	12-26
Calling an API from an External Business Function	12-27
Stdcall Calling Convention	12-27
Cded Calling Convention	12-28
Calling a Visual Basic Program from OneWorld	12-29
Working with Business Functions	12-31
Launching Business Function Design	12-31
Viewing Object Codes	12-32
Adding Related Tables and Functions	12-32
Debugging Business Functions	12-33
Changing a Business Function Parent DLL or Location	12-33
Creating and Specifying Custom DLLs	12-35
Creating a Custom DLL	12-35
Specifying a Custom DLL for a Custom Business Function	12-36
Working with Business Function Builder	12-37
Understanding the Business Function Builder Form	12-37
Functions List	12-38
Available Business Functions	12-38
Build Output	12-38
Building a Single Business Function	12-39
Linking Business Function Objects	12-40
Setting Build Options	12-40
Using Utility Programs	12-42

Reading Build Output	12-42
Makefile Section	12-43
Begin DLL Section	12-43
Compile Section	12-43
Link Section	12-44
Rebase Section	12-44
Summary Section	12-45
Building All Business Functions	12-45
Reviewing Error Output	12-52
Resolving Errors	12-52
Transaction Master Business Functions	12-55
Master Business Function Naming Convention	12-55
Creating Processing Options for a Master Business Function	12-56
Naming Transaction Master Business Function Modules	12-57
Components of a Transaction Master Business Function	12-57
Begin Document	12-58
Edit Line	12-60
Edit Document	12-62
End Document	12-63
Clear Cache	12-64
Cancel Document (optional)	12-65
Cache Structure	12-66
Building Transaction Master Business Functions	12-68
Implementing Transaction Master Business Functions	12-69
Single Record Processing	12-69
Document Processing	12-70
Master File Master Business Functions	12-71
Information Structure	12-74
Performance Impact	12-75
Business Function Documentation	12-77
Creating Business Function Documentation	12-77
Generating Business Function Documentation	12-81
Viewing Business Function Documentation	12-85
Understanding Business Function Processing Failovers	12-89

Caching

Caching	13-1
Understanding JDECACHE	13-3
When to Use JDECACHE	13-3
Performance Considerations	13-4
The JDECACHE API Set	13-4
JDECACHE Management APIs	13-4
JDECACHE Manipulation APIs	13-5
Working with JDECACHE	13-7
Calling JDECACHE APIs	13-7
Setting Up Indices	13-8
Initializing the Cache	13-10
Using an Index to Access the Cache	13-11

Using the jdeCacheInit/jdeCacheTerminate Rule	13-14
Using the Same Cache in Multiple Business Functions or Forms ...	13-14
JDECACHE Cursors	13-15
Opening a JDECACHE Cursor	13-15
HJDECURSOR	13-15
JDECACHE Dataset	13-16
Types of JDECACHE Cursors APIs	13-17
Updating Records	13-18
Deleting Records	13-19
The jdeCacheFetchPosition API	13-19
The jdeCacheFetchPositionByRef API	13-20
Resetting the Cursor	13-20
Closing a Cursor	13-20
JDECACHE Multiple Cursor Support	13-20
JDECACHE Partial Keys	13-21
How a JDECACHE Partial Key Works	13-22
JDECACHE Errors	13-23
JDECACHE Example Program	13-25
JDECACHE Standards	13-35
Cache Programming Standards	13-36
Individual Cache Business Function Header File	13-40

Additional Features

Processing Options	14-1
Processing Options Templates	14-2
Defining a Processing Options Data Structure (Template)	14-5
Defining a Template	14-5
J. D. Edwards Processing Option Naming Standards	14-9
Processing Option Data Structure	14-9
Tab Title	14-9
Comment	14-10
Data Item	14-11
Language Considerations for Processing Options	14-12
Attaching a Processing Options Template	14-17
Transaction Processing	15-1
Commits and Rollbacks	15-1
Understanding OneWorld Transaction Processing	15-3
Transaction Processing Scenarios	15-4
Transaction Processing and Business Functions	15-5
Transaction Processing in Remote Business Functions	15-6
Transaction Processing System Functions	15-6
Working with Transaction Processing	15-7
Defining Transaction Processing for a Form	15-8
Extending a Transaction Boundary	15-8
Defining Transaction Processing for a Report	15-17
Setting the jde.ini for Transaction Processing and Lock Manager	15-19
Understanding Concurrent Release Support	15-19

Understanding Transaction Processing Logging	15-20
Setting the jde.ini for Transaction Processing and Lock Manager ...	15-21
Settings for the [TP MONITOR ENVIRONMENT] Section (Prior to B73.3) .	15-21
Settings for the [LOCK MANAGER] Section (B73.3 and higher) .	15-26
Record Locking	16-1
Understanding Record Locking	16-3
Optimistic Locking	16-3
Pessimistic Locking	16-4
Currency	17-1
OneWorld Currency Implementation	17-1
Advantages	17-1
Working with Currency	17-3
Understanding the Build Triggers Option	17-3
Understanding How Table Event Rules Work with Currency Processing	17-4
Setting Up Currency Conversion	17-6
Showing Currency-Sensitive Controls	17-7
Creating a Currency Conversion Trigger	17-8
Menu Design	18-1
Understanding Menus	18-3
Menu Filtering	18-3
Menu Design Tables	18-4
Working with Menus	18-5
Defining a New Menu	18-5
Reviewing Selections for a Menu	18-8
Printing a Menu Report	18-8
Example: Print Menus Report	18-9
Working with Menu Selections	18-11
Adding or Changing a Menu Selection	18-11
Adding an Application to a Menu	18-14
Adding or Changing Web Addresses on OneWorld Explorer Help ..	18-21
Creating a Web View Subheading on a Menu	18-22
Linking Menus	18-22
Creating Fast Path Selections	18-23
Working with Menu Selection Revisions	18-27
Copying a Menu Selection	18-27
Changing Menu Text for Languages	18-28
Changing Menu Selection Text	18-30
Renumbering a Menu Selection	18-30

Tips of the Day	19-1
Working with Tips of the Day	19-3

Messaging

Messaging	20-1
Message Types	20-1
Error Messages	20-2
Workflow Messages	20-2
Level Messages	20-2
Information Messages	20-3
Understanding Error Handling	20-5
Event-Driven Model	20-5
When is an Error Set?	20-6
How Does the Application Know Which Field to Highlight? ...	20-6
How Is an Event ID Used for Error Setting?	20-6
Error Setting	20-7
Automatic Error Setting	20-7
Manual Error Setting	20-7
Resetting Errors	20-9
Resetting Errors on the OK Button	20-9
Resetting Errors on the Find Button	20-10
Multilevel Error Messaging for C Business Functions	20-11
Working with Error Messages	20-15
Locating an Existing Error Message	20-15
Creating a Simple Error Message	20-16
Adding an Interactive Message Data Item	20-17
Creating a Text Substitution Error Message	20-18
Adding an Interactive Message Data Item	20-19
Defining the Glossary Text for a Runtime Message	20-21
Attaching a Data Structure Template to a Message	20-22
Attaching an Interactive Message Data Item	20-23
Working with the Send Message System Function	20-27
Batch Error Messages	21-1
Understanding Batch Error Messaging	21-3
Level-Break Messages	21-3
Text Substitution Error Messages	21-5
Action Messages	21-6
Understanding How Level Break Messages Work	21-6
Work Center API	21-7
Creating a Level-Break Message	21-9
Creating a Data Dictionary Item for a Level-Break Message	21-9
Creating a Data Structure for the Data Dictionary Item	21-12
Creating a Level-Break Message Business Function Data Structure .	21-13
Creating a Level-Break Business Function	21-15
Sample Source Code Highlights	21-16
Calling the Work Center Initialization API	21-18
Calling the Processing Work Center APIs	21-21

Terminating the Work Center Process	21–23
---	-------

Debugging

Debugging	22–1
Overview of the Debugging Process	22–1
Interpretive and Compiled Code	22–3
Working with the Event Rules Debugger	22–5
Understanding the Event Rules Debugger	22–5
Object Browse Window	22–6
Event Rules Window	22–6
Variable Selection and Display Window	22–7
Breakpoint Manager	22–7
Search Combo Box	22–8
Debugging an Application	22–9
Inspecting or Modifying a Variable	22–10
Just in Time Debugging	22–11
Setting Breakpoints	22–11
Debugging Business Functions Using Microsoft Visual C++	22–13
Working with the Visual C++ Debugger	22–19
Useful Features of the Visual C++ Debugger	22–19
The Go Command	22–19
The Step Command	22–19
The Step Into Command	22–19
Setting Breakpoints	22–20
Using Watch	22–20
Locals Window	22–20
Example: Debugging with the Visual C++ Debugger	22–20
Customizing the Environment	22–21
Debugging Strategies	22–23
Is the Program Ending Unexpectedly?	22–23
Is the Application Encountering Errors?	22–24
Is the Output of the Program Incorrect?	22–24
Where Could the Problem Be Coming From?	22–24
Is the Function Being Called as Expected?	22–24
Debug Logs	22–25
SQL Log Tracing	22–26
Debug Tracing	22–29
Turning on Debug Tracing	22–29
System Function Tracing	22–30

Web Applications

Developing Web Applications	23–1
Understanding the HTML Client	23–5
Generating Web Applications	23–7
Logging in	23–8
Setting Generator Options	23–8

Generating Forms	23-9
Generating Reports	23-11
Generating All Objects	23-12
Generating Other Objects	23-12

Performance

Performance	24-1
Performance Tips	24-3
Things to Look For	24-3
Data Dictionary Performance	24-4
Triggers	24-4
Overrides	24-4
Validation	24-5
Table Design Performance	24-5
Index Limitations for Various Databases	24-6
Access32	24-6
SQLSERVER	24-6
DB2 for OS/400	24-7
Oracle	24-7
Key Column Violation	24-7
Specification File Corruption	24-8
Table I/O Objects	24-8
Business View Performance	24-9
Data Structure Performance	24-10
Data Selection and Sequencing	24-10
Form Design	24-11
Find/Browse	24-11
Header Detail and Headerless Detail	24-12
Batch Application Performance	24-12
Event Rules Performance	24-12
Business Function Performance	24-14
Error Messaging Performance	24-15
Transaction Processing Performance	24-15

Appendices

Appendix A: OneWorld Modification Rules	A-1
What an Upgrade Preserves and Replaces	A-3
General Rules for Modification	A-4
Interactive Applications	A-4
Reports	A-6
Application Text Changes	A-7
Table Specifications	A-8
Control Tables	A-9
Business Views	A-10
Event Rules	A-10
Data Structures	A-11

Business Functions	A-13
Versions	A-13
Appendix B: Form Processing	B-1
Process Flow for Find/Browse Form	B-3
Process Flow for Parent/Child Form	B-9
Process Flow for Fix/Inspect Form	B-15
Process Flow for Header Detail Form	B-19
Process Flow for Headerless Detail Form	B-27
Process Flow for Search/Select Form	B-35
Process Flow for Message Form	B-39
Process Flow for Edit Control	B-41
Process Flow for Grid Control	B-45
Appendix C: Date Reference Scan	C-1
Working with the Date Reference Scan	C-3
Business Function Date Reference Scan	C-3
Event Rule Date Reference Scan	C-4

Glossary

Index



OneWorld Tools

The OneWorld Tools are an integrated set of application development tools. These tools allow business analysts to develop complete interactive or batch applications (for example forms and reports). The tools simplify the development process and limit the amount of programming necessary to create applications.

OneWorld Tools also allow you to create applications for client/server environments using the stability of J.D. Edwards methodology and the ease of the Microsoft Windows interface.

OneWorld tools is composed of the following topics:

- Understanding objects and applications
- Understanding how to build an application



Understanding Objects and Applications

J.D. Edwards uses OneWorld Tools to build objects, which are used to build applications.

Understanding objects and applications is composed of the following topics:

- Understanding an object
- Understanding an application

Understanding an Object

By industry standards, an object is a self-sufficient entity that contains data as well as the structures and functions used to manipulate that data. In OneWorld, an object is a reusable entity that is based on software specifications created by the OneWorld Tools.

A specification is a complete description of a OneWorld object. Each object has its own specification, which is stored on both the server and the workstation. Some specifications describe different types of objects. For example, the data structure specification is used to describe a business function data structure, a processing option structure and a media object structure.

The OneWorld architecture is object-based. This means that discrete software objects are the building blocks for all applications, and that developers can reuse the objects in multiple applications. It is this use of objects (applications being broken down into smaller components) that allows J.D. Edwards to provide true distributed processing. Developers create the objects using OneWorld Tools.

Examples of OneWorld objects include the following items:

- Batch applications
- Business functions (encapsulated routines)
- Business views
- Data dictionary items
- Data structures
- Event rules
- Interactive applications

- Media objects
- Tables

Understanding an Application

An application is a collection of objects that performs a specific task. J.D. Edwards uses the OneWorld Tools to build its standard groups of related applications:

- Architecture, engineering, and construction
- Distribution
- Energy and chemical systems
- Financial
- Human resources
- Manufacturing
- Technical

These applications share a common user interface because they are all generated through OneWorld Tools. Applications refer to both interactive and batch applications. For example, all of the following are applications:

- Address Book Revisions
- Sales Order Entry
- General Ledger Post
- Trial Balance Report

Understanding How to Build an Application

You can use the OneWorld Tools to build your applications. This chapter briefly describes the development cycle and each OneWorld tool.

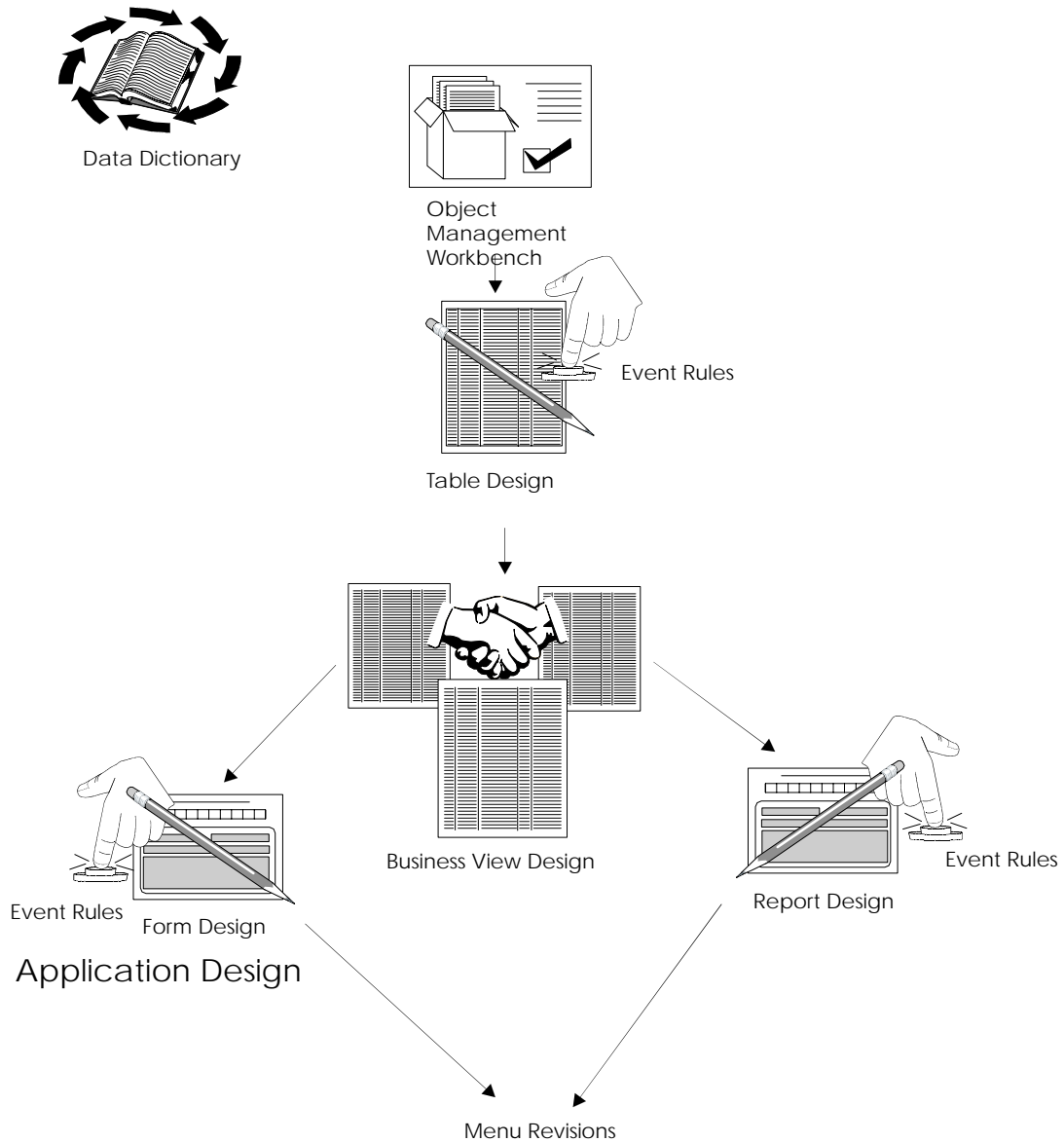
You might not need to use every tool to create an application; however, you always begin your application development from the Object Management Workbench. For example, you might not need to add or modify data items. If so, you can proceed to Table Design from the Object Management Workbench. If one or more existing database tables already contain all of the data items you want in your application, then you can skip the step of designing a table and proceed to Business View Design.

Understanding how to build an application is composed of the following topics:

- Understanding the development cycle
- Understanding how OneWorld stores objects
- Understanding OneWorld Tools

Understanding the Development Cycle

The following figure illustrates the software development cycle, showing relationships between the tools.



Understanding How OneWorld Stores Objects

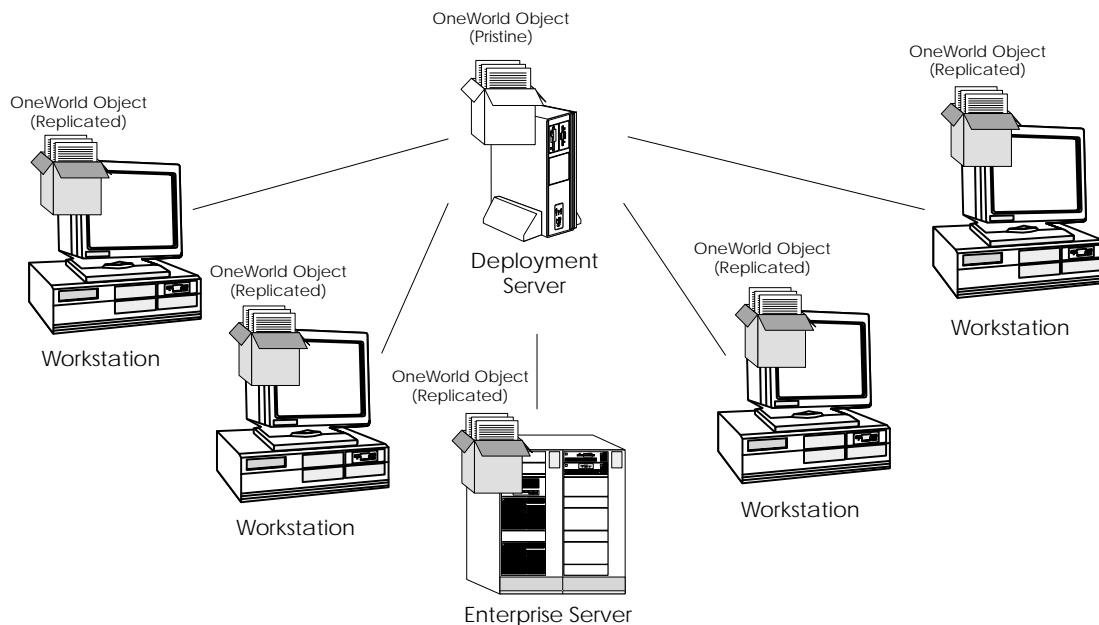
OneWorld stores objects in two places:

- A “central-storage” server, which stores central objects.

Central objects reside in a central location where you can deploy them. Objects, such as specifications, are stored in a relational database. Other objects, such as DLLs and source code, are stored on a file server.

- On any machine (workstation or server) that runs OneWorld and stores replicated objects.

A copy (replicated) set of the central objects must reside on each workstation and server that run OneWorld. The path code indicates the directory where these objects are located.



You move objects between the server and workstation using the design tool for the specific object type, using check in and check out. When you create an object, it resides initially on your workstation. Unless you check it into the server, it is available only to you. Once checked into the server, it is available for check out to other users.

When you check out an object, all object specification records (a collection of data that defines a OneWorld object) are replicated from the server to your workstation.

Understanding the OneWorld Toolset

The OneWorld Toolset is composed of several tools that you use to create an interactive or batch application:

- Object Management Workbench
- Data Dictionary
- Table Design
- Business View Design
- Form Design
- Report Design
- Business Function Design
- Event Rules
- Processing Options
- Menus

Object Management Workbench

The OneWorld Object Management Workbench manages all objects. Developers use the Object Management Workbench to check out objects from a central development environment, copying the objects to their desktop. They can then use the tools to change objects and check the objects back in for others to access. Developers can access only OneWorld Tools through the Object Management Workbench.

Data Dictionary

Just as a dictionary contains word definitions, the J.D. Edwards data dictionary is a central repository that contains data item definitions and attributes. These attributes determine how a data item:

- Appears on reports and forms (such as number of decimals and default values)
- Validates data entry within an application
- Assigns column and row descriptions
- Provides text for field-sensitive help
- Is stored in a table

The data dictionary is active because changes are automatically reflected in applications without having to recompile software.

Table Design

A relational database table stores data that an application uses. You can create one or more tables for use in an application. To create a table, select data items for a table, and assign key fields as indices for retrieving and updating data.

Business View Design

Business views are the link between applications and data. A business view defines the data items from one or more tables that an application uses. After you create the tables, select the data items from one or more tables that you want in your business view. With business views, you can select only the columns needed in the application, which increases performance due to less data moving over the network. For example, you could create a business view that contains only employee names and addresses from a table containing all employee data.

Form Design

An interactive application is the bridge between a user and a database (based on a business view). A user accesses an interactive application to add, modify, or view data using a form.

To create an application, determine the type of form your application requires, and associate each form with a business view. To design forms, you add controls such as a grid, edit fields, push buttons, and radio buttons.

There are several standard form types that have much of the required processing already established. Examples include:

Find/Browse	Used for inquiry forms, such as Work with Sales Orders
Fix/Inspect	Used to add or modify a single record, such as Sales Order Header
Header Detail	Used to modify multiple records at a time (particularly on normalized tables), such as Sales Order Detail
Headerless Detail	Used to modify multiple records in a table that is not normalized, such as Voucher Entry or Bill of Material
Search and Select	Used to automatically retrieve data into a visual assist field

Parent/Child	Used to display multiple records that have a parent/child relationship in an Explorer-like tree view, for example Bill of Material
Message	Used to display messages or request action, for example Delete Confirmation

Report Design

You use Report Design to create reports and batch processes. In Report Design, you design sections instead of forms. To create a report, determine the data you want displayed. Attach event rules that provide business logic, and specify processing options that control the format, page breaks, report totaling, and how the report processes data. Examples of reports and batch processes are:

- Sales Update
- General Ledger Post
- Trial Balance by Cost Center
- Invoices
- Table Conversions
- Financial Reporting

Business Function Design

A business function is an encapsulated reusable routine that can be called through event rules. This code can be written in most industry standard third-generation languages. You use Business Function Design to write business functions to provide background processing to handle specialized tasks needed in an application, such as specialized editing on a field.

Event Rules

Event rules are logic statements. You create event rules and attach them to events. Events are activities that occur in an application, such as entering a form, exiting a field, exiting a row, or initiating a page break on a report. Event rules process when the user or the system initiates an event. Events are attached to controls, such as a particular field, the entire form, the grid, or a report section.

You use event rules to create complex business logic without the difficult syntax that comes with most programming languages. Examples of business logic that you can accomplish with event rules include:

- Perform a mathematical calculation
- Perform table I/O (fetch, insert, delete)
- Pass data from a source field on a form to a target field on another form
- Connect two forms or applications
- Hide or display a control using a system function
- Evaluate If/While and Else conditions
- Assign a value to an expression in a field
- Create variables (work fields) as you are working
- Perform a batch process upon completion of an interactive application
- Attach a business function or system function
- Initiate workflow processes

There are two types of Event Rules:

- Business function event rules
- Embedded event rules

Business Function Event Rules

Business function event rules are encapsulated, reusable business logic created through event rules rather than C programming. They are stored as objects and are compiled.

Embedded Event Rules

Embedded event rules are specific to a particular table, interactive application, or batch application. They are not re-useable. Examples include form-to-form calls, hiding a field based on a processing options value, and calling a business function. You can have embedded event rules in application event rules (interactive or batch) or in table event rules.

- Application Event Rules

You can add business logic that is specific to a particular application. Interactive applications connect Event Rules using the Form Design Aid, while batch event rules use Report Design.

- Table Event Rules

You can create OneWorld database triggers, or rules that are attached to a table and are executed through Table Design Event Rules. The logic attached to a table is executed whenever any application initiates that database action. For example, you might have rules on a master table to delete all children when a parent is deleted, which maintains referential

integrity. Any application that initiates a delete of that table does not need the parent/child logic imbedded in the application because it resides at the table level.

Processing Options

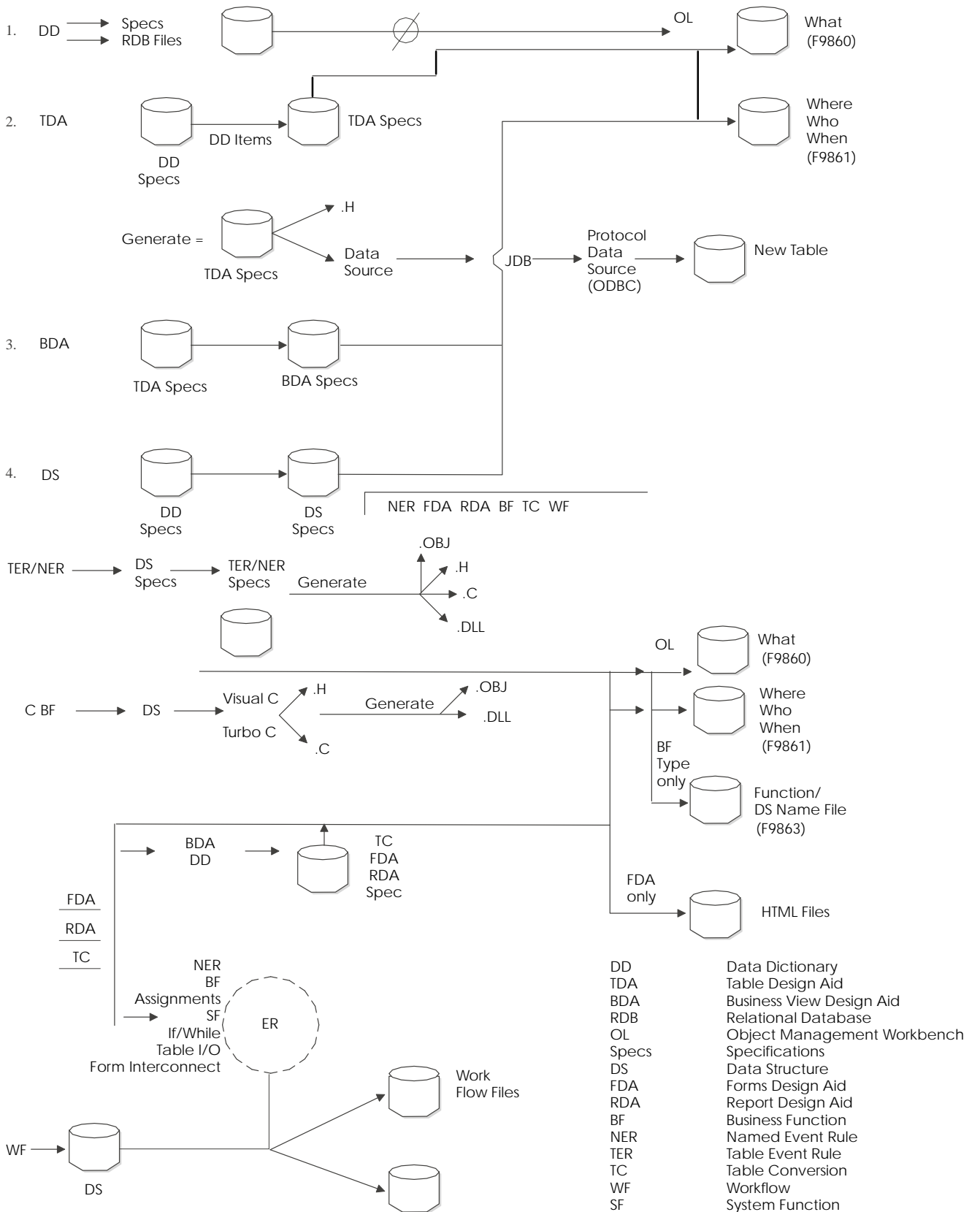
Processing options control how interactive and batch applications process data. You can have several versions of the same basic set of processing options. Each version might have one slight difference from another version. For example, you can use processing options to do the following:

- Determine the sequence of the forms in an application
- Set default values
- Turn on and off special processing, such as currency or kit processing in Sales Order Entry

Menus

Menus are the entry point to J.D. Edwards applications and reports. To access an application or report from a menu, it must be attached to a menu selection, and menu selections must be attached to a menu. See the *OneWorld Foundation* guide for information about creating customized menus.

How OneWorld Works at Design Time



Fundamentals



Object Management Workbench

The Object Management Workbench (OMW) is the change management system for OneWorld development. A change management system is vital to a productive development environment because it helps organize a myriad of development activities and helps prevent problems such as intermixing components from different releases and simultaneous changes to an object by multiple developers. The purpose of the OMW is to automate many of these change management activities.

The three OMW systems are:

Graphical User Interface (GUI) Unifies all development under an intuitive interface

Configuration System Controls all development from a central location

Logging System Automatically tracks all program changes

This section discusses the OMW GUI. The configuration and logging systems are discussed in *Object Management Workbench* in the *OneWorld System Administration* guide.

- Understanding the Object Management Workbench
- Working with the Object Management Workbench
- Working with projects
- Working with objects
- Working with tokens
- Working with users
- Working with attachments



Understanding the Object Management Workbench

To understand the Object Management Workbench (OMW), you should be familiar with the following items:

- Object Management Workbench concepts
- Object Management Workbench interface
- Object Librarian and non-Object Librarian objects

See Also

- Configuring Notification Subscriptions in the OneWorld System Administration* guide for information about using the OMW to notify you when specific events occur

Object Management Workbench Concepts

Object Management Workbench concepts include the following items:

- Projects
- Default project
- User roles
- Allowed actions
- Tokens

Projects

Projects are composed of objects and owners. All development of objects within OneWorld must be performed within the context of a project. Usually, you must first create or choose a project, add an object to it, and then you can work with that object. Typically, objects are included in a project because they have been modified or created by a developer to complete a specific task.

In addition to objects, users can be associated with different projects. In fact, before you can add an object to a project, you must have been added to the project as a user in a role that has permission to add objects. A user can be assigned to the same project more than once with different roles. Projects may also contain other projects.

Default Project

The default project is your personal project that can be used for development and research. It holds any miscellaneous development objects that you want to work with but that you have not associated with a specific project. OneWorld creates a default project when you run the OMW for the first time. OneWorld uses your OneWorld logon to name the default project.

Use your default project to:

- Research, develop, and prototype objects
- Review objects you do not need to modify or check in

The default project is similar to a project; however, the status of a default project does not change. Therefore, you cannot use a default project to transfer objects.

Some objects, such as versions, menus, workflow data, and reports can be created and edited outside of the OMW. Nevertheless, any changes made to these objects must be tracked and managed. The default project is used to manage these objects. If you create or access such objects outside of the OMW, these objects are added to your default project.

User Roles

Users must be assigned to a project before they can affect the project or the objects within that project. When you add a user to a project, you identify that user's role within the project as well. The user role defines the user's function within the project organization and might limit their access to certain OMW functions, depending on the allowed actions associated with the role. User roles and their allowed actions are defined in the Object Management Configuration application.

Note: Do not confuse user roles in the OMW with the concept of user roles as applied in other components of OneWorld software, such as the ActivEra Solution Explorer. OMW roles function independently of all other role-based systems in OneWorld.

Allowed Actions

Allowed actions are rules that define the actions that may be performed by a user assigned to a user role at a given project status. An administrator sets up these rules for each user role, object type, and project status using the OMW configuration application.

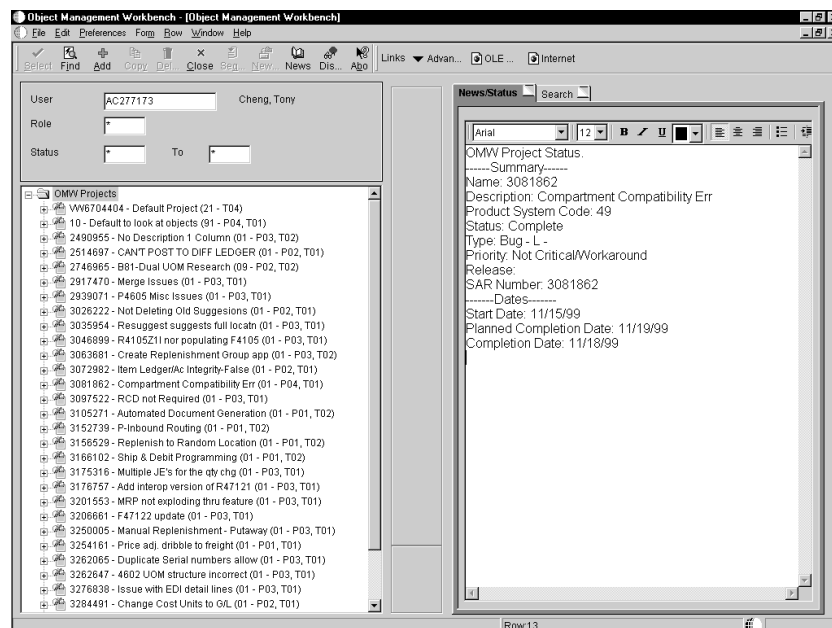
Tokens

Some objects use tokens to minimize the possibility of one user overwriting another user's changes to an object. Projects hold an object's token, and each

object has only one token. You can check out an object only if your project holds the object's token. In this way, an object can reside in several projects, but can only be checked out and in by qualified users of the project holding the token. If desired, however, you can allow other projects to share an object's token, allowing the object to be checked out and in by qualified users of one or more projects. However, an object can be checked out by only one person at a time.

Note: Only Object Librarian objects have tokens. See *Object Librarian and Non-Object Librarian Objects* for more information on Object Librarian objects.

Object Management Workbench Interface



From left to right, the initial OMW form displays:

- The project window to display your projects and their related objects and users. To view your current projects, click Find.

The color of an Object Librarian Object icon indicates its status as described below:

- Gray Object Icon with Checkmark: Another project holds the token for this object.
- Colored Object Icon (not gray): The object's project holds the token for this object.
- Colored Object Icon with Checkmark (not gray): The object's project holds the token for this object, and the object is checked out.

- **Gray Object Icon:** This object is not checked out and no project currently holds this object's token.

Non-Object Librarian Object icons do not vary in appearance.

Objects to be deleted are marked in bold in this window.

- The center column contains action buttons used to perform actions on a selected object. Available buttons vary based on your roles in the current project and on the status of the project in which the selected object resides. When you first launch the OMW, no buttons appear in the center column because no item has been selected.
- The information window, to display a web site, project status and release information, object or user information, and search results. Initially, the window displays a web site or HTML page. The contents change based on your tab and object selections. For example, when you select a project or an object in the project window, the information window displays information about the selected project or object. To return this window to its initial state, click News.

Object Librarian and Non-Object Librarian Objects

The OMW provides control of OneWorld objects in a simple, integrated, graphical user interface for OneWorld development. In OneWorld, an object is a reusable entity based on software specifications created by OneWorld Tools. OneWorld objects include Object Librarian objects such as interactive applications (APPL), batch applications (UBE), and data structure (DSTR) objects.

The OMW also allows control of some non-Object Librarian objects that are data source-based rather than path code-based. These objects include User Defined Codes (UDC), Workflow, Menus, and Data Dictionary Items.

OneWorld objects include the following Object Librarian and non-Object Librarian objects:

Object Librarian Objects are the following objects:

- Batch applications and versions (UBE)
- Business functions (BSFN)
- Business views (BSVW)
- Data structures (DSTR)
- Interactive applications (APPL)
- Media objects (GT)
- Tables (TBLE)

Non-Object Librarian Objects are the following objects:

- Data dictionary items
- User Defined Code items
- Workflow items
- Menus

Working with the Object Management Workbench

After your system administrator has configured the Object Management Workbench, including setting up security and OMW roles, you can start working with the OMW. This topic suggests the order of actions to take to accomplish certain tasks with the OMW.

- To use your default project
- Understanding project life cycle

To Use Your Default Project

Although your default project appears immediately, you only have one role as configured by your system administrator (usually Originator). You might need to add yourself to your default project in another role such as Developer. See *Adding Users to Projects* for instructions on adding yourself in additional roles to a project.

Understanding Project Life Cycle

This topic discusses a typical project life cycle from inception to completion. It includes steps required by a SAR-based system. If you are not using a SAR-based system, some of the following steps might not apply to you. Furthermore, depending on your business's software development procedures, the steps you follow and their order might vary from the process described below.

1. Based on the task to be accomplished, create a new project.

See *Creating New Projects* for instructions on creating a new project.

2. Add users to the project.

When you add a user, you will define that user's role based on the actions you want that user to be able to perform within this project. You might need to add a user more than once if you want the user to be able to perform actions allowed by different roles. As the project progresses, you can continue to add (or remove) users as required.

When you create a project with SAR integration turned off, you are automatically added to that project in the role determined by your system

administrator (usually, as the Originator). You might want to add yourself to the project in other roles as well.

When you create a project with SAR integration turned on, the person who entered the SAR is added to the project in the role of Originator.

See *Adding Users to Projects* for instructions on adding users to a project.

3. Add objects to the project.

Qualified users might be adding objects to the project throughout much of its life cycle.

If you create a new object, drag and drop the object from your default project to the project when appropriate to do so.

See *Adding Objects to Projects* for instructions on getting existing objects. Note that getting an object is not the same as checking out an object.

4. Check objects out and in.

To be able to save your changes to an object, you must check the object out, apply your changes, and check the object in. See *Checking Objects In and Out* for instructions on checking objects in and out.

When you attempt to check out an object, you will be successful if no other projects hold the token for that object. If the token is available, it is passed to your project when you check the object out. If another project already holds the token for the object, you can join a token queue to be notified when the token becomes available. See *Working with Tokens* for information about how tokens work.

After checking out an object and modifying it, you can save your changes without checking the object in. See *Changing Objects* for instructions on saving objects.

When you check an object in, the system might not release the token from the project, depending on how the OMW has been configured. As long as your project holds the token, another qualified user in your project could check the object out, but users in other projects could not. You can allow users in other projects to check out an object by removing the object from the project (see *Removing Objects from Projects*), by releasing the token (see *Releasing Tokens Manually*), by switching the token (see *Switching Tokens*), or by sharing the token with another project (see *Inheriting Tokens*).

5. Advance the project.

As the project progresses through its life cycle, you must change its status. You do this by advancing the project. See *Advancing Projects* for

instructions on advancing a project. When you advance a project, some roles' allowed actions might change and some objects might be transferred to other locations. Status-based role changes and transfers are configured by your system administrator.

6. Complete the project.

Based on your processes, you might archive or delete the project when finished. See *Deleting Projects* for instructions on deleting a project. The OMW considers 01 to be a closed status.

Working with Projects

In the Object Management Workbench (OMW), all development is performed within the context of a project. Working with projects is composed of the following topics:

- Viewing projects in the project window
- Creating new projects
- Changing project properties
- Advancing projects
- Adding existing projects to a project
- Deleting projects

Viewing Projects in the Project Window

By default, when you click Find on Object Management Workbench, the project window displays all of those projects to which you have been added in one or more roles. The project list can become lengthy in some cases, and you might want to filter the list so that only certain projects appear. For example, if you hold a developer role on some projects, you might want to filter your list so you view only those projects with a development status.

In addition to projects in which you have a role, you can also view any other projects in the system. You can search for projects based on a variety of criteria, including object.

Viewing projects in the project window contains the following tasks:

- Filtering projects
- Searching for projects

Filtering Projects

You can choose to filter the projects displayed in your project window by user, role, and status.

► **To filter projects**

1. On Object Management Workbench, complete the following fields in the project window:

- User

This field is required. When you launch the OMW, this field displays your ID. You can also enter other user IDs in this field.

- Role
- Status

The range you enter in these fields is inclusive. To search for projects with a specific status, enter the status code in both fields.

2. Click Find.

Field	Explanation
User	<p>The user or group for which the To status is valid. If the user is not setup for a specific status as a valid To status then the may not advance the project to that status. *PUBLIC is allowed.</p> <p>..... <i>Form-specific information</i></p> <p>This field allows you to put in the wild card character, '*', so that you can perform wild card searches similar to what you can do in a QBE. This field will not validate the user ID in this field, it always attempts to search for projects with anything in this field. You will know it is a valid user ID if the alpha description comes up in the associate description</p>
Role	<p>User Roles are set up for all the kinds of players that can participate in a project. The role essentially defines the user's function within the project organization. Project managers will generally assign a user to a project. When they do so, they will indicate what role that user will be playing. Examples of User Roles are:</p> <ul style="list-style-type: none"> 01 Originator: Personnel that originated the project. 02 Developer: Personnel that actually create the project. 03 Manager: Personnel that manage the project. 04 Quality Assurance: Personnel that check the project's functionality. 06 Administrator: Personnel that configure project status, user roles, objects, etc.

Field	Explanation
Status	<p>When a default project is added, this will be the status at which it is added. This can be one of the project status codes, which are:</p> <ul style="list-style-type: none"> 01 Complete 11 New Project Pending Review 21 Programming 25 Rework-Same Issue 26 QA Test/Review 28 QA Test/Review Complete 38 In Production 40 Production Development 41 Transfer Production to Prototype 42 Transfer Prototype to Development 45 Pristine Get 91 Canceled-Entered in Error <p>For any other project that is added, other than the default, this will be the status at which the project is added. This can be any one of the project status codes, which are listed above.</p> <p>This project status should be the starting point in the Status Activity Rules as defined in Object Management Configuration Application (P98230).</p>

Searching for Projects

You can search for projects you do not play a role on by object or other criteria. If you complete the filter fields in the project window before you perform a search, you can refine the search based on the information you enter in the filter fields.

Searching for projects contains the following tasks:

- Performing a project search
- Searching for projects by object

Note: Searches are case sensitive. When completing fields, be sure you enter the commonly accepted spelling in standard capitals and lower case for your search query. If you receive no search results, try different capitalization or spelling.

To perform a project search

1. On Object Management Workbench, select Advanced Search from the Form menu.

2. If you entered a user ID in the previous form, the OMW Project Search & Select by Project User form appears, and you can limit the search by completing the following fields:

- User ID
- Role
- Project Status

To search for projects with a specific status, enter the status code in both fields. The range you enter in these fields is inclusive.

The OMW Project Search & Select form appears if you did not complete any of the filter fields in the project window. These fields are unavailable on the OMW Project Search & Select form.

3. Enter the desired criteria in the Query By Example (QBE) columns and then click Find.
4. Choose one or more projects, and then click Select.

The projects you chose appear in the project window.

► To search for projects by object

This search method places all the selected projects directly in the project window.

1. On Object Management Workbench, select Search by Object from the Form menu.

UDC	OMW Object Name	Object Type	Description	Project	Description	SAR	Project Status
0001	UDC	Division	AE5915710	Default			0
0001	UDC	Division	B733OW8	Default			0
0001	UDC	Division	B733OW10	Default			0
0001	UDC	Division	B733OW13	Default			0
0001	UDC	Division	B733OW21	Default			0
0002	UDC	Region	B733OW8	Default			0
0003	UDC	Group	DD1411493	Default			0
0003	UDC	Group	B733OW8	Default			0
0004	UDC	Branch Office	DD1411493	Default			0
0004	UDC	Branch Office	B733OW8	Default			0
0004	UDC	Branch Office	ACC09	Default			0
0004	UDC	Branch Office	CB891392	Default			0
0005	UDC	Department Type	QA_USER	Default			0
0005	UDC	Department Type	B733OW8	Default			0
0005	UDC	Department Type	MT6085156	Default			0
0006	UDC	Person Responsible	P25764392	Default			0
0006	UDC	Person Responsible	B733OW8	Default			0
0007	UDC	Business Unit Reporting Code 7	QA_USER	Default			0
0007	UDC	Business Unit Reporting Code 7	P25764392	Default			0
0007	UDC	Business Unit Reporting Code 7	MM6808254	Default			0
0007	UDC	Business Unit Reporting Code 7	B733OW8	Default			0
0008	UDC	Business Unit Reporting Code 8	B733OW8	Default			0
0008	UDC	Business Unit Reporting Code 8	ACC09	Default			0
0009	UDC	BU Equipment Division Code	B733OW8	Default			0
0010	UDC	Business Unit Reporting Code10	JR174377	Default			0
0010	UDC	Business Unit Reporting Code10	MM6808254	Default			0
0010	UDC	Business Unit Reporting Code10	B733OW8	Default			0

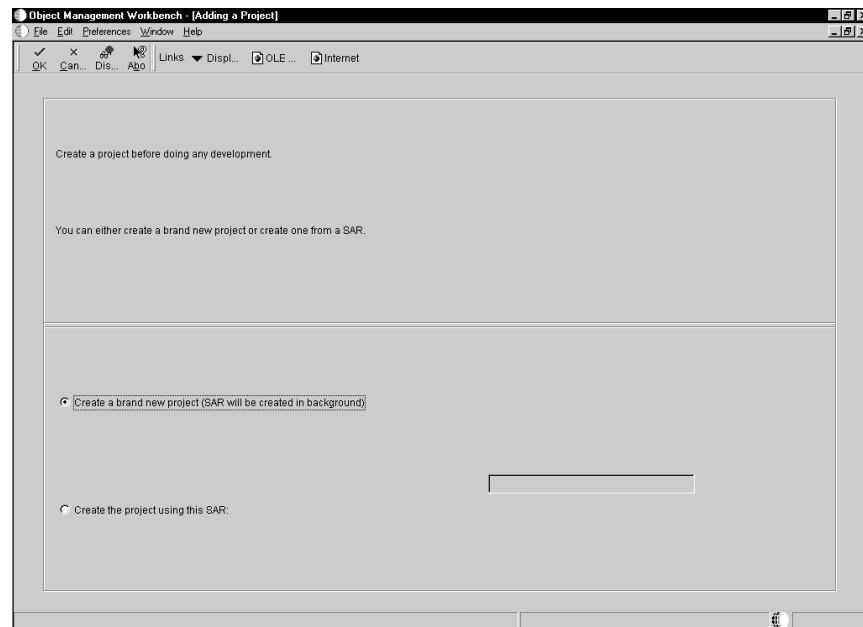
2. Enter the desired criteria in the Query By Example (QBE) columns, and then click Find.
3. Choose one or more projects, and then click Select.

Creating New Projects

Create a project to act as a container for objects and users grouped for a specific purpose. You might create projects for different system enhancements, for example. Through logging, projects also allow you to track the evolution of objects within the project, as well as the project itself.

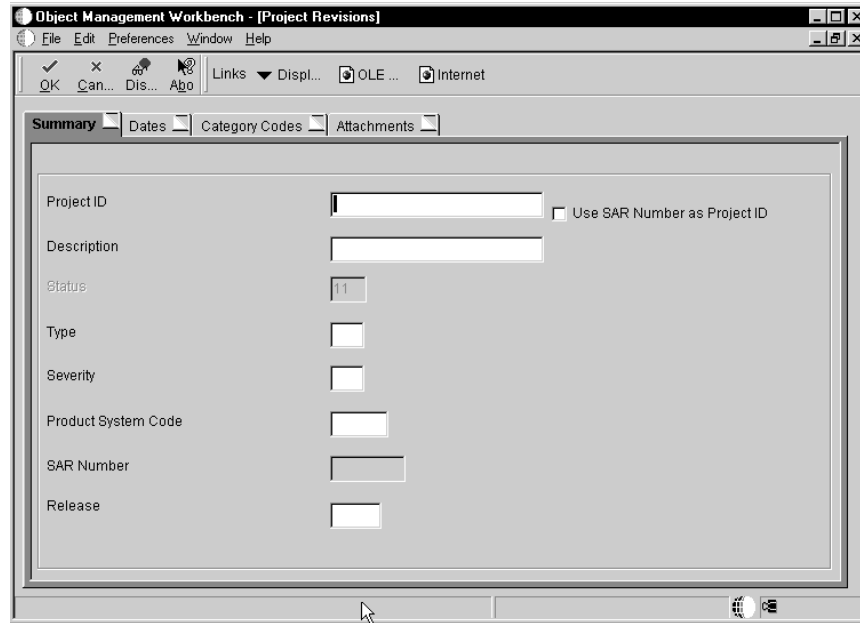
► To create new projects

1. On Object Management Workbench, click Add.
2. On Add OneWorld Object to the Project, click OMW Project, and then click OK.



3. Choose how you want to create the project, and then click OK.

The option to create a project using a SAR is valid when SAR integration is enabled. This form is unavailable if your system does not use the J. D. Edwards SAR system.



4. Click the Summary tab, and then complete the following fields:

- Project ID

J.D. Edwards recommends that you use the following format when naming your projects: *YYYzzzzz*

YYY = a company-specific code (JDE is reserved for J.D. Edwards projects)

zzzzz = a unique five-digit number

For example, ABC00001 might be the name of a project.

- Description
 - Type
 - Severity
 - Product System Code
 - Release
5. Click the Dates tab, and then complete the following field:
 - Planned Completion Date
 6. Click the Category Codes tab, and then complete the following optional fields:
 - Category Code 1 through Category Code 10
 7. Click the Attachments tab, and then add optional text comments to document the new project.

8. Click OK.

Field	Explanation
Project ID	A OneWorld project is made up of a group of OneWorld objects that have been modified or created by a developer to complete a task. All work objects within OneWorld must be done within the context of a project.
Product System Code	A code that designates the system number for reporting and jargon purposes. See UDC 98/SY.
SAR Number	A number that identifies an original document. This can be a voucher, an order number, an invoice, unapplied cash, a journal entry number, and so on.
Release	The transfer rule only applies to objects in the project that are for this release. (Objects are always tied to a release within a project). The From and To Release fields should always be equal.

Changing Project Properties

You can view and modify the following properties of any project you select:

- Description
- Type
- Severity
- Product system code
- Release information
- Start date
- Planned completion date
- Category codes
- Text attachments

To change project properties

1. On Object Management Workbench, click a project, and then click Select.

You can also click the Design button in the center column.
2. On Project Revisions, click the Summary tab, and then revise the following fields:

- Description
 - Type
 - Severity
 - Product System Code
 - Release
3. Click the Dates tab, and then revise the following fields:
 - Date Started
 - Planned Completion Date
 4. Click the Category Codes tab, and then revise the following optional fields:
 - Category Code 1 through Category Code 10
 5. Click the Attachments tab, and then add optional text comments to document the project.
 6. Click OK.

Advancing Projects

After development is complete for all objects in a project, the project's status must be advanced to send the project through the development cycle. Changing a project's status might affect the allowed actions of certain roles. The OMW can be configured to allow users to perform specific actions when a project is at a specific status based on their roles. For example, a user assigned to a project in a developer's role might be able to perform the following actions before the project is advanced: check out, design, and check in. However, once the project is advanced to the next status, a developer might not be able to perform any actions at all.

Changing the status of a project can also initiate actions, such as transferring objects in the project and deleting objects from the system that have been marked for removal. You cannot advance a default project.

Before You Begin

- Ensure all the project's objects are checked in. This includes objects in projects that are inheriting a token. In SAR-based systems, ensure you complete all required SAR fields.

To advance projects

1. On Object Management Workbench, click the project to be advanced.
2. Click the Advance Project button in the center column.

- Click the field labeled “>>>,” and then enter the desired project status.

Your choices are limited based upon the current status of the project and on your company’s specific procedures defined in the OMW Configuration application.

Note: Click the Validate Only checkbox to validate the status change without actually advancing the status of the project. This allows you to verify that the project is valid before attempting any object transfers. Any projects linked through token inheritance are validated at this time as well.

- Click OK.

If you did not click the Validate Only checkbox, the system advances the project status and initiates any required object transfers and deletions. Otherwise, only project status validation occurs.

Use the OMW logging system to view any errors that occurred during the status change. If you cannot advance the project, check the following conditions:

- All of the project’s objects must be checked in. This includes objects in projects that are inheriting a token.
- If you are using a SAR system, ensure you have completed all required fields in the SAR.

Field	Explanation
Validate Only	An option that specifies whether the system runs the change status validation routines without actually changing the project status or transfer any objects. Run without actually changing a project status to ensure a clean status change.

Adding Existing Projects to a Project

Besides objects and users, projects can contain other projects. You can add a project to a project, or, if the target project and the project to be added both appear in your project window, you can move it under the target project using drag-and-drop. The methods for adding and moving projects are identical to adding and moving objects.

See Also

- Adding Objects to Projects*
- Moving Objects*

Deleting Projects

When you delete a project, the system removes all objects and owners from the project. The project is then completely deleted from the system.

If you delete a project that contains objects that are checked out, the system erases the check-out on each object before deleting the project. If the project holds any tokens, the system releases them at this time as well.

▶ **To delete projects**

1. On Object Management Workbench, click a project, and then click Delete.

The system confirms the deletion.

2. Click OK in the Delete Confirm query.

Working with Objects

This section describes how to use the Object Management Workbench (OMW) to perform the following object-related functions:

- Creating objects
- Searching for objects
- Adding objects to projects
- Moving objects
- Removing objects from projects
- Deleting objects
- Getting object specifications
- Checking objects in and out
- Changing objects
- Maintaining objects in multiple software releases

Creating Objects

You can create a variety of objects with the OMW, including:

- Applications
- Business functions
- Data structures
- Tables
- Business views
- Data and menu items
- User defined codes (UDCs)
- Workflow processes

This topic describes how to create objects in general.

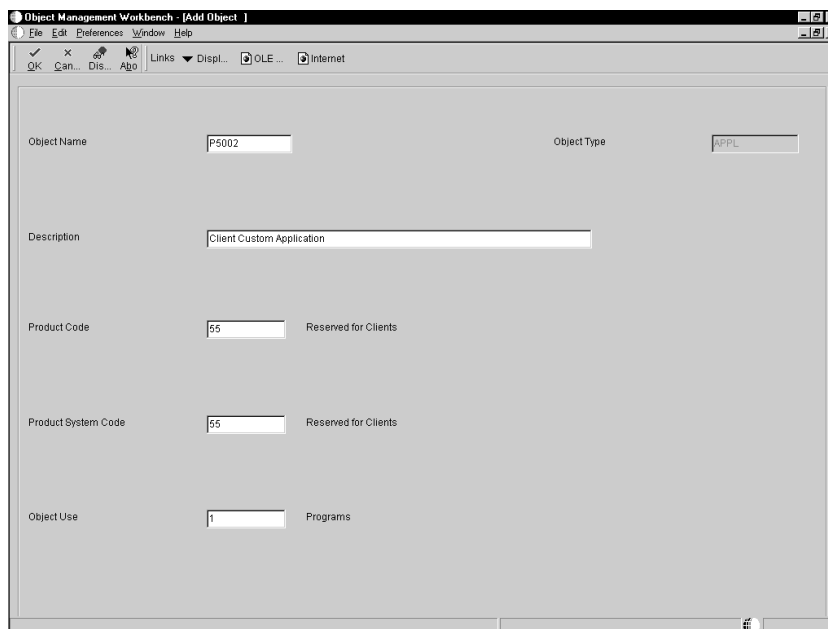
See Also

- ❑ *OneWorld Development Tools* guide for instructions on creating various Object Librarian and Control Table objects (most objects have their own sections)
- ❑ *OneWorld Enterprise Workflow Management* guide for instructions on creating workflow processes

▶ To create objects

1. From the Object Management Workbench, click Add.
2. On Add OneWorld Object to the Project, click the object type you want to create, and then click OK.

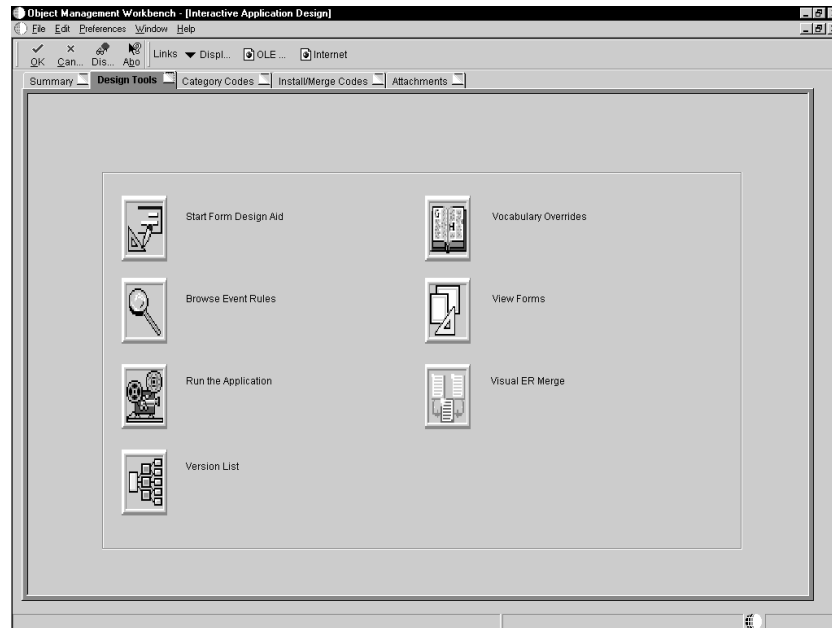
The Add Object form appears. The contents of this form vary based on the type of object you are creating.



The screenshot shows the 'Add Object' dialog box in the Object Management Workbench. The dialog has a menu bar (File, Edit, Preferences, Window, Help) and a toolbar (OK, Cancel, Dismiss, Abort, Links, Display, OLE, Internet). The form contains several fields: 'Object Name' with value 'P5002', 'Object Type' with value 'APPL', 'Description' with value 'Client Custom Application', 'Product Code' with value '55' and a note 'Reserved for Clients', 'Product System Code' with value '55' and a note 'Reserved for Clients', and 'Object Use' with value '1' and a note 'Programs'.

3. On Add Object, complete the fields as appropriate for the type of object you are creating, and then click OK.

Depending on the object you are creating, a design form might appear that provides the functions you need to design the object. For example, if you create an interactive application, the Interactive Application Design form appears. Click the Design Tools tab for access to buttons that launch the Form Design Aid, Work with Vocabulary Overrides, Work with Interactive Versions, and so forth.



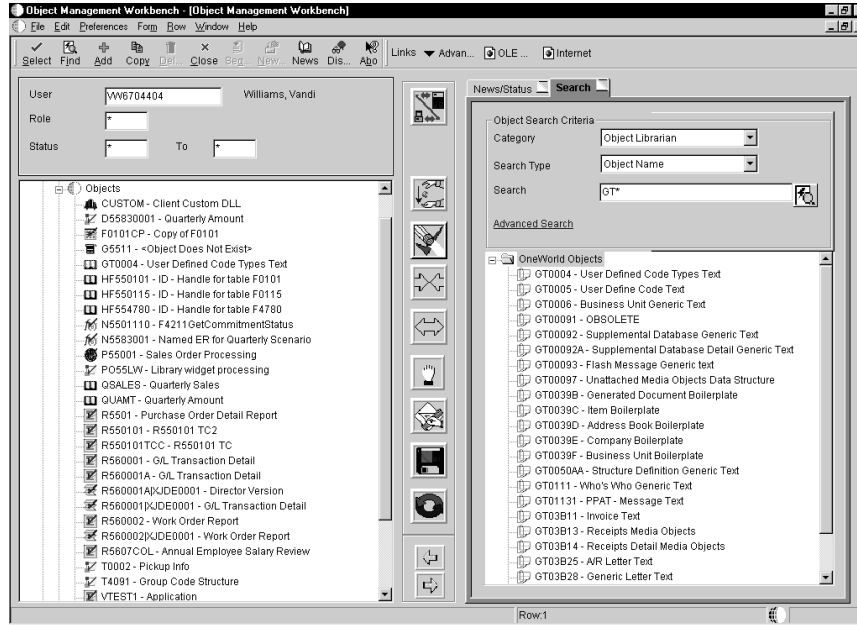
Searching for Objects

Conducting an efficient search is preliminary to adding objects to a project. You can search for objects by category and type, or you can perform an advanced search and find objects based on other criteria.

Note: Searches are case sensitive. When completing fields, be sure you enter the commonly accepted spelling in standard capitals and lower case for your search query. If you receive no search results, try different capitalization or spelling.

► To search for objects

1. On Object Management Workbench, click the Search tab.



2. Complete the following fields, and then click the button next to the Search field:

- Category

You can search a variety of categories. For example, to find a report, select Object Librarian as the category because reports are Object Librarian objects. To find a project, select OMW Project. To find a user, select Owners.

- Search Type

Valid choices for this field vary based on the category you select.

If you set the search type to Object Name | Version Name, you can use the | delimiter to specify a search suffix. For example, assume Category = Object Librarian and Search Type = Object Name. Entering R0008P | XJDE* displays all XJDE versions of object R0008P.

- Search

Entries in this optional field must match the Search Type selected.

3. To search for objects based on criteria other than category, search type, and name, click Advanced Search.
4. On Object Librarian Search and Select, enter the desired criteria in the Query By Example (QBE) columns, and then click Find.
5. Choose one or more objects, and then click Select.

The objects you selected appear in the information window. See *Adding Objects to Projects* for information about moving objects to the project window.

Adding Objects to Projects

An object must exist within one of your projects before you can work with it. You can add an existing object to a project, or you can create a new object for a project. When you create a new object, OneWorld places it in the selected project. If you did not select a project prior to creating the object, OneWorld places it in your default project. Adding an object to a project neither checks out the object nor downloads the object's specifications to your local environment.

Note: If you try to add an object to a project that already exists in that project, the Release Search & Select form appears because OneWorld allows you to modify the same object across multiple releases.

This task can be used to add users to a project. You can also use this task to add a project to another project.

Adding objects to projects contains the following tasks:

- Adding an object
- Adding multiple objects

See Also

- Moving Objects* for instructions on moving an object from one project in your project window to another project in your project window
- Checking Objects In and Out* for instructions on checking objects out so you can modify them
- Getting Object Specifications* for instructions on downloading the specifications of an object to your workstation without checking the object out

► To add an object

1. On Object Management Workbench, click the project where the object will be added.
2. Find the object to add to the destination project by performing a search using the Search tab in the information window.

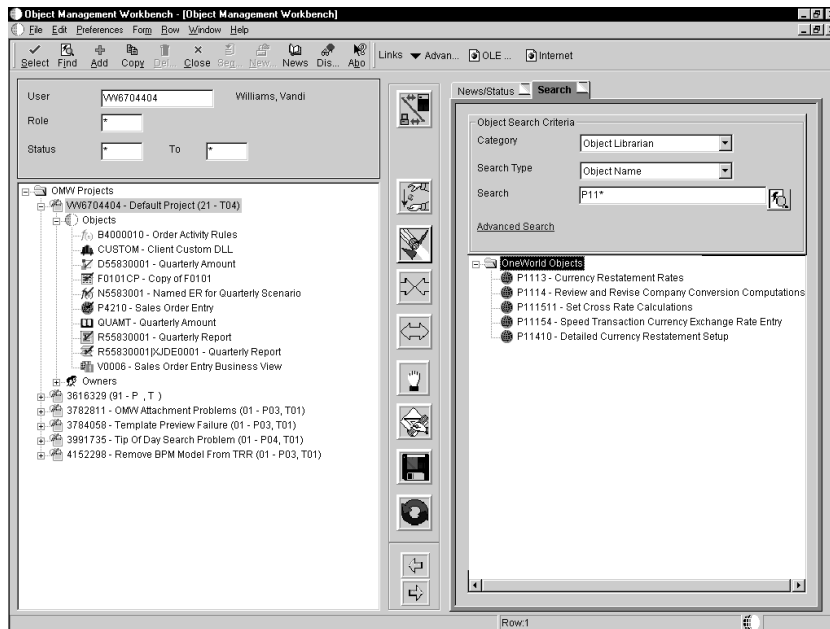
See *Searching for Objects* for more information on performing searches.

3. When the search completes, on the search form, select the object to be added to the destination project.
4. Verify that the destination project in the project window is highlighted. If it is not highlighted, click it.
5. With the object to be added highlighted, click the *Add Object or User to Project* button in the center column.

► **To add multiple objects**

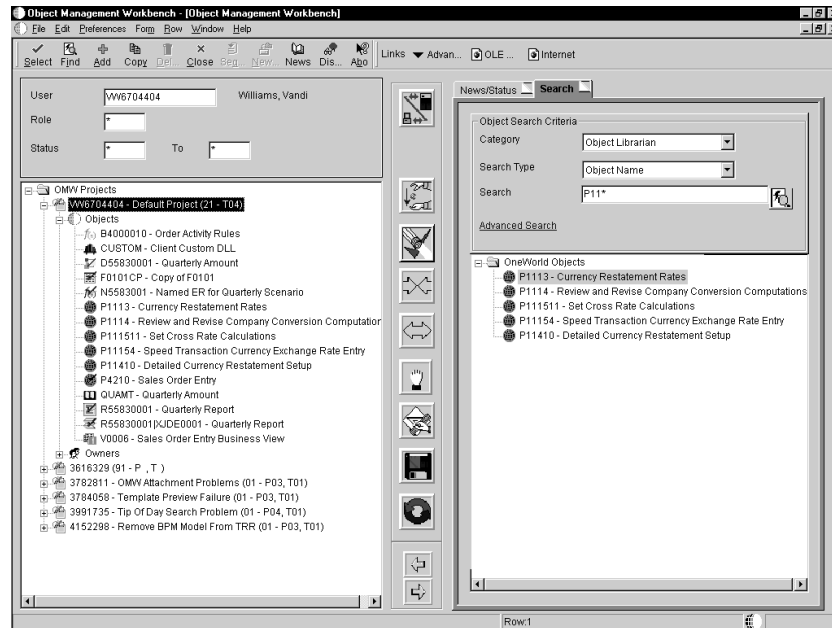
1. On Object Management Workbench, click the project where the objects will be added.
2. Find the objects to add to the destination project by performing a search using the Search tab in the information window.

See *Searching for Objects* for more information on performing searches.



3. Verify that the destination project in the project window is highlighted. If it is not highlighted, click it.
4. From the Row menu, choose Advanced, and then choose Add All Objects.

The system adds all of the objects that fit the search criteria to the project you selected in step 1.



Moving Objects

You can move objects from one project to another by dragging and dropping them. Both projects and the object must be visible in your project window. This task can be used to move users from one project to another. You can also use this task to move a project under another project.

Moving objects contains the following tasks:

- Moving an object
- Moving multiple objects

► To move an object

1. On Object Management Workbench, in the project window, click the object that you want to move.
2. Drag the object over the target project and drop the object.

The object is removed from the source project and added to the target project.

► To move multiple objects

1. On Object Management Workbench, in the project window, click the project containing the objects that you want to move.

2. From the Row menu, choose Advanced, and then choose Move Objects.
3. In the To Project field, enter the project to which you want to move the selected objects.
4. On Move Multiple Objects Search & Select, click the objects you want to move.
5. Click Select.

The system moves the objects from the source project to the target project. This process might take a while, depending on the number of objects selected.

Removing Objects from Projects

This action simply removes the reference to the object from the project; it does not actually delete the object. This task can be used to remove users from a project as well.

To remove objects from projects

1. On Object Management Workbench, select an object in the project window.
2. Click the *Remove Object or User from Project* button in the center column.

Deleting Objects

You can delete any object from the server defined for the current status. You can also mark an object for deletion from its transfer locations upon project advancement or from its current save location (the location where the system saves the object when you click the Save button in the center column of the OMW).

Use this task to remove an Object Librarian object's specifications from your workstation as well.

When you select Delete Object from Server for a non-Object Librarian object, the system deletes the object from locations defined in the transfer activity rules when you click OK. If you select Mark Object to be Deleted from Transfer Locations, the system deletes the object from any other configured locations when the project advances.

For an Object Librarian object, you can delete the local and save specifications and, if the Object Librarian object is checked in, the checked-in version of this object by selecting Delete Object from Server. If you select Mark Object to be Deleted from Transfer Locations, the Object Librarian object is deleted from its

administratively defined transfer locations that are defined in the transfer activity rules when the Project Status is advanced.

► To delete objects

1. On Object Management Workbench, select an object in the project window.
2. Click Delete.

The Delete Object Confirmation form appears. Your available options vary based on object type and if the object has been checked in.

3. Select one or more of the following options, and then click OK:

- Delete Object from Server

Click View Locations for a list of locations from which the object is deleted when you select this option. This action occurs as soon as you click OK.

- Delete Object Locally

This action occurs as soon as you click OK.

- Delete Object from the SAVE location

This action occurs as soon as you click OK.

- Mark Object To Be Deleted From Transfer Locations

Objects marked to be deleted from transfer locations appear in bold letters in the project window. They are deleted from the transfer locations when the project status is advanced.

- Remove Object from ALL locations

This option selects all of the above options.

Getting Object Specifications

To download checked-in object specifications from the server defined for the current status, select the object and click the Get button in the center column. This function is useful if someone else has been working on the object and you want to see the changes or if you have made changes to the object but want to abandon them in favor of another version of the object.

The Get button allows you to get the specifications for objects residing in your path code only. However, you can download the specifications of an object

residing in other areas of the system as well. For example, you might want to get the specifications for an object as it existed in a previous software release. Use the following process to specify the location of the object that you want to download.

Note: If you want to review the object and not save any changes, use the Get button to copy the latest specifications to your local workstation instead of checking out the object and then erasing the checkout.

► Using advanced get

1. On Object Management Workbench, select an object in the project window.
2. From the Row menu, select Advanced, and then select Get.

The system verifies that you want to overwrite your local specifications.

3. On Path Code Search & Select, click Find.
4. Choose the location of the object you want to get, and then click Select.

Checking Objects In and Out

You can check out Object Librarian objects residing in your projects, provided the object's token is either available or held by the project in which the object currently resides. Only one user at a time can check out an object. Check out fails if the object is already checked out or if the token is unavailable. If the token is unavailable, you can opt to join the token queue for the desired object. If you join the token queue, you will be notified when the token is available and your project will receive the token.

Check in an object when you want to upload its specifications to the server and make it publicly available. When you check an object in, the system records the project in which the object resides as well to ensure that only changes made under the current project are transferred when the project is advanced to a status that triggers a transfer. If you move an object from one project to another using the drag-and-drop method, the system tracks the change and records the object's new project. However, consider the following scenario:

- You add an object to a project and check it out.
- You change the object and check it in.
- You use the right-facing arrow in the center column to remove the object from the project.
- You later add the object to a different project.

In this scenario, the system cannot track the object because it passes out of a project completely. Therefore, when you advance the second project, if the system needs to transfer the object as part of the advance, the transfer will fail because the object's last known check-in project location and its current project location do not match. When you drag-and-drop an object, the system updates its tables in such a way that the transfer can occur. This is not the case when you remove an object from a project and then add it to a different project later.

If an object is checked out, you can erase the check out. When you erase a checkout, local changes are not uploaded to the server. Erasing an object's checkout does not release its token, but it does allow other developers assigned to the same project to check out the object.

Checking objects in and out is composed of the following tasks:

- Checking objects out
- Checking objects in
- Erasing checkouts

See Also

Working with Tokens

► **To check objects out**

1. On Object Management Workbench, select an object in the project window.
2. Click the Checkout button in the center column.

The OMW indicates an object is checked out by superimposing a checkmark over the object's icon. Additionally, data about the object that is displayed in the information window is updated to reflect its checked out status.

Note: If the object is unavailable, the system asks if you want to be added to the object's token queue. If you opt to join the queue, you will be alerted when the token is released, and your project will be assigned the token. To determine which project holds an object's token, select the object in the project window and click the News/Status tab in the information window. Additionally, if you have joined a token queue, your position in the queue will be displayed here.

► **To check objects in**

1. On Object Management Workbench, select a checked-out object in the project window.

2. Click the Check-in button in the center column.

The OMW indicates an object is checked in by removing the checkmark superimposed over the object's icon when it was checked out.

To erase checkouts

1. On Object Management Workbench, select a checked-out object in the project window.
2. Click the Erase Checkout button in the center column.

The OMW indicates an object is no longer checked out by removing the checkmark superimposed over the object's icon when it was checked out.

Changing Objects

When you create an object using the Object Management Workbench, the OMW allows you to define the object's properties. The OMW also provides access to design tools and system actions for the object. Similarly, after the object has been created, you can use the OMW to modify the object and its specifications.

Your system administrator can also specify a separate save location that is different from your local environment and from the object's location on the server. Save objects to this location by selecting the object and clicking the Save button in the center column. Retrieve an object from its save location by selecting the object and clicking the Restore button in the center column. Note that an object's save location and its system location must be different.

You must check out the object before you modify it to be able to check the object back in and upload the changes.

As users modify objects, the changes exist only in their local environments until they either save the object to its save location or check in the object to its system location.

To change objects

1. On Object Management Workbench, choose an object in the project window.
2. Click the Design button in the center column.

An appropriate design form for the object appears. The object's current properties appear on the form.

3. When finished, click OK.

Maintaining Objects in Multiple Software Releases

Same-named objects in different software releases can be modified in the OMW in the same project. After adding the objects to the project, you can maintain them independently or you can update one to match the other. When working on objects from separate releases, the OMW handles save and checkin file paths for you based on the Object Management Configuration. Simply perform the necessary modifications and use the OMW functions as you would normally.

Caution: Changing and maintaining objects in multiple releases can cause problems due to OneWorld object intra-dependencies. Changing an object in one version and then updating the object in another version to match might cause dependent objects to malfunction.

Complete the following tasks:

- Adding same-named objects to a project
- Changing the release level of an object in your project
- Updating an object to match another object
- Updating different objects in different releases

Before You Begin

- You will need to know the paths of the objects that you modify initially.

To add same-named objects to a project

1. On Object Management Workbench, add the first object to the project.

Note: The object is added to the project at the current release level of your OneWorld software.

2. Add the same object to the project again.
3. On the Release Search and Select form, click Find.

All available releases for which the object can be added to the project appear.

4. Click the release you want, and then click Select.

The object is added to the project for the selected release level.

► **To change the release level of an object on your project**

1. On Object Management Workbench, choose Advanced from the Row menu, and then choose Change Release.
2. On the Release Search and Select form, click Find.

All available releases for which the object can be added to the project appear.

3. Click the release you want, and then click Select.

The object is added to the project for the selected release level.

► **To update an object to match another object**

1. Check out the object A from release A.
2. Modify the object.
3. Check in the modified object A.
4. Check out the object B from release B.
5. Select object B, and from the Row menu choose Advanced, and then choose Get.
6. On Path Code Search & Select, find and select the path code where the release A version of the object was checked in to, and then click Select.

In your project, the release B version of the object is modified to match the release A version of the object.

7. Check in object B.

► **To update different objects in different releases**

1. Check out the object from release A.
2. Modify the object.
3. Check in the modified object.
4. Check out the object from release B.
5. Modify the object.
6. Check in the modified object.

Working with Tokens

In the Object Management Workbench (OMW), Object Librarian objects use tokens to minimize the possibility of one user overwriting another user's changes. Each object has a single token, and it is associated with a project when the object is checked out. Checking in the object does not release the token, however. Instead, the token is released when the project's status changes to a level determined by your system administrator. At that time, another developer can check out the object and receive the token.

The following three actions are allowed while your project holds the token:

- Allow another project to inherit the token. This forces both projects to be advanced together as if they were one project and allows multiple project fixes to be applied to a single object. No matter how many projects have inherited the token, however, only one user at a time can check out the object. For a project to successfully inherit a token, the target project must be at the same status as the source project.
- Switch the token to another project. After the token is switched, the project losing the token will be placed in the token queue as the first project waiting for the token. To maintain object security, token switching should be restricted to a specific user role when configuring the OMW.
- Release the token. An owner on the project can simply give up the token and allow the next project in the queue to receive it.

Keep in mind that the Object Management Workbench may have been configured to release tokens for different object types at different project status levels. Therefore, all object types may not give up their tokens during the same change in Project Status.

Working with tokens is composed of the following topics:

- Understanding the token queue
- Inheriting tokens
- Switching tokens
- Releasing tokens manually

Understanding the Token Queue

The OMW attempts to acquire a token for an object when you check out an object. If the token is unavailable, the information window displays information about the token such as what project currently holds it, the user who checked it out, and when it was checked out. Additionally, you can opt to join the token queue so you are notified when the token is released and your project is assigned the token. Projects in the token queue are assigned the token in the order in which the token was requested. Additionally, after joining the token queue, you can choose to inherit the token.

Once a project has a token, the token stays with that project until the project advances to a status configured in the activity rules for release of the token or until it is switched or released manually. When the token is released, the next project in the token queue is notified and assigned the token. There is one token per Object Librarian object per release.

If you joined a token queue and then decide later that you do not need the token, remove the object from your project to relinquish your position in the queue.

Use the following process to view the token queue for an object.

To view a token queue

1. On Object Management Workbench, click an object in the project window.
2. From the Row menu, choose Advanced, and then choose Token Queue.

The View Object's Token Queue form appears. The form shows which project currently holds the token and which projects, in order, are in the queue.

See Also

 *Inheriting Tokens*

Inheriting Tokens

Token inheritance can be useful when developers have the same object in multiple projects for which they would like to implement fixes simultaneously, without having to wait for other projects holding the token to progress through the project life cycle.

To inherit tokens, both the project holding the token and the inheriting project must be at the same project status. After a token is inherited, these projects will be linked and will automatically advance in project status together. Therefore, if

the Project Status of one project is advanced, the Project Status of its linked project also advances. If one or more projects are linked through token inheritance, you should ensure all development in the linked projects is complete before advancing the projects. The user attempting to advance the project must be assigned a role that permits this action in all of the linked projects or the advance attempt will fail.

All project advancement requirements must be met for all projects linked through token inheritance; if one project fails to advance, OMW will not advance any of the other linked projects. If an advancement failure occurs, check the logs for all the linked projects to determine where the errors occurred.

► To inherit tokens

1. Attempt to check out an object whose token is held by another project.

The system asks you whether you wish to enter the object's token queue or inherit the token.

2. Choose to inherit the token, and then click OK.

Note: If you have inherited the token but cannot check out the object, the object is already checked out by another user. You cannot check out the object until the other user checks it in or until checkout is erased. This prevents overwriting changes when the token is inherited.

Switching Tokens

A project owner whose role allows switching tokens may take the token from the project that currently holds it and assign it to another project. An example of how this capability can be useful is for the implementation of an emergency fix. If a fix in another project needs to be implemented to an object in your project, you can switch the token to the other project to allow the fix.

Note: After the token has been returned, the user from whom the token was taken can save the object, check the object out, and then restore the object to return the object to its state before switching. However, the user will have to implement any changes made during the switch manually.

To switch a token, you must be an owner in both the holding and the requesting projects. Your role in both projects must be one that allows the action, switch token, at the project's current status and for the object type in question.

Before You Begin

- The token requester should attempt to check out the object and then should opt to join the token queue.

▶ **To switch a token**

1. On Object Management Workbench, select the object with the token you want to switch.
2. Click the Switch Token button on the central column.
3. On Project Token Queue Search and Select, click Find.

A list of projects in the token queue appears.

4. Choose the project you want to give the token to, and then click Select.

The current token owner should save the object before switching the token.

Releasing Tokens Manually

You can release a token manually if you decide you do not need to modify an object. Additionally, you can release the token if you would like to allow the next person in the token queue to check the object out for development. If you have made changes to an object and checked it in, another developer in another project must refrain from checking the object in until after your project has been promoted to a status where the system transfers the object to the next path code, or your changes will not be transferred.

See Also

- Understanding the Token Queue*
- Advancing Projects*

▶ **To release tokens manually**

1. On Object Management Workbench, either erase the check-out or check in the object with the token you want to release, if appropriate.
2. Select the object, then click the Release Token button in the center column.

Working with Users

This section describes how to use the Object Management Workbench (OMW) to perform the following user-related functions:

- Searching for users
- Adding users to projects
- Removing users from projects
- Changing user properties

Searching for Users

Conducting an efficient search is preliminary to adding users to a project. You can search for users name or ID or you can perform an advanced search and find users based on their class or group.

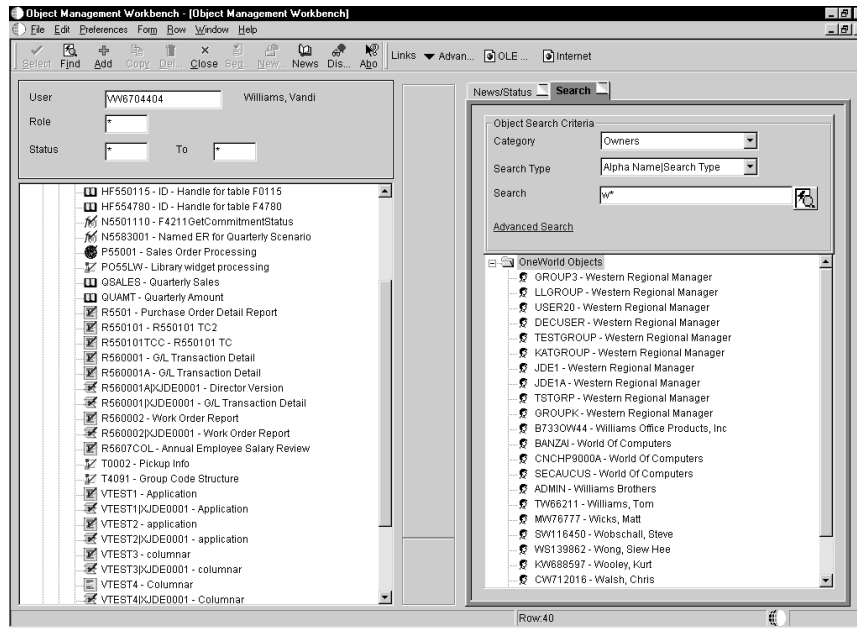
Searching for users describes the following search types:

- Searching for users by name or ID
- Searching for users by class or group

Note: Searches are case sensitive. When completing fields, be sure you enter the commonly accepted spelling in standard capitals and lower case for your search query. If you receive no search results, try different capitalization or spelling.

To search for users by name or ID

1. In Object Management Workbench, click the Search tab.



2. Complete the following fields:

- Category

Enter Owner.

- Search Type
- Search

Entries in this optional field must match the Search Type selected.

You can use | to specify a search suffix. Example: Assume Category = Owners and Search Type = Address Book#|Search Type. Entering *|E displays all entries in the Address Book with a search type of E for employee.

3. Click the Search button next to the Search field.

► **To search for users by class or group**

1. On Object Management Workbench, click the Search tab.

2. Complete the following fields:

- Category

Enter Owner.

- Search Type

3. Click Advanced Search.

4. On OneWorld User ID Search and Select, complete one or more of the QBE columns and click Find.
5. Choose the users you want, and then click Select.

Adding Users to Projects

To affect a project and the objects within that project, a user must be added to the project. When added to the project, a user is assigned a specific role. This role dictates the kind of actions they can perform. A user may be added to a project more than once with different roles. Additionally, some roles may be associated with several users. For instance, a project might include several developers.

► To add users to projects

1. On Object Management Workbench, click the project where the user will be added.
2. Set up a list of users to add to the destination project by performing a search using the Search tab in the information window.

See *Searching for Users* for more information on performing searches.

3. When the search completes, on the search form, select the user to be added to the destination project.
4. Verify that the owners node in the destination project in the project window is highlighted. If it is not highlighted, click it.
5. With the user to be added highlighted, click the *Add Object or User to Project* button in the center column.
6. On Add User to Project, complete the following fields, and then click OK:
 - Role
 - Lead

Note: To add a user in more than one user role, repeat the add user procedure and select a second user role for the same user. Different functions are enabled for different user roles according to their allowed (user) actions. These actions are configured by the administrator for your project using the configuration program of the OMW.

Field	Explanation
Role	User Roles are set up for all the kinds of players that can participate in a project. The role essentially defines the user's function within the project organization. Project managers will generally assign a user to a project. When they do so, they will indicate what role that user will be playing. Examples of User Roles are: 01 Originator: Personnel that originated the project. 02 Developer: Personnel that actually create the project. 03 Manager: Personnel that manage the project. 04 Quality Assurance: Personnel that check the project's functionality. 06 Administrator: Personnel that configure project status, user roles, objects, etc.

Removing Users from Projects

Removing a user from a project does not delete the user from the system.

► To remove users from projects

1. On Object Management Workbench, select a user in the project window.
2. Click the *Remove Object or User from Project* button in the center column.

Changing User Properties

You can modify the following user properties:

- User Role
- Estimated Hours
- Lead

► To change user properties

1. On Object Management Workbench, select a user (owner) in the project window, and then click Select.
2. On Project User Details, complete the following fields, and then click OK:
 - User Role
 - Project Lead
 - Estimated Hours

Working with Attachments

The Object Management Workbench (OMW) allows you to add text, graphic, OLE, and file attachments to projects and to Object Librarian objects within projects. These attachments are available only through the OMW; they neither affect the way the object functions nor are they available when a user employs the object. This feature is frequently used to document the creation, purpose, and intended use of objects in the system.

You can work with attachments from either of two places:

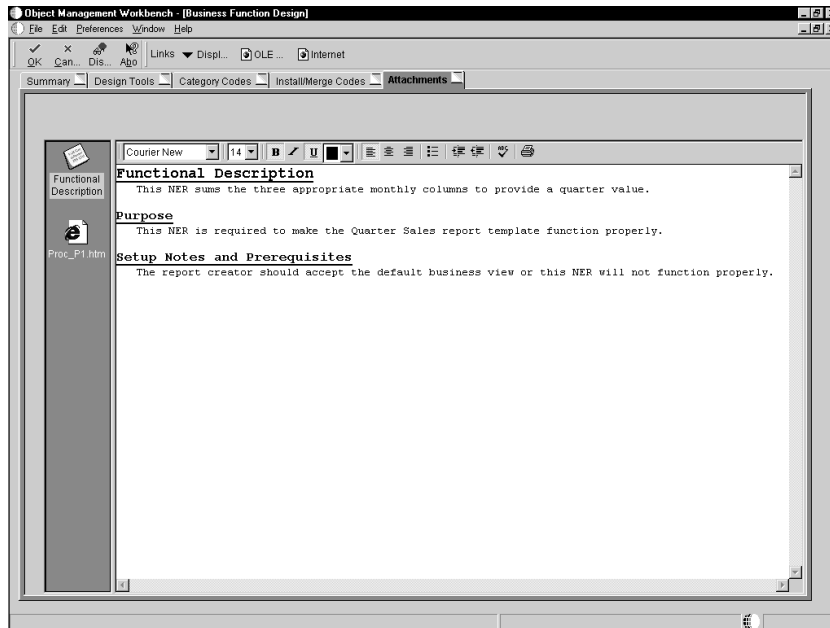
- Viewing attachments in Design view
- Viewing attachments in the Object Management Workbench

See Also

- Media Object Attachments* in the *OneWorld Foundation* guide for information on adding and working with attachments
- Media Objects and Imaging* in the *OneWorld System Administration* guide for information on enabling media objects
- Creating a Media Object Data Structure and Processing Media Objects*

▶ **To view attachments in Design view**

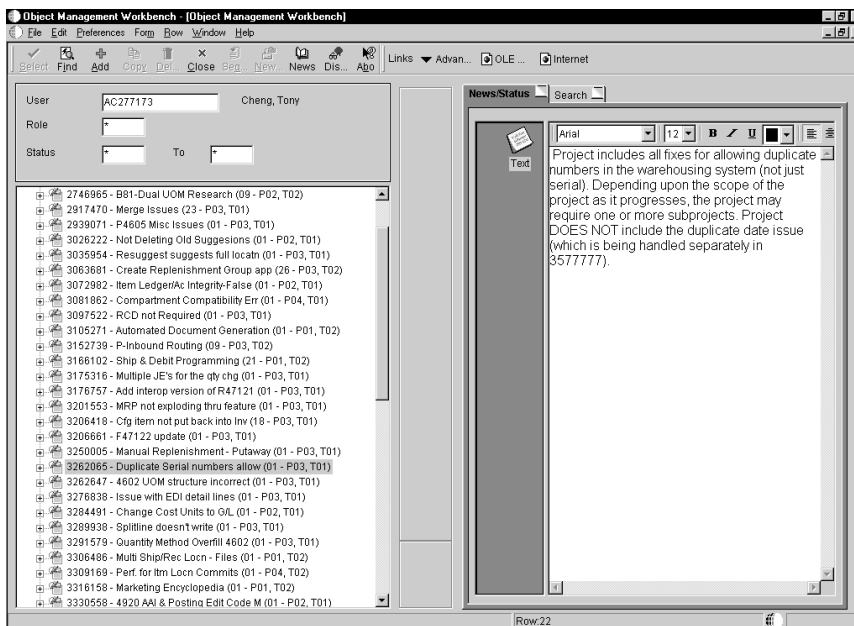
1. In Object Management Workbench, create an object or project, or choose an existing object or project and click the Design button in the center column.
2. On the resulting Design form, click the Attachments tab.



► **To view attachments in the Object Management Workbench**

1. On Object Management Workbench, choose a project.
2. Click the News/Status tab.
3. From the Row menu, choose Attachments.

If attachments exist, they appear in the information window.





Data Dictionary

Just as a dictionary contains word definitions, the J.D. Edwards data dictionary is a central repository that contains data item definitions and attributes. A data item identifies a unit of information. The data item definition defines how the item can be used and includes information such as the type of item and its length. The data item attributes determine how a data item:

- Appears on reports and forms (such as number of decimals, and default values)
- Validates data entry within an application
- Assigns column and row descriptions
- Provides text for field-sensitive help
- Is stored in a table

The data dictionary is dynamic. That is, any changes made to a data item are effective immediately across all applications. Applications access the data dictionary at runtime and immediately reflect modifications to data item attributes, such as field descriptions, column headings, decimals, and edit rules.

Use the data dictionary to create, view, and update attributes for data items. You can copy a data item that has similar attributes and modify it for your specific needs, rather than creating a new one from scratch. This may be quicker and easier than creating a new one. If you do this, you must modify the alias to distinguish between the copy and the original.

When you change a data item the changes are immediately reflected throughout the OneWorld tools at runtime. Changing the type and any of the attributes may affect how your data is stored and cause discrepancies between records.

Caution: The data dictionary does not verify whether a data item is used by an application when you delete it. If you delete a data item that an application uses, the application will fail.

Data dictionary is composed of the following topics:

- Understanding the data dictionary
- Using the data dictionary
- Defining a data item



Before You Begin

Before you begin working with the data dictionary, you should be familiar with the concepts in:

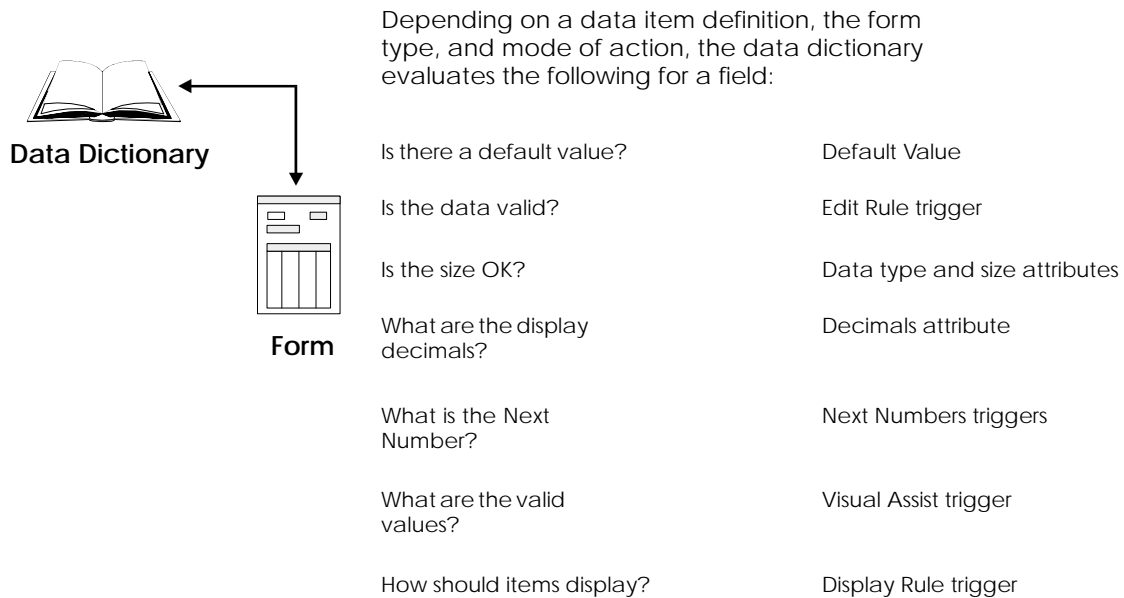
- OneWorld Foundation Guide*
- Object Management Workbench*

Understanding the Data Dictionary

Understanding the data dictionary is composed of the following topics:

- How the data dictionary is used at runtime
- Naming data items
- Storing the data dictionary and data dictionary items
- Glossary items
- Error messages
- Understanding default triggers

How the Data Dictionary is Used at Runtime



At runtime, applications access the following data dictionary fields:

- Display Decimals
- File Decimals
- Alpha Description

- Data Type
- Size
- Glossary
- Allow Blank Entry *
- Upper Case Only *
- All Triggers *
- Row and Column Headings *

The application retrieves field information from the data dictionary. Fields marked by an asterisk (*) can be overridden in Forms Design and Report Design. In these instances, the application retrieves the overrides, if any exist.

Naming Data Items

The following illustrates the naming components for a data item:

Data Item Name		Alias		Alpha Description
Company	=	CO	=	Company
CostCenter	=	MCU	=	Business Unit

After a data item is created, you can change only the alpha description; you cannot change the data item and alias. When you add a data item, the data dictionary edits the data item and alias fields to ensure that they are unique.

See *Development Standards: Application Design Guide* for naming standards.

Storing the Data Dictionary and Data Dictionary Items

The data dictionary is stored in two places:

- The central object data dictionary is stored in a relational database. All changes to the data dictionary that will be replicated a must be done here.
- The replicated data dictionary allows each workstation to have a set of data dictionary tables stored in specification tables on the client machine.

Data dictionary items reside on enterprise (logic) servers in relational database tables. Workstations retrieve from the publisher data dictionary (the relational database tables) only those data items necessary for the applications that you are using. This replication occurs when you use an application for the first time after installing OneWorld. This data dictionary information is stored on your workstation in a permanent cache under the same local path code/spec directory as the following global tables:

- glbltbl.xdb (references for the data)
- glbltbl.ddb (the data items)

If data items are changed and you want the changes replicated to workstations immediately, you must use the Data Replication (P98DREP) application. When you have data replication set up and an item is changed, the next time a machine that is set up as a subscriber signs onto OneWorld, the permanent cache for that data item will be deleted. Then the next time they use an application that has that changed data item, OneWorld will detect that the information is not in the permanent cache and retrieve the information from the publisher data dictionary (the relational database tables).

If you are using OneWorld and World coexistence, you must maintain two data dictionaries. You cannot share one dictionary because in World the \$ is reserved for business partners and is used for the beginning of an alias. The \$ does not compile and thus cannot be shared with OneWorld. Some of the file formats in World and OneWorld are also different.

See Also

- *Data Dictionary Administration* in the *OneWorld System Administration* guide

Glossary Items

Glossary items are items that cannot be attributes in tables. Glossary items are typically used as information messages.

Error Messages

Error messages used in OneWorld are stored as data items. Use the data dictionary glossary item application to display error messages because you do not need to use all the fields required for regular data items.

Understanding Default Triggers

A default trigger is an editing or display routine that is attached at the dictionary level and initiated at runtime. Default triggers are reusable objects and, therefore, automatically associated with each application that uses the data item. Default triggers leverage your time and code because you only have to create the business logic once, but you can use it within multiple applications. Default triggers ensure accuracy and integrity of data across all applications.

You use triggers to:

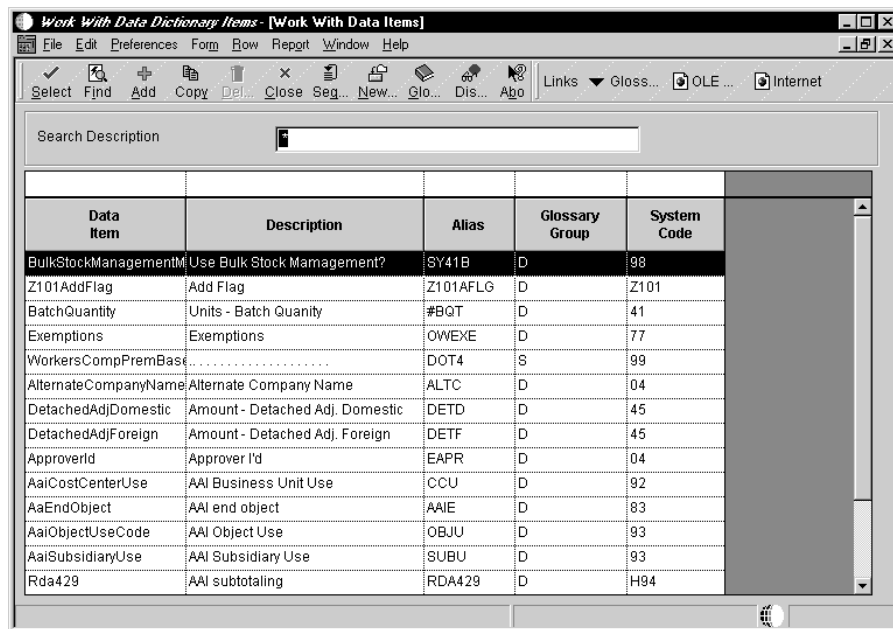
- Establish field default values
- Link data items to a user defined code (UDC) table of valid values
- Activate a visual assist search program when a user tabs into a field. The types of visual assists are:
 - calendar
 - calculator
 - Search and Select of another file
 - UDC
- Establish rules and procedures that are embedded in the editing and formatting of a field's data
- Determine a next number scheme that should be used when assigning a number to data

Using the Data Dictionary

You can create new data dictionary items and view existing ones with the Object Management Workbench or with Work With Data Items (P92001). Once created, use Work With Data Items to define jargon and language translations for data items.

► To use the Data Dictionary

From the Data Dictionary menu (GH951), choose Work With Data Dictionary Items (P92001).



The screenshot shows a software window titled "Work With Data Dictionary Items - [Work With Data Items]". The window has a menu bar with "File", "Edit", "Preferences", "Form", "Row", "Report", "Window", and "Help". Below the menu bar is a toolbar with icons for "Select", "Find", "Add", "Copy", "Del...", "Close", "Seg...", "New...", "Glo...", "Dis...", and "Abo". To the right of the toolbar are buttons for "Links", "Gloss...", "OLE...", and "Internet". Below the toolbar is a search box labeled "Search Description". The main area of the window contains a table with the following data:

Data Item	Description	Alias	Glossary Group	System Code
BulkStockManagement	Use Bulk Stock Management?	SY41B	D	98
Z101AddFlag	Add Flag	Z101AFLG	D	Z101
BatchQuantity	Units - Batch Quantity	#BQT	D	41
Exemptions	Exemptions	OWEXE	D	77
WorkersCompPremBase	DOT4	S	99
AlternateCompanyName	Alternate Company Name	ALTC	D	04
DetachedAdjDomestic	Amount - Detached Adj. Domestic	DETD	D	45
DetachedAdjForeign	Amount - Detached Adj. Foreign	DETF	D	45
ApproverId	Approver I'd	EAPR	D	04
AaiCostCenterUse	AAI Business Unit Use	CCU	D	92
AaEndObject	AAI end object	AAIE	D	83
AaiObjectUseCode	AAI Object Use	OBJU	D	93
AaiSubsidiaryUse	AAI Subsidiary Use	SUBU	D	93
Rda429	AAI subtotaling	RDA429	D	H94

Field	Explanation
Search Description	<p>Categorizes data item names. Enter text in upper and lower case. The system uses this field to search for similar data items. To enter an alpha description, follow these conventions:</p> <ul style="list-style-type: none"> Dates – Begin all Date fields with Date Amounts – Begin all Amount fields with Amount Units – Begin all Unit, Quantity, and Volume fields with Units Name – Begin all 30-byte description fields with Name Prompt – Begin any Y/N prompting field with Prompt Address Number – Begin all address numbers (employee, customer, owner) with Address Number
Data Item	<p>An identifier that refers to and defines a unit of information. It is a 32-character, alphabetical field that does not allow blanks or special characters such as % & , . +.</p> <p>The data item cannot be changed.</p> <p>It forms the C-code data name (for example AddressNumber) that is used in business functions, data structures, and event rules.</p> <p>Also identify a data item by the alias or alpha description.</p>
Alpha Description	<p>Categorizes data item names. Enter text in upper and lower case. The system uses this field to search for similar data items. To enter an alpha description, follow these conventions:</p> <ul style="list-style-type: none"> Dates – Begin all Date fields with Date Amounts – Begin all Amount fields with Amount Units – Begin all Unit, Quantity, and Volume fields with Units Name – Begin all 30-byte description fields with Name Prompt – Begin any Y/N prompting field with Prompt Address Number – Begin all address numbers (employee, customer, owner) with Address Number

Field	Explanation
Alias	<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>
G G	<p>For World, a code which designates a type of data used to select data dictionary terms for printing. See User Defined Codes, system code '98', record type 'GG'.</p> <p>The data item names for error messages are assigned automatically.</p> <p>NOTE: If you need to assign your own error message numbers, use 4 digit numbers greater than '5000'.</p> <p>The data item name for a non-database field (used on a video or report but not in a file – glossary group U) must begin with a #, \$ or @.</p> <p>For help text (glossary group H), the data dictionary “Inquiry/Revision Program” field may be used to specify the name of a follow-on item.</p> <p>To create your own messages for the IBM message file (glossary group J), begin the data item name with your own three characters (e.g., CLT0001).</p> <p>For OneWorld, validates against UDC H98/DI. A code used to designate the type of data item.</p> <p>Items in glossary group D or S can be included in database tables. Items in other glossary groups (for example, error messages) cannot be added to a table.</p>

Field	Explanation
Syst Code	A user defined code (98/SY) that identifies a J.D. Edwards system.

Defining a Data Item

You define data item specifications when you add, modify, or copy a database data item. Defining a data item includes the following tasks:

- Creating a data item
- Naming a data item
- Defining general information
- Attaching default triggers
- Updating the glossary
- Defining jargon and alternate language terms

You can also perform one or more of the following tasks:

- Attaching a default value trigger
- Attaching a visual assist trigger
- Attaching an edit rule trigger
- Attaching a display rule trigger
- Attaching a user defined code trigger
- Attaching a next number trigger
- Attaching a smart field trigger

Creating a Data Item

Use the following process to create the base data dictionary item. Use the processes that follow this one to name and define the data dictionary item.

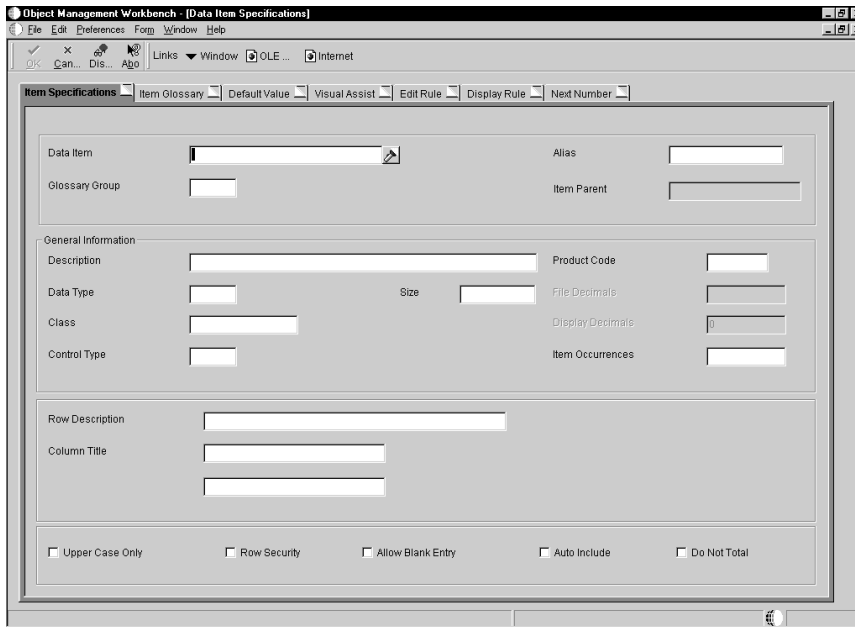
 **To create a data item**

1. From the Object Management Workbench, click Add.

2. On Add OneWorld Object to the Project, choose the Data Item option, and then click OK.

The system asks whether you want to create a normal data dictionary item or a glossary data item. Glossary data items are used primarily for batch error messaging.

3. To create a normal data dictionary item, click No.



Naming a Data Item

You must assign a unique name and attributes to a new data dictionary data item before it can be used by an application.

► To name a data item

1. On Data Item Specifications, complete the following fields:
 - Data Item
 - Alias
 - Glossary Group

Data Dictionary Item Naming Conventions

J.D. Edwards recommends the following naming standards. Refer to the *Development Standards: Application Design Guide* for more information about naming standards.

Data Item Name

The data item name is a 32-character, alphabetical field that identifies and defines a data item. You must allow enough room in the field name for a 30 percent expansion of the English text for translation.

The data item name forms the C-code data name (for example AddressNumber) that is used in business functions, data structures, and event rules.

When creating a J.D. Edwards data dictionary item, do not use a Y or Z in the first character of the data item name. Y and Z are reserved for Partners in Development business partners. (J.D. Edwards data items beginning with Y or Z may exist prior the B73.2 release of OneWorld. These items will remain J.D. Edwards items.)

Blanks and the characters % & , . + are not allowed as part of the data item name in OneWorld.

You can also identify a data item by the alias or alpha description.

Once added the data item name cannot be changed.

Data Item Name for an External Data Dictionary Item

When creating an external data item, you must use a Y or Z in the first character of the data item name to distinguish an external data dictionary item from a J.D. Edwards data dictionary item.

The data item name can be a maximum of 32 alphanumeric characters and uses the following format:

Yssssssssssssssssssssssssssssssssssssss, where:

Y or Z = designates an external data dictionary item and is the first digit of any JDE-assigned external system code

sss = the system code number assigned by the Partners in Development system administrator, or 55xx-59xx for enterprise-level development of new modules, or 60xx-69xx for JDE custom development

ssssssssssssssssssssssssssssssssssssss = the name of the data item name

Data Item Alias

The data item alias is an 8-character alpha code. If the data dictionary item is exclusive to OneWorld applications, the alias is five or more characters in length.

When adding a data item that will be used in a table by an RPG program, the alias must not exceed four characters.

The data item alias is used when searching, within database routines (application program interfaces used in business functions), and within Table Design when creating a table. For each table, a prefix is added to the alias, which makes it unique to this table. For example, ABMCU indicates that MCU is within the Address Book.

When assigning an alias, do not begin the alias with TIP or TERM. Aliases that begin with TIP are reserved for OneWorld tips information; aliases that begin with TERM are reserved for use by J.D. Edwards Publications to provide term glossaries in OneWorld guides.

Blanks and the characters % & , . + are not allowed as part of the data item alias in OneWorld.

You can also identify a data item by the data item name or alpha description.

Once added, you cannot change the data item name or alias.

Alias for an External Data Dictionary Item

An external data dictionary item is one that is created by a developer outside of J.D. Edwards for use in OneWorld. For external data items, the data dictionary alias can be a maximum of eight alphanumeric characters and uses the following format:

Ysssdddd, where:

Y or Z = designates external data dictionary item and is the first digit of any JDE-assign external system code.

sss = the system code number assigned by the Partners in Development system administrator, or 55xx-59xx for enterprise-level development of new modules, or 60xx-69xx for JDE custom development

dddd = the name of the data dictionary item

Field	Explanation
Data Item	<p>An identifier that refers to and defines a unit of information. It is a 32-character, alphabetical field that does not allow blanks or special characters such as % & , . +.</p> <p>The data item cannot be changed.</p> <p>It forms the C-code data name (for example AddressNumber) that is used in business functions, data structures, and event rules.</p> <p>Also identify a data item by the alias or alpha description.</p>
Alias	<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>

Field	Explanation
Glossary Group	<p>For World, a code which designates a type of data used to select data dictionary terms for printing. See User Defined Codes, system code '98', record type 'GG'.</p> <p>The data item names for error messages are assigned automatically.</p> <p>NOTE: If you need to assign your own error message numbers, use 4 digit numbers greater than '5000'.</p> <p>The data item name for a non-database field (used on a video or report but not in a file - glossary group U) must begin with a #, \$ or @.</p> <p>For help text (glossary group H), the data dictionary "Inquiry/Revision Program" field may be used to specify the name of a follow-on item.</p> <p>To create your own messages for the IBM message file (glossary group J), begin the data item name with your own three characters (e.g., CLT0001).</p> <p>For OneWorld, validates against UDC H98/DI. A code used to designate the type of data item.</p> <p>Items in glossary group D or S can be included in database tables. Items in other glossary groups (for example, error messages) cannot be added to a table.</p>

Defining General Information

After you have named a data item, you must provide additional general information on Data Item Specifications.

The screenshot shows a software window titled "Work With Data Dictionary Items - [Data Item Specifications]". The window has a menu bar (File, Edit, Preferences, Window, Help) and a toolbar with icons for OK, Cancel, Dismiss, Abort, and a dropdown menu for Links. Below the toolbar are several tabs: "Item Specifications" (selected), "Item Glossary", "Default Value", "Visual Assist", "Edit Rule", "Display Rule", and "Next Num".

The "Item Specifications" tab contains the following fields:

- Data Item:** AddressNumber
- Alias:** AN8
- Glossary Group:** D
- Primary Data Elements:** (checked)
- Item Parent:** (empty)

The **General Information** section includes:

- Description:** Address Number
- System Code:** 00
- Data Type:** 9 Numeric
- Size:** 8
- File Decimals:** (empty)
- Display Decimals:** (empty)
- Class:** (empty)
- Control Type:** 4 Generic Edit
- Item Occurrences:** (empty)

The **Row Description** section includes:

- Row Description:** Address Number
- Column Title:** Address
- Number:** (empty)

At the bottom of the window, there are several checkboxes:

- Upper Case Only
- Row Security
- Allow Blank Entry
- Auto Include
- Do Not Total

► To define general information

1. On Data Item Specifications, click the Item Specifications tab.
2. Complete the following required fields:
 - Description
 - System Code
 - Data Type
 - Control Type
 - Row Description

This description is for the base language only, unless you update the description for another language.

- Column Title

This description is for the base language only, unless you update the description for another language.

- Size
- File Decimals
- Display Decimals

3. Complete the following optional fields:
 - Class
 - Item Occurrences

Item occurrences is used for arrays. This allows you to create an item as a child of another item. The data dictionary performs validation to ensure that attributes are consistent between the parent and the child. If you change the parent item, the changes are duplicated to the child items. The data item names use the parent data item name and a number, for example, a parent item ABC and child items ABC1, ABC2, and so on.

4. Click any of the following options:

- Upper Case Only
- Row Security
- Allow Blank Entry
- Auto Include
- Do Not Total

Field	Explanation
Description	<p>Categorizes data item names. Enter text in upper and lower case. The system uses this field to search for similar data items. To enter an alpha description, follow these conventions:</p> <ul style="list-style-type: none"> Dates - Begin all Date fields with Date Amounts - Begin all Amount fields with Amount Units - Begin all Unit, Quantity, and Volume fields with Units Name - Begin all 30-byte description fields with Name Prompt - Begin any Y/N prompting field with Prompt Address Number - Begin all address numbers (employee, customer, owner) with Address Number
System Code	<p>A user defined code (98/SY) that identifies a J.D. Edwards system.</p>

Field	Explanation
Data Type – OneWorld	<p>The style or classification of data, such as numeric, alphabetic, and date.</p> <p>J.D. Edwards recommends that you do not change the data item type if it is used within an existing application. Otherwise, you must regenerate your table and review all business functions that use this data item.</p> <p>Data types include:</p> <p>Character a single letter, always the size of one</p> <p>Date a Date</p> <p>Integer an integer</p> <p>Character (Blob) can be translated from EBCDIC (8-bit character code commonly used on IBM mainframes) to ASCII (7-bit character code)</p> <p>Binary (Blob) cannot be translated, appears in machine code, and is found as an executable file under Win.help</p> <p>Binary a flag representing two choices. Usually a combination of the digits 1 and 0 to represent on and off, true and false.</p> <p>String always the same size or length</p> <p>Variable an item of variable size</p> <p>Identifier (ID) used within the program logic for controls. An ID is used to write a C program and reference third party software that returns a pointer. A JDE API then save the pointer that references the ID. The parameter passed to the C program is the ID.</p> <p>Numeric a long integer</p>
Size	<p>The field size of the data item.</p> <p>NOTE: All amount fields should be entered as 15 bytes, 0 decimals, and the data item type should be P (packed).</p>
File Decimals	<p>The number of positions to the right of the decimal of the data item that are stored. Addresses how the information is stored in the database.</p>

Field	Explanation
Display Decimals	Designates the number of decimals in the currency, amount, or quantity fields the system displays. For example, U.S. Dollars would be 2 decimals, Japanese Yen would be no decimals, and Cameroon Francs would be 3 decimals.
Control Type	<p>Defines the type of graphical user control that is associated with the data item. For example, a data item can appear as a push button, check box, user defined code, and so on.</p> <p>Control type is used by Forms Design to automatically add the correct control to a form for a specific data item. For example, if a data item will normally be used as a check box, then the Data Dictionary control type should be a check box. When you use a Quick Form, the data item will default as a check box control instead of a generic edit on your form.</p> <p>You can override in Forms Design; however, you should anticipate how it will most commonly be used and set it up accordingly.</p>
Row Description	<p>For World, creates the title on text and reports. It is used in a manner similar to the column description in the query facility. It should be less than 35 characters. Use abbreviations whenever possible. For example:</p> <ul style="list-style-type: none"> U/M Units of measure YTD Year-to-date MTD Month-to-date PYE Prior year end QTY Quantity G/L General ledger A/P Accounts payable DEPR Depreciation <p>For OneWorld, the row description is used for field description on forms and reports.</p>
Column Title	The first line of description that will be used in column headings on a report or form. This description should be no larger than the data item size, if possible. If the column heading is only one line, it should be placed in this column. Use the second line of the Column Title when one is not clear.
Class	Data item class. A class defines the essential attributes and characteristics of a data item. Informational only.

Field	Explanation
Item Occurrences	<p>In setting up a data item in the data dictionary, you may specify a number of array elements. This will cause the automatic creation of one additional data item for each array element.</p> <p>The array data item names are restricted to certain lengths depending on the number of array elements:</p> <ul style="list-style-type: none"> 3 bytes - 1 to 9 elements 2 bytes - 10 to 99 elements 1 byte - 100 to 999 elements
Upper Case Only	If the value of this field is a Y, the user will be allowed to enter only upper case characters in the entry control.
Allow Blank Entry	<p>Turning this option on allows a blank value to be written to the database under the following conditions:</p> <ul style="list-style-type: none"> (1) If the field is edited against a UDC table, a blank value will be allowed regardless of whether a blank value is valid for the table. (2) If the field is specified to be a mandatory entry, a blank value will be allowed as a valid entry.
Row Security	A flag to indicate that the field can be used in setting up row security.
Auto Include	Determines whether this column should be automatically included in all database fetches to tables that contain this item. This option should only be set for items that are essential for certain database trigger processes or security validation.
Do Not Total	Turning this option on designate that this data item should not be totalled. Certain numeric data items, such as address number (AN8) should never be totalled in a report or batch process.

Attaching Default Triggers

You use triggers to initiate display and edit routines associated with data items at application runtime.

There are several specific types of triggers you can attach, including:

- Attaching a default value trigger
- Attaching a visual assist trigger
- Attaching an edit rule trigger
- Attaching a display rule trigger

- Attaching a next number trigger
- Attaching a smart field trigger

► **To attach default triggers**

1. On Data Item Specifications, click the appropriate tab to attach any of the following applicable triggers to a data item:
 - Default Value
 - Visual Assist
 - Edit Rule
 - Display Rule
 - Next Numbers
 - Smart Field

Although you can override any of these triggers in Forms Design in the Edit Control Properties, you should anticipate how they will be used most often.

Field	Explanation
Default Value	Use this field to assign a default value to the data item if the field is blank. <ul style="list-style-type: none">• The value entered for numeric characters must be the exact same length as the data item and preceded with zeroes.• If this is a user defined code trigger, verify that the value entered is valid for the attached user defined code.
Visual Assist	There are three types of visual assist: <ul style="list-style-type: none">• Calculator - assigns a calculator to the field. Use this on all numeric data items where a user might need to type in an amount.• Calendar - assigns a calendar to the field. Use this on data items where a user might need to type in a date.• Search form - assigns a search form to the field. Use this on all fields where the valid values for the field are found in a file.

Field	Explanation
Edit Rule	<p>Use this to assign edit rules and procedures to a data item.</p> <p>Edit rules specify the editing technique that is applied when data is entered in the field. For example, value must be within a specified range.</p> <p>Use business functions to attach special logic that cannot be done through one of the edit rules. For example, if you would like to automatically verify the Address Book file every time you enter a Customer Number, then attach a procedure that says, Edit Address Book.</p>
Display Rule	<p>Use this to assign rules and procedures to a data item that governs how it is displayed.</p> <p>Display rules are keywords that describe a formatting technique that is applied when data is displayed.</p> <p>Use business functions when you need special processing that cannot be done through one of the five display rules. For example, if you would like to format the appearance of the account number, attach a business function called Display Account Number.</p>
Next Number	<p>Use this to assign next number logic to this data item.</p> <p>These rules include which system code is used to locate the next number and the array index number the data will use in the Next Number file (F0002).</p>
Smart Field	<p>This trigger is available for items in Glossary Group K and defines the business functions to be used to calculate a value and for column headings displayed on a report (using the reporting tool).</p>

Attaching a Default Value Trigger

A default value trigger automatically inserts a specified default value in a field when:

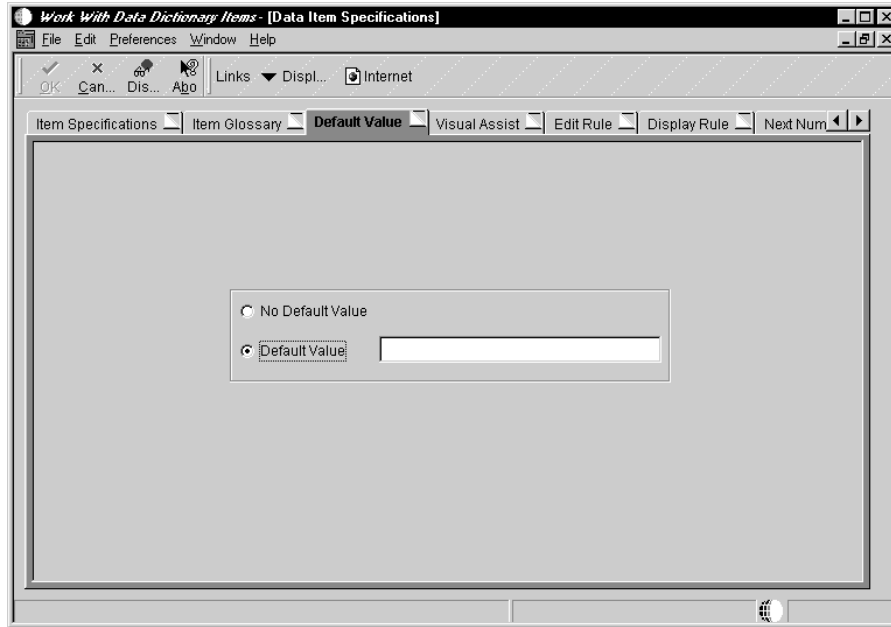
- Control is Exited
- Dialog is Initialized

You use the default value trigger for fields where the default value is appropriate in almost every situation. If a value is passed to this field through a processing option, business function, form interconnection, data structure, or if the user types in a value, this default value will not be used.

To attach a default value trigger

1. On Data Item Specifications, click the Default Value tab.

2. Click the following option:
 - Default Value
3. Enter the default value.



Field	Explanation
None Default Value	Indicates whether there is a Default Value on the field. Choose either of the following options: <ul style="list-style-type: none"> • None - no Default Value is assigned • Default Value - assigns a default value to the field. Specify the default value for entry that this field will have.

Attaching a Visual Assist Trigger

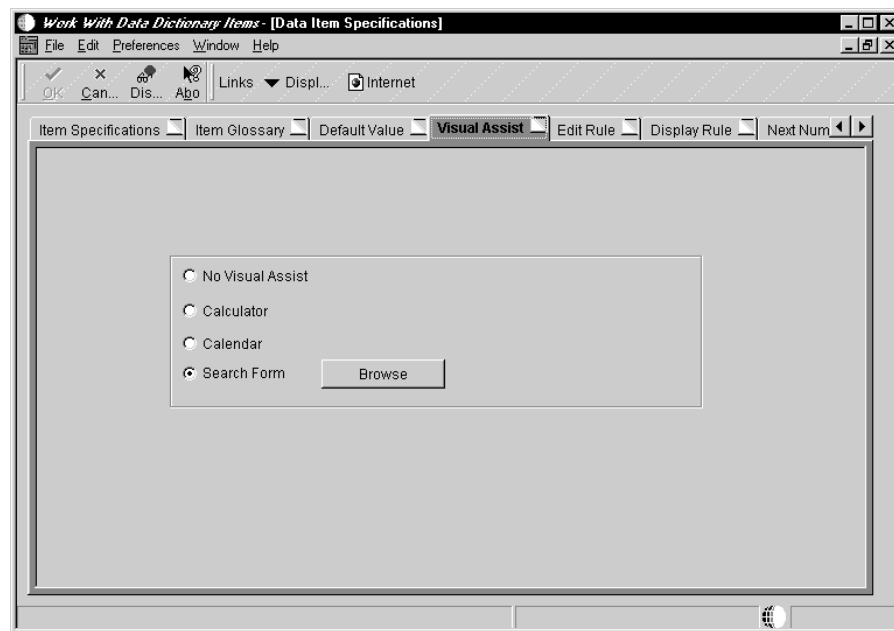
The following three types of triggers are available:

- Search form (flashlight)
- Calculator
- Calendar

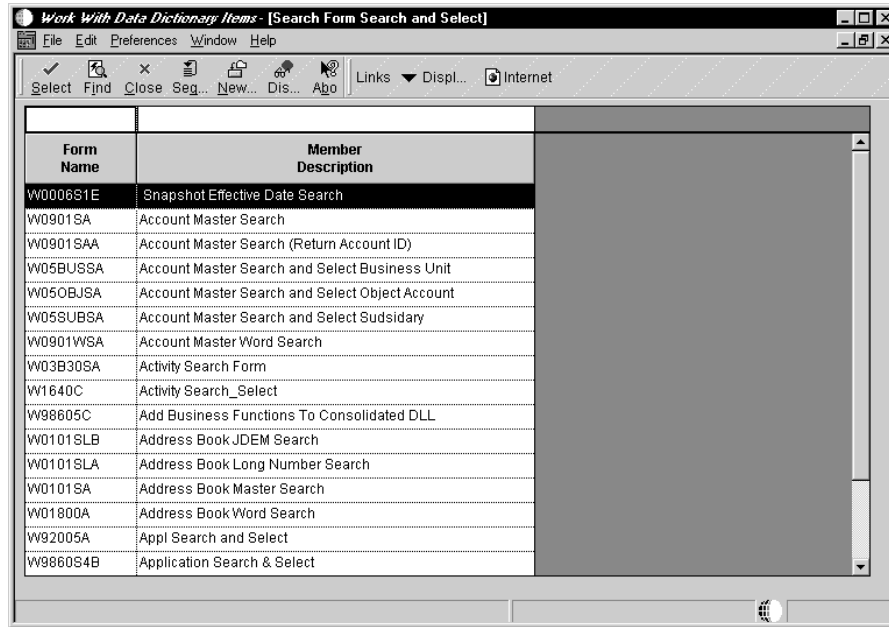
Use a search form trigger to link a field to a search form for locating valid values. The search form must exist before you attach a search form trigger.

▶ To attach a visual assist trigger

1. On Data Item Specifications, click the Visual Assist tab.
2. On Visual Assist, click one of the following options:
 - No Visual Assist
 - Calculator
 - Calendar
 - Search Form



3. If you click Search Form, click the Browse button to select the form you want to access for valid values.



If you attached a Search Form assist trigger, at runtime when you click the Search Form flashlight button, the selected form appears. You can enter search criteria and return with a valid value. These Search and Select forms are based on files, not UDC tables.

Field	Explanation
Visual Assist	<p>Indicates whether there is a visual assist on a field and the type. Choose from the following options:</p> <ul style="list-style-type: none"> • None - no Visual Assist is assigned • Calculator - assigns a calculator to the field. Use this on all numeric data items where a user might need to type in an amount. • Calendar - assigns a calendar to the field. Use this on data items where a user might need to type in a date. • Search form - assigns a search form to the field. Use this on all fields where the valid values for the field are found in a file.

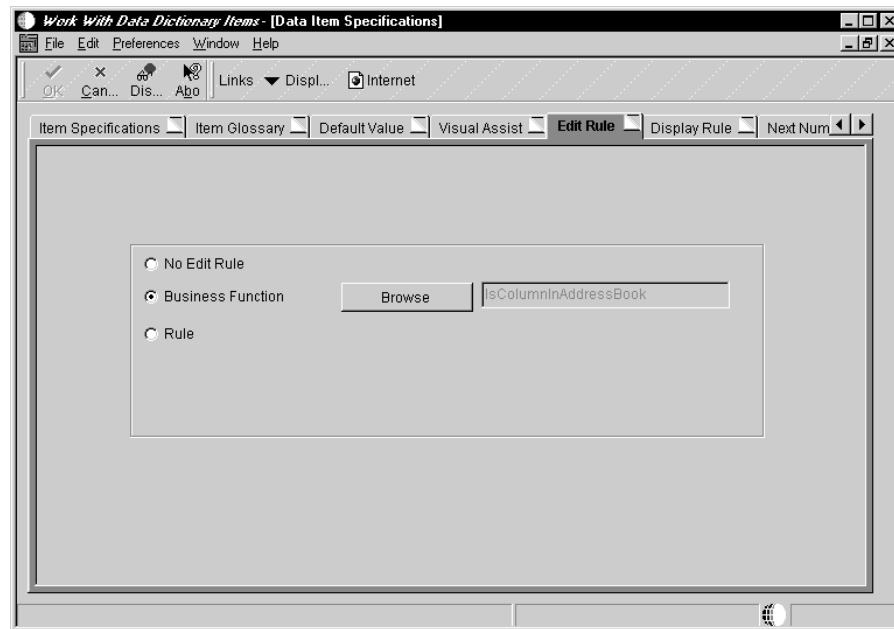
Attaching an Edit Rule Trigger

You use edit rule triggers to validate field values based on business functions or rules. For example, you can define a rule that:

- Validates and compares a field with a particular value
- Ensures that a field value is within a specified range of values
- Links a field to a specific UDC Search and Select form
- Checks for Y and N values

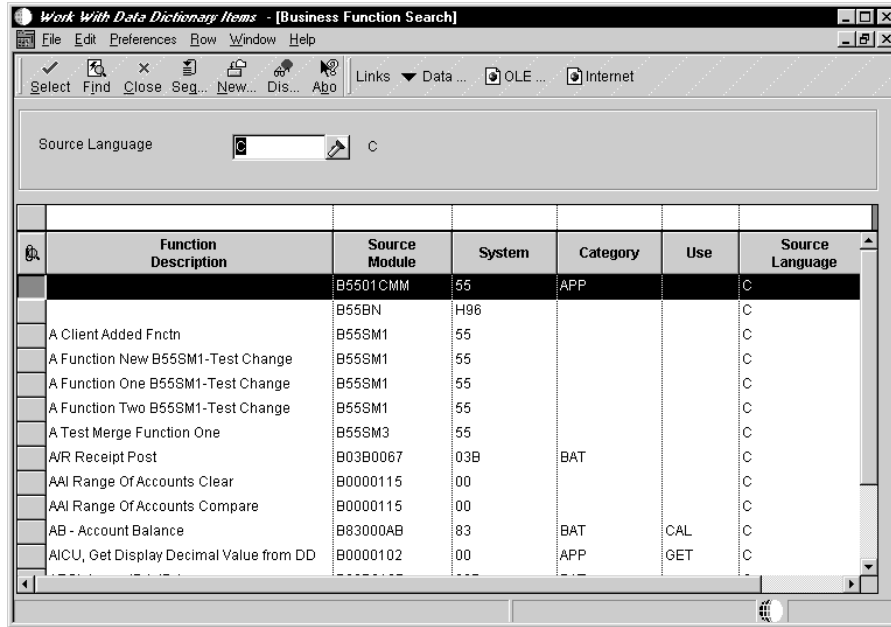
► **To attach an edit rule trigger for a business function**

1. On Data Item Specifications, click the Edit Rule tab.
2. On Edit Rule, click one of the following options:
 - Business Function
 - Rule



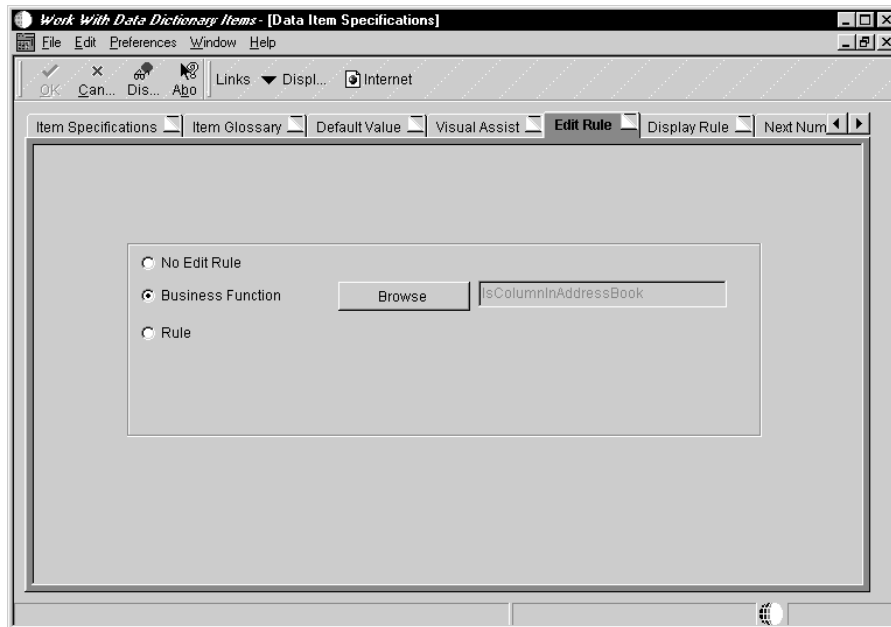
3. If you clicked the Business Function option, click the Browse button to choose from available business functions.

Create the business function if it does not exist.



4. Choose the desired business function.

Read the Attachments to learn what each function does.



5. If you clicked the Rule option, click the flashlight button to choose from the available rules.

These rules may include user defined codes such as:

EQ	Equal
GE	Greater or Equal
GT	Greater
HNDL	Table Handle
LE	Less Than or Equal
LT	Less Than
NE	Not Equal
NRANGE	Not Between
RANGE	Between
UDC	User Defined Code
VALUE	In a List
ZLNTH	Allocated Length (VARLEN flds)

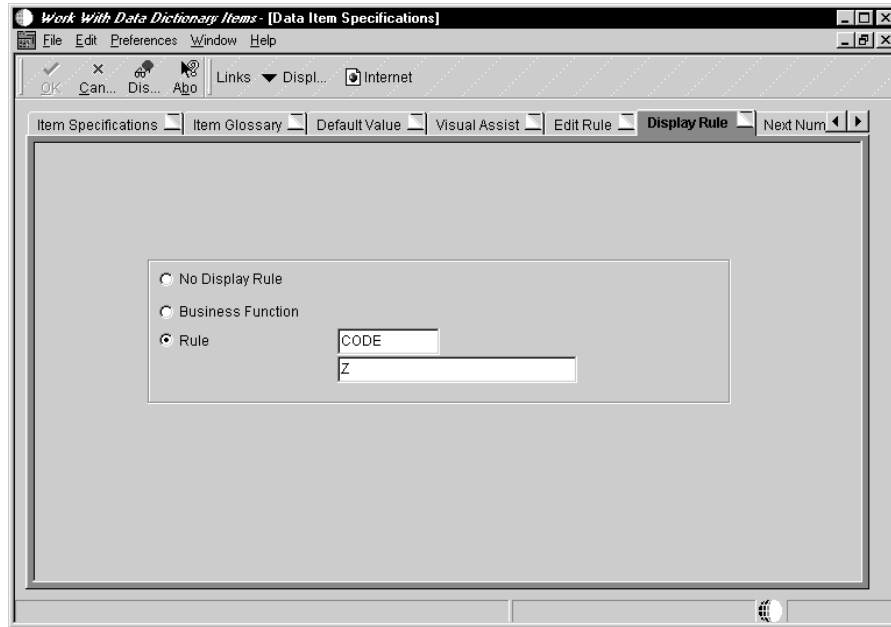
Attaching a Display Rule Trigger

You use a display rule trigger to format data. You attach a display rule trigger based on either a business function or a user defined code. The following two types of display rule triggers are available:

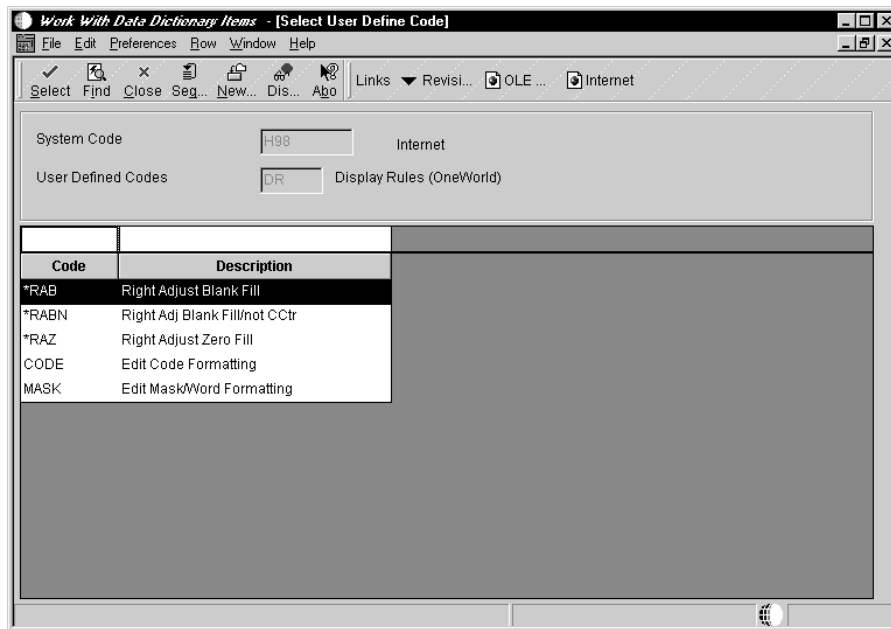
- Business Function
- Rule

To attach a display rule trigger

1. On Data Item Specifications, click the Display Rule tab.
2. On Display Rule, click the following option:
 - Rule



3. Click the Visual Assist for data display rules to browse and select from available user defined codes.



4. Choose from the following values:

***RAB** Right-adjusts the value and precedes it with blanks. Data items that define business units use this rule.

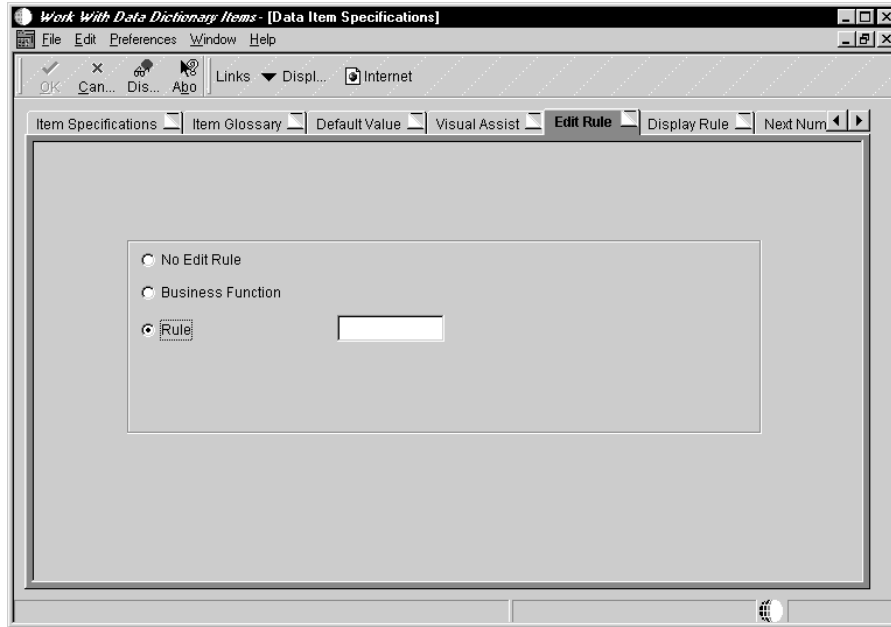
*RABN	Right-adjusts the value and precedes it with blanks. Data items that do not define business units use this rule.
*RAZ	Right-adjusts the the value and precedes it with zeroes. For example, Company would display as 00001. This is used for non-numeric fields only.
CODE	Uses the specified Edit Codes to format numeric fields. See UDC 98/EC for a list of valid codes. The code should be entered into the parameter field.
MASK	<p>Embeds the specified characters within the data when it is displayed. For example, to display Social Security Number (SSN_) with embedded dashes, the mask parameter would show as:</p> <p>bbb-bb-bbbb</p> <p>Mask can only be used with char or string Data Item types.</p>

Field	Explanation
Formatting	<p>Indicate whether or not there is a Display Rule for the field. Choose from the following options:</p> <ul style="list-style-type: none"> • None - no Display Rule • Business Function - assigns a business function ID to the field. Specify the business function procedure in the field that becomes available when you choose this option. Use the visual assist to view and select a business function. • Rule - assigns display rule to the field. Specify the display rule in the fields that become available when you choose this action. Use the visual assist to view and select rules.

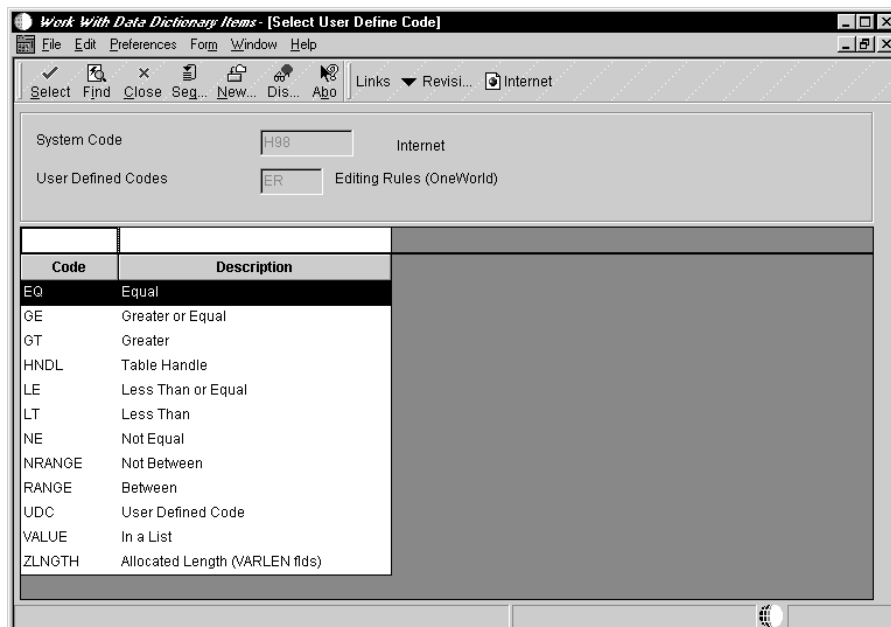
Attaching a User Defined Code Trigger

▶ To attach a User Defined Code trigger

1. On Data Item Specifications, click the Edit Rule tab.
2. On Edit Rule, click the following option:
 - Rule



3. Click the Visual Assist to browse and select UDC, User Defined Codes.



UDC is one of several choices that are available. Each option is described below:

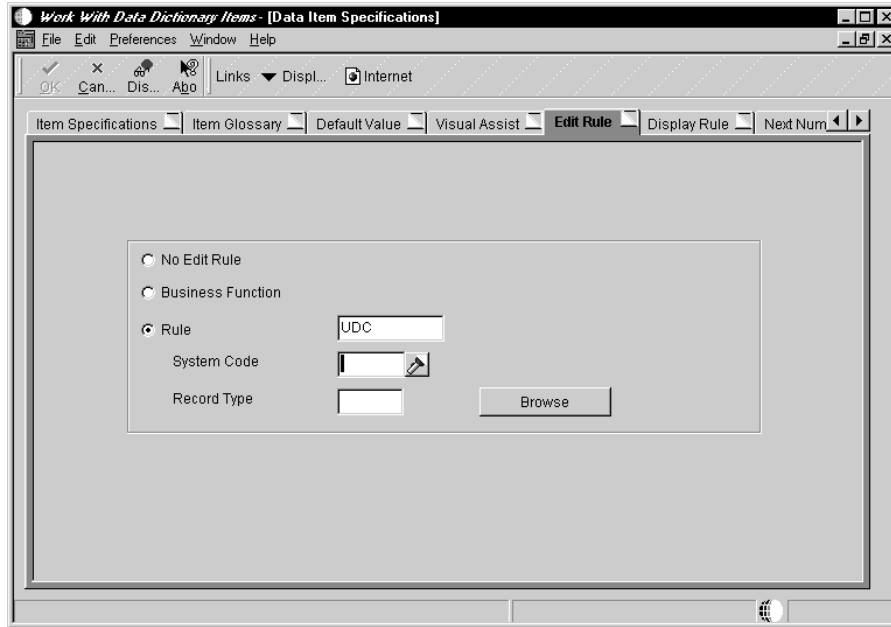
EQ Equal to a particular value

GE Greater than or equal to

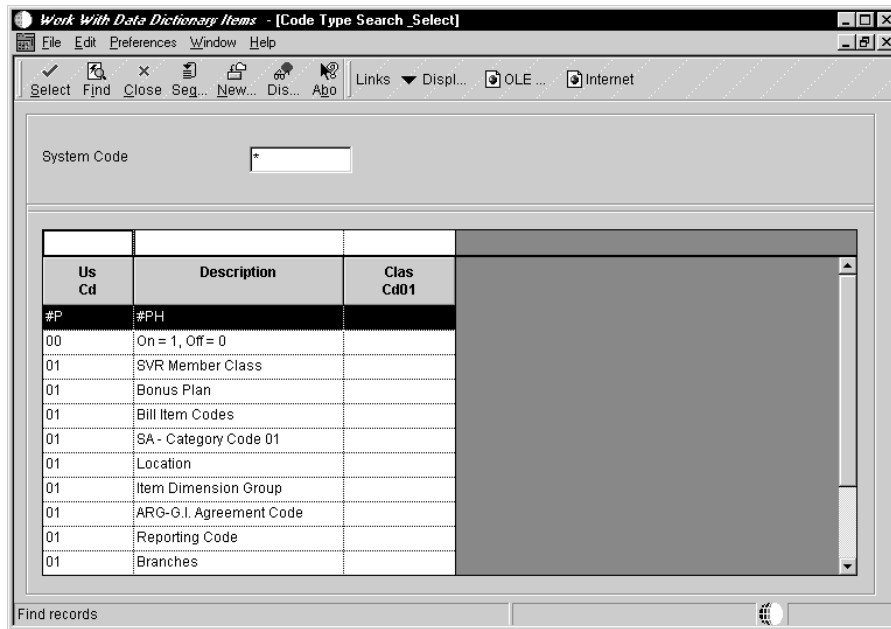
GT	Greater than
LE	Less than or equal to
LT	Less than
NE	Not equal
NRANGE	Not in range
RANGE	Range
UDC	Link to a specific UDC Search and Select form
VALUE	When Y N 1 0 are the only valid values; Y N 1 0 will default into the parameter field.
ZLNTH	AS/400 database only. The parameter field contains the allocated length for a variable-length field. This specification is optional.

When you enter UDC in the Rule field, the following two fields are enabled:

- System Code
- Record Type



4. Click the system code Visual Assist to choose a system code.
5. Click the Browse button to choose a Record Type.



Field	Explanation
System Code	A user defined code (98/SY) that identifies a J.D. Edwards system.
Record Type	A code that identifies the table that contains user defined codes. The table is also referred to as a code type.

Attaching a Next Number Trigger

The next numbers facility controls the automatic numbering for such items as new general ledger account numbers, voucher numbers, and address numbers. It allows you to specify what numbering system code you want to use and gives you a method of automatically incrementing numbers to reduce transpositions and keying errors.

Use a next number when you want a numeric data item to default in a preassigned next number if the user does not type in a number. Next numbers are assigned from an array. The combination of system code and index defines how the next number will be assigned.

Next Numbers

The Next Numbers table is F0002.

- 10 element array
- 1 record per system
- Modulus 11 check optional

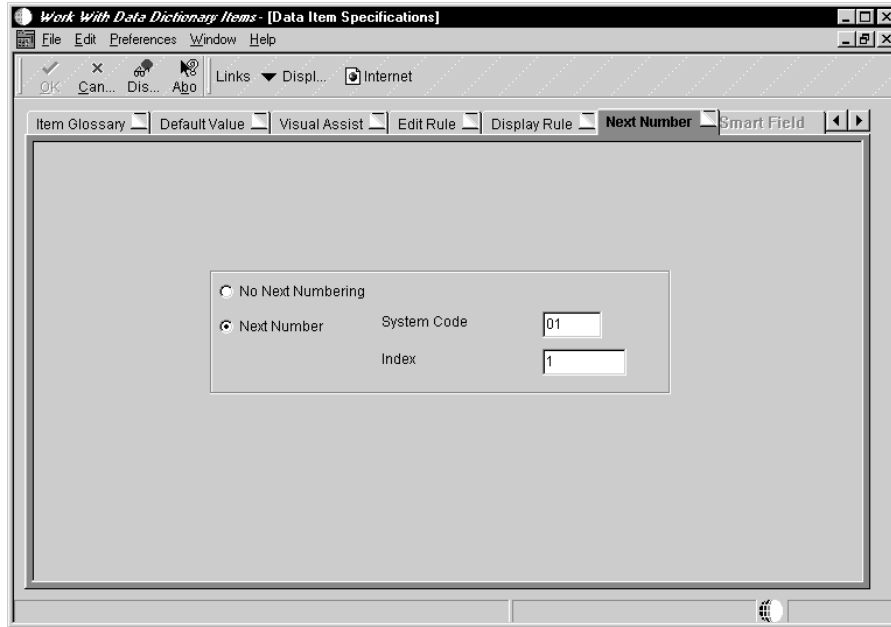
After you set next numbers, do not change it. If you change next numbers, the following problems occur:

- System performance is impacted.
- Next numbers will not duplicate numbers; when it reaches the maximum, it starts over
- You cannot change position or add a new entry without programming modifications

Next numbers ties with the data dictionary. A data item in the data dictionary points to the next number system. You can view the next numbers facility by using the OneWorld Fastpath G00.

To attach a next number trigger

1. On Data Item Specifications, click the Next Number tab.
2. On Next Number, click the following option:
 - Next Number



3. Complete the following fields:
 - System Code
 - Index

Field	Explanation
Next Number	Indicate whether there is a Next Number on this field. Choose either of the following options: <ul style="list-style-type: none"> • None - no Next Number is assigned • Next Number - assigns next number logic to the data item. Specify the Next Number System Code and Next Number Index the data uses in the Next Numbers file.
System Code	Designates the system number for the Next Number retrieval. See User Defined Codes, system code '98', record type 'SY'.
Index	The array element number retrieved in the Next Number Revisions program. For example, the next voucher number is array element '02' of system '04'.

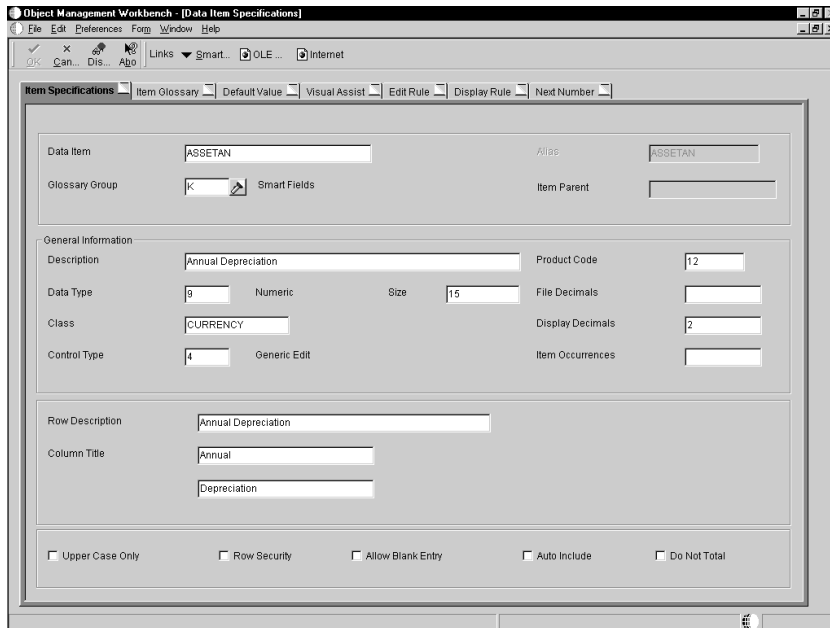
Attaching a Smart Field Trigger

Smart Fields are actually data dictionary items with attached business functions. The business functions that are attached include named mappings. This makes selecting a data item with particular functionality much easier. Instead of the end user having to know which business function to use and what parameters to pass, the user simply selects a data item that inherently has this information. Smart fields can be used in all section types in Report Design. For example, you

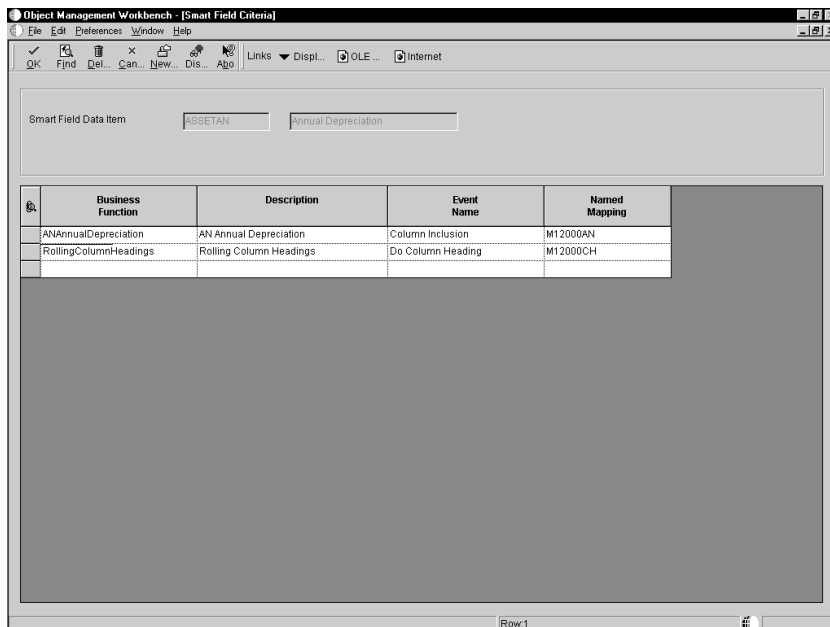
can use smart fields to derive a column heading or an object value in a tabular section. Smart fields are always glossary group K.

▶ **To attach a smart field trigger**

1. On Data Item Specifications, click the Item Specifications tab.
2. Complete the following required fields:
 - Description
 - System Code
 - Data Type
 - Control Type
 - Row Description
 - Column Title
 - Size
 - File Decimals
 - Display Decimals
3. Complete the following optional fields:
 - Class
 - Item Occurrences
4. Click any of the following options:
 - Upper Case Only
 - Row Security
 - Allow Blank Entry
 - Auto Include
 - Do Not Total



5. From the Form menu, choose Smart Fields.



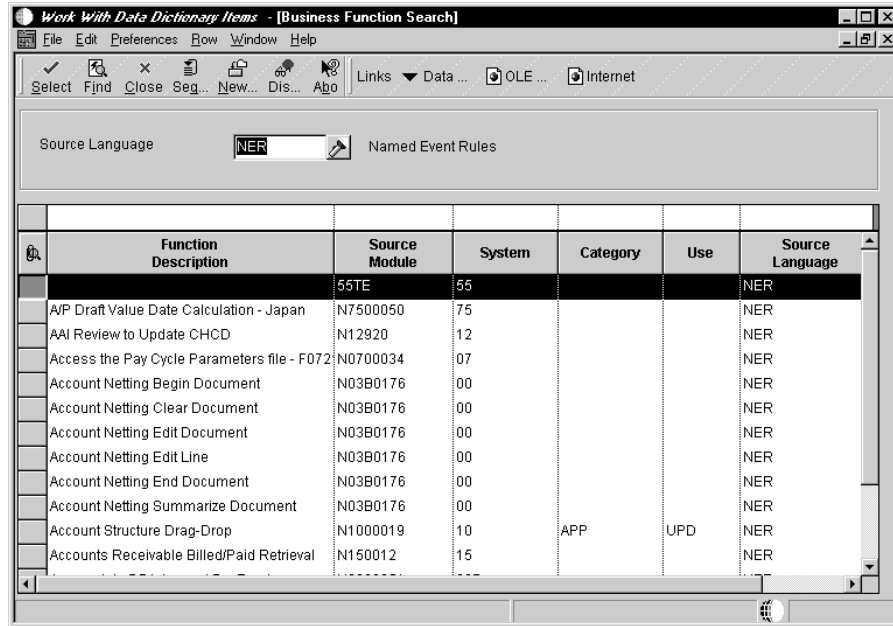
6. On Smart Field, complete the following fields:

- Business Function
Enter the business function that was created for the smart field.
- Event Name
Specify the event where the smart field should be triggered.

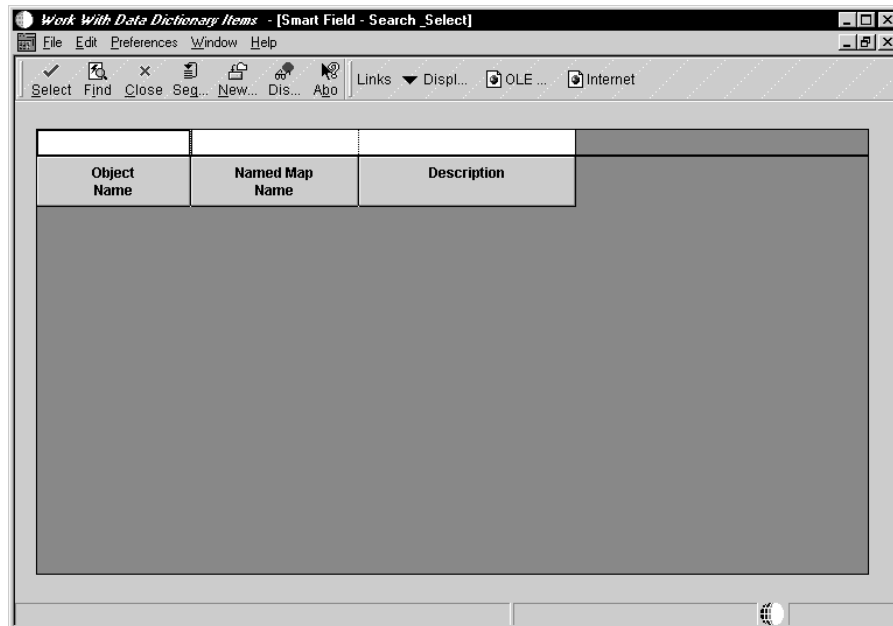
- Named mapping

Enter the named mapping that is associated with the business function data structure.

On Smart Fields, you can click the Browse button to select a business function.



On Smart Fields, you can click the visual assist on Named Mapping to select a value.



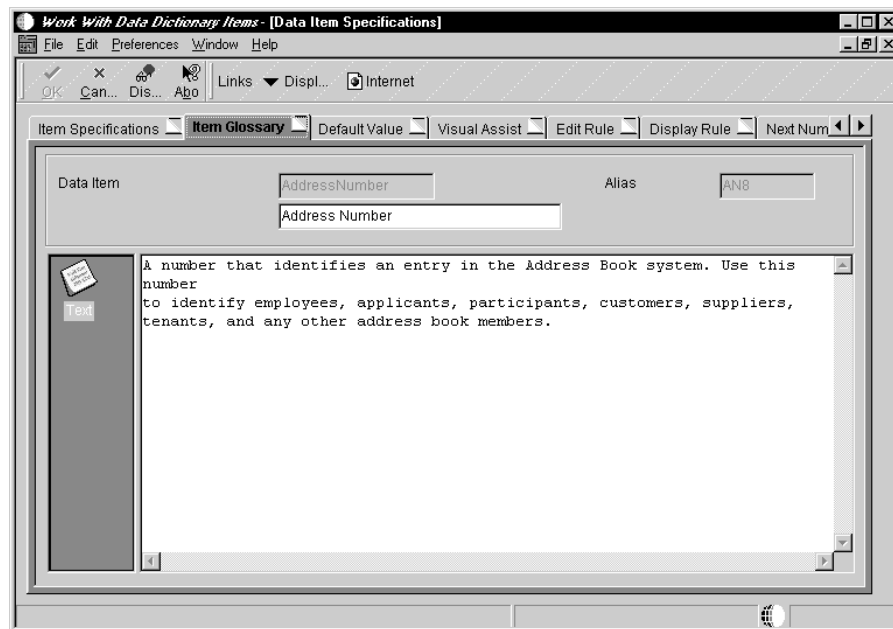
Updating the Glossary

Each data item is described within the glossary. This text is seen by the end user of an application at runtime. Thus, it should explain how the field is used in the application. Within an application, you can access this glossary description in field-sensitive help. You can also:

- Add glossary text for new data items
- Create form-specific or system-code-specific glossary text
- Create glossary text for different languages
- Customize J.D. Edwards glossary text to suit your needs

▶ **To update the glossary**

1. On Data Item Specifications, click the Item Glossary tab.



2. Complete the following field:

- Glossary

Field	Explanation
Glossary	Updates the glossary description that appears when field-sensitive help (F1 or on-line help) is requested for this data item field within an application. This description may be customized to suit your environment.

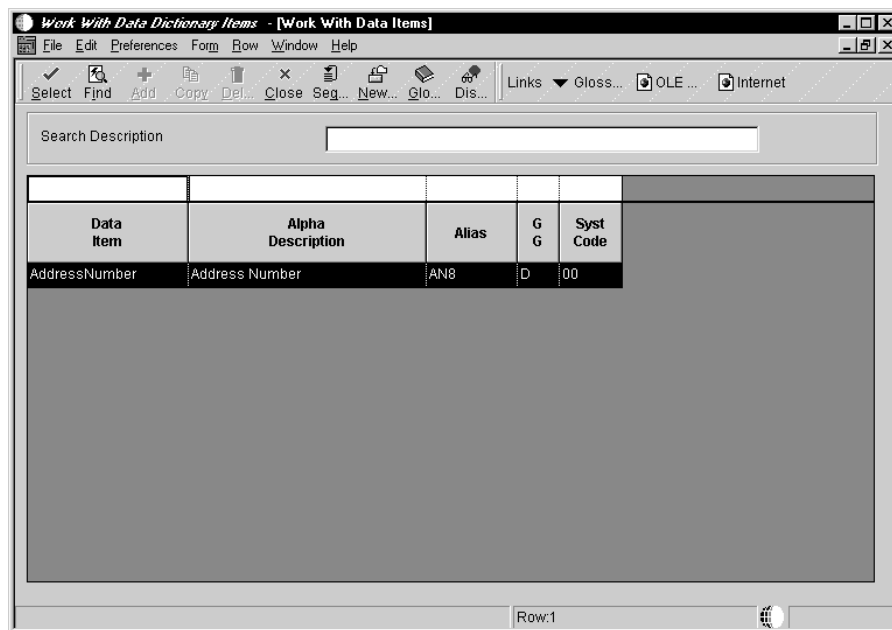
Entering glossary text for a form displays that glossary description as additional information when you are on that form only.

Adding Glossary Text for Languages

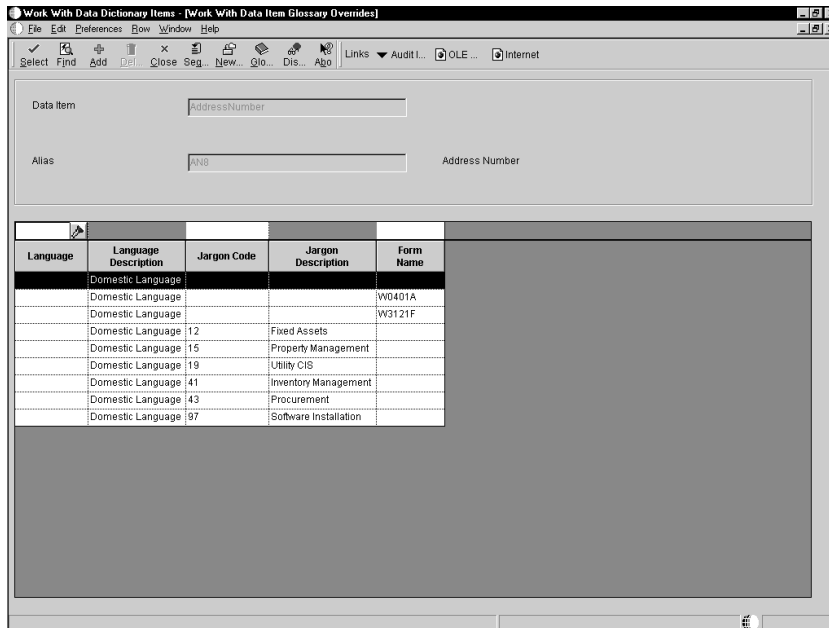
You can add separate glossary text for a new or existing item for different languages. For example, you can create glossary text for the base English language item and also have glossary text for French, Spanish, and German. Glossary text for languages must be added after the data dictionary item has been created.

► To add glossary for an item with languages

1. On Work with Data Items, choose the item you wish to change.



2. From the Row menu, choose Glossary Overrides.



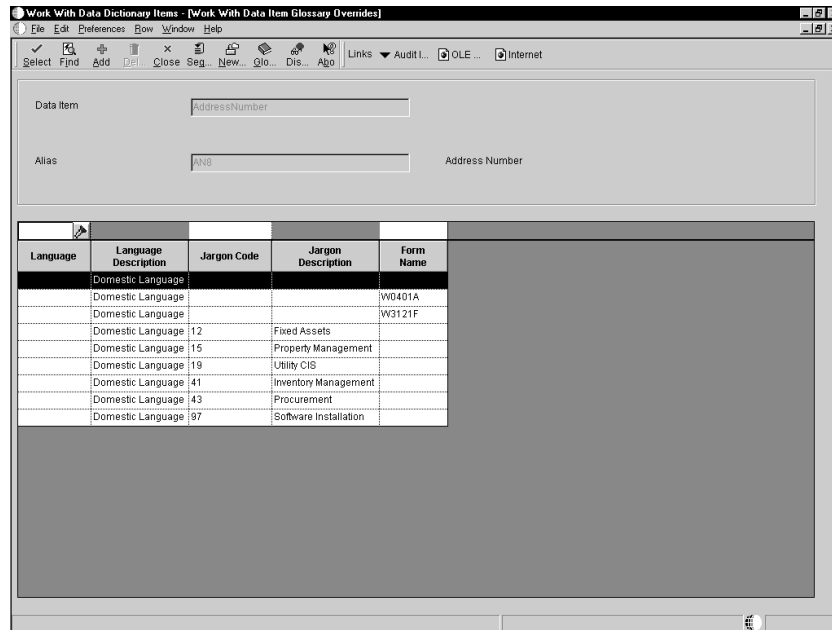
3. On Work with Data Item Glossary Overrides, click Add.
4. On Data Item Glossary Header, complete the following fields, and then click OK:

- Language
- Form

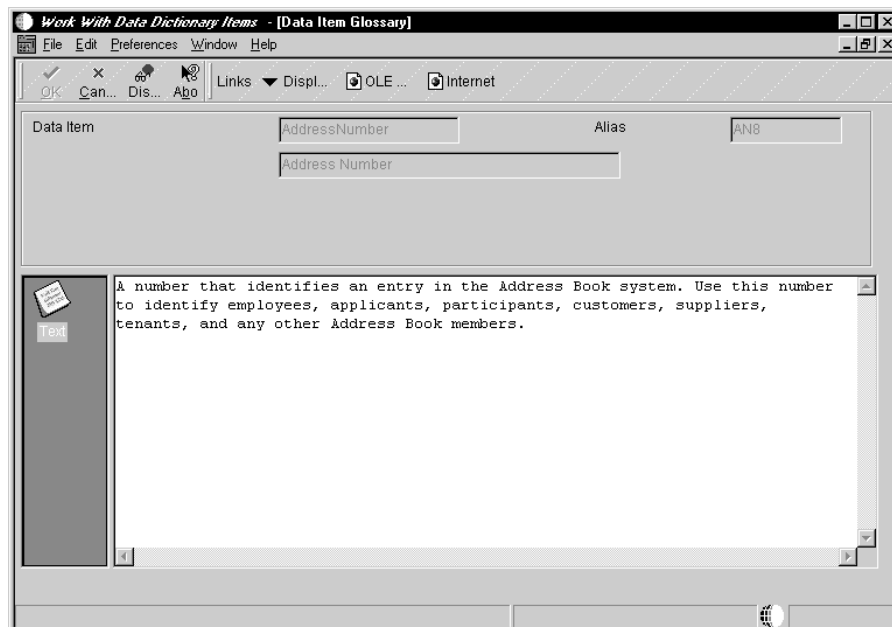
Enter in a Form Name if you want the glossary to be for a specific form only.

If you do not enter a form name, the glossary is applied to all forms that use this item.

5. On Work with Data Item Glossary Overrides, choose the row you just added.



6. From the Row menu, choose Glossary.



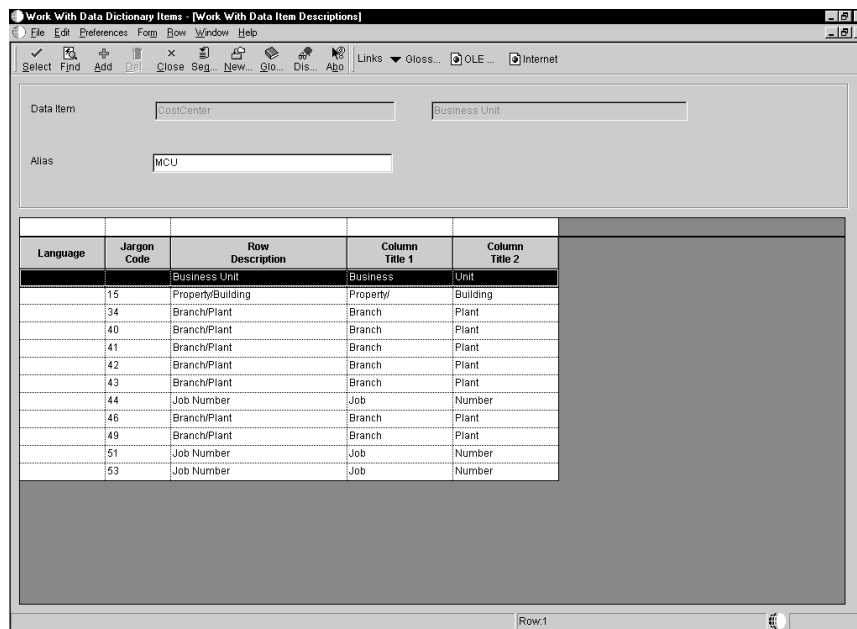
7. On Data Item Glossary, enter the glossary text you wish to add.

You can also change the glossary for an item with existing language overrides.

Defining Jargon and Alternate Language Terms

When you create a data dictionary item, you assign it descriptions for the row, column, and glossary. These descriptions may not offer the terminology flexibility you need, however. The solution is to assign alternate jargon or language descriptions to each item. Alternate descriptions allow the same data dictionary item to appear with different row, columns, and glossaries to different users, depending on the system (product) code of the object they are using.

For example, the cost center field MCU is widely used throughout the system. Its row description is Business Unit, which is a term used by financial applications. However, in distribution applications this same data item appears as Branch/Plant. In warehousing applications, the data item appears as Warehouse.



Be aware that in addition to any alternate terms you define, users can implement their own language overrides at the application level. OneWorld checks for and resolves overrides as follows:

First, if a user applies a language override in the application (such as the Form Director Aid or Report Design), OneWorld uses the term indicated by the language override, if one exists.

Second, if the user did not specify a language override in the application, then OneWorld determines at runtime if a system code has been attached to the menu selection. If the menu selection has an attached system code, then OneWorld displays the alternate term dictated by the system code, if one exists.

Third, if no alternate term has been indicated for the menu selection, OneWorld determines at runtime if a system code has been attached to the application. If

the application has an attached system code, then OneWorld displays the alternate term dictated by the system code, if one exists.

Fourth, if no alternate term has been indicated for the application, then the data dictionary text is shown.

In all cases, OneWorld first checks the user's preferred language for an alternate term before checking without language. Language and language overrides always take precedence over non-language overrides. For example, assume that in an environment where English is the base language, all the forms have French translations which a user can opt to view by selecting the French override. A form might contain a data dictionary item that in English has an alternate term; however, the French version of the data dictionary item does not have an alternate term in this example. When displayed in English, the form displays either the main term or the alternate term as appropriate. When displayed in French, however, the form displays only the main term, even when an alternate term is called for, because the language override takes precedence over displaying the alternate term.

Jargon and alternate language terms must be added after the data dictionary item has been created.

Perform the following tasks:

- Defining jargon
- Updating a data item for language
- Changing row and column text for all applications

See Also

- Updating the Glossary* for information on how to create alternate language glossary items for a data item

► **To define jargon**

1. On **Work with Data Dictionary Items**, find the item you wish to change.
2. From the **Row** menu, choose **Descrip. Overrides**.
3. Click **Add**.

The **Data Item Descriptions** form appears.

4. On **Data Item Descriptions**, complete the following fields, and then click **OK**:
 - **Jargon Code**

Enter or select the system code with which you want to associate the current jargon. Different system codes are associated with specific categories and applications in the system.

- Row Description
 - Column Title
5. Repeat this process for each jargon term you want to associate with the data item.

► **To update a data item for languages**

1. On Work with Data Dictionary Items, find the item you wish to update.
2. From the Row menu, choose Descrip. Overrides.
3. Click Add.

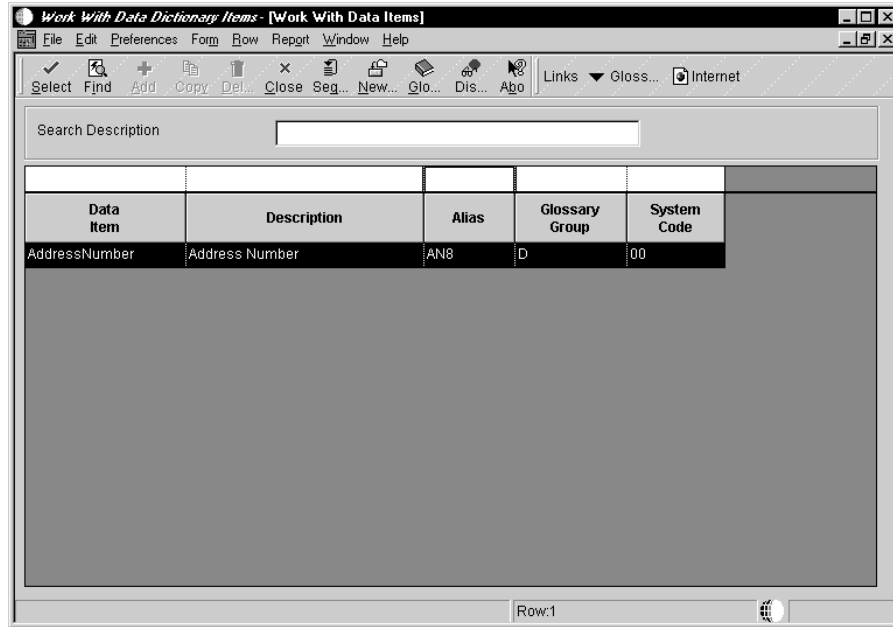
The Data Item Descriptions form appears.

4. Complete the following fields and click OK:
 - Language
 - Row Description
 - Column Title
5. Repeat this process for each language you want to associate with the data item.

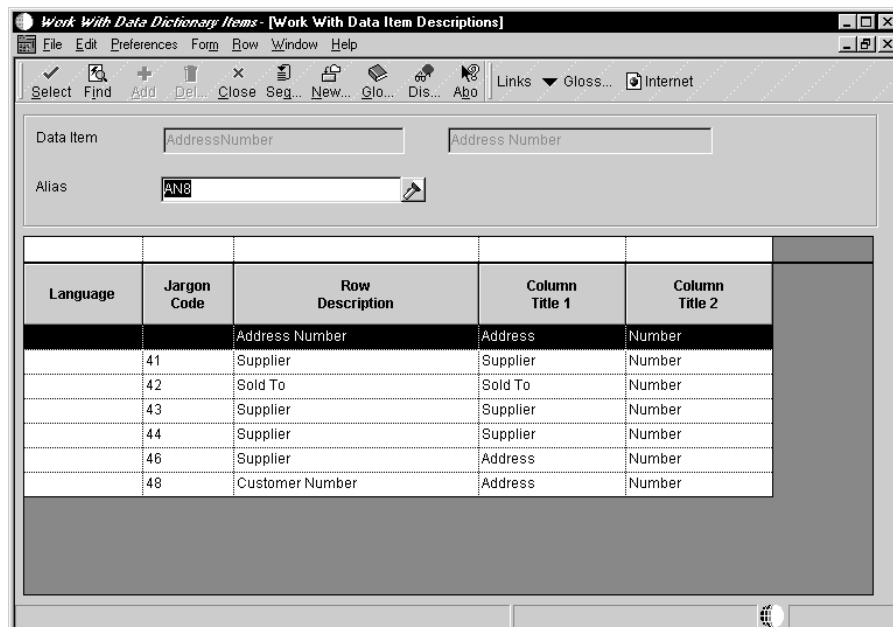
You can change the row and column text for all applications (interactive or batch) that use a data item.

► **To change row and column text for all applications**

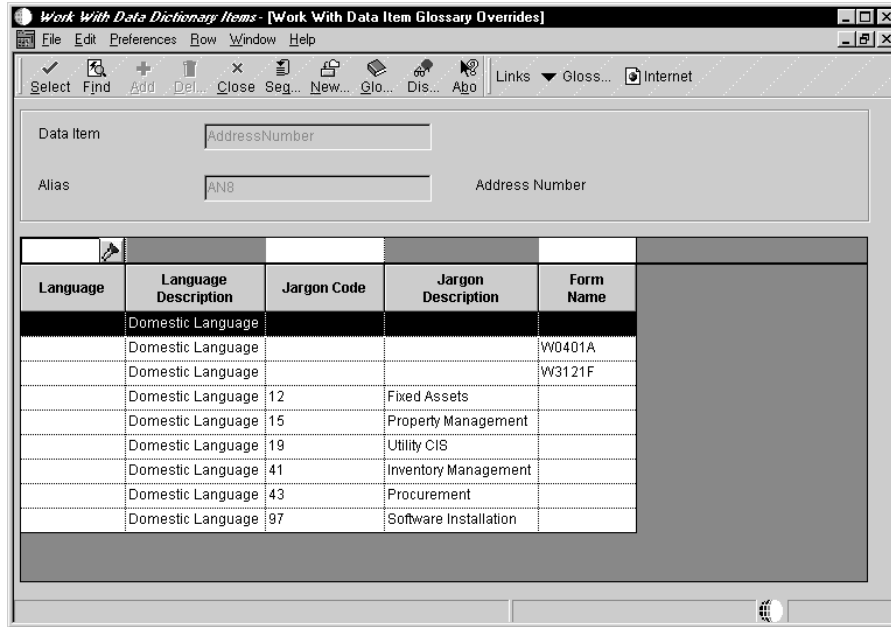
1. On Work with Data Dictionary Items, find the item you wish to update.



- From the Row menu, choose Descrip. Overrides.



- Choose a language record and modify it.



You can change the alpha description by clicking on Glossary Overrides and creating a record. You then use the Glossary form to change the description.

For information about changing the row and column text for a particular application (interactive or batch) that use this data item refer to the *Package Management Guide*.

Changes to row and column descriptions are not replicated through data replication. To deploy row and column changes to workstations you must deliver a new full or partial package, or an update package with the applications affected. This deletes the existing row and columns that are stored in a cache on the workstation.



Table Design

A relational database table stores data that an application uses. You can create one or more tables for use in an application. To create a table, you select data items (the data items must exist in the data dictionary) for a table and assign key fields as indices for retrieving and updating data. You must define your table so that OneWorld recognizes that the table exists.

You must use Table Design to generate the table whenever want to perform the following:

- Create a new table
- Add or delete a data item
- Add or modify an index

A table stores a set of data in columns and rows. Each column is a data item. Each row is a record.

An index identifies records in a table. A primary index identifies unique records in a table. An index is composed of one or more keys, or data items, within the table. An index enables a database management system (DBMS) to sort and locate records faster.

Table design is composed of the following topics:

- Adding a table
- Working with table design
- Viewing the data in tables



Adding a Table

Determine whether an existing table contains the data items required by your application. If not, you must add a new table.

► To add a table

1. On Object Management Workbench, click Add.
2. On Add OneWorld Object to the Project, choose the Table option, and then click OK.

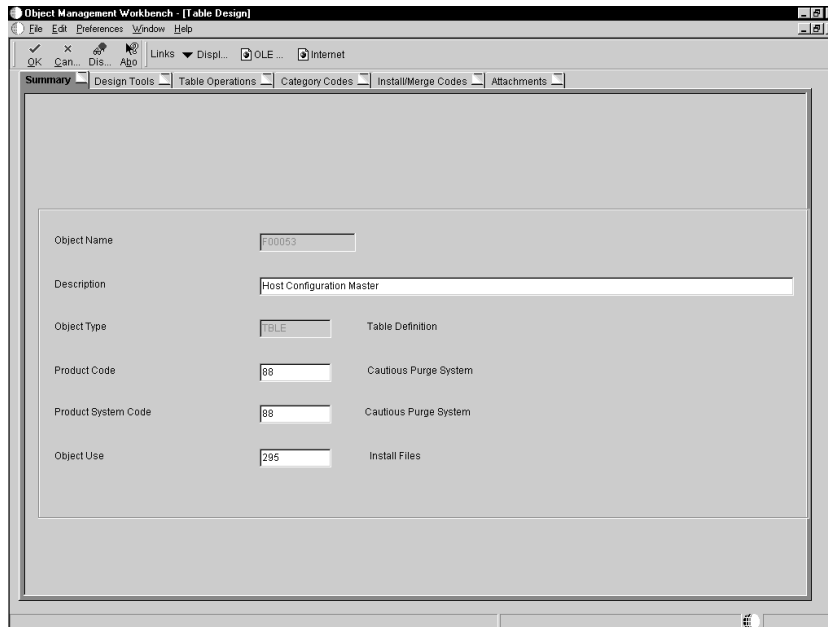
The screenshot shows the 'Add Object' dialog box in the Object Management Workbench. The dialog is titled 'Object Management Workbench - [Add Object]'. It has a menu bar with 'File', 'Edit', 'Preferences', 'Window', and 'Help'. Below the menu bar is a toolbar with icons for 'OK', 'Cancel', 'Dis...', 'Ab...', 'Links', 'Displ...', 'OLE...', and 'Internet'. The main area contains several input fields: 'Object Name' with the value 'F55001', 'Object Type' with the value 'TABLE', 'Description' with the value 'Sales Processing Codes', 'Product Code' with the value '55' and a note 'Reserved for Clients', 'Product System Code' with the value '55' and a note 'Reserved for Clients', 'Column Prefix' with the value 'JD', and 'Object Use' with the value '2' and a note 'Files'.

3. On Add Object, complete the following fields, and then click OK:
 - Object Name
See the J. D. Edwards naming standards below.
 - Description
See the J. D. Edwards naming standards below.
 - Product Code

- Product System Code
- Prefix – File

See the J. D. Edwards naming standards below.

- Object Use



4. On Table Design, click the Summary tab and change the data in the following fields to alter the table properties:
 - Description
 - Product Code
 - Product System Code
 - Object Use
5. To document the table, click the Attachments tab, and then add attachments.

See *Working with Attachments* for more information about adding attachments to an object.

J.D. Edwards recommends you use the following naming conventions when you add a table:

The Object Librarian name for a table can be a maximum of 8 characters and should be formatted as follows: **Fxxxxyyy**

F = data table

xx (second and third digits) = the system code, such as

00 - OneWorld Foundation environment

01 - Address Book

03 - Accounts Receivable

xx (fourth and fifth digits) = the group type, such as

01 - Master

02 - Balance

1X - Transaction

yyy (sixth through eighth digits) = object version, such as programs that perform similar functions but vary distinctly in specific processing

LA through LZ - Logical file

JA through JZ - Table join

Provide up to a 60-character description for a table.

The table description is the topic of the table. If it came from the AS/400, it should be the same name as the file it represents, such as Address Book Master (F0101) and Item Master (F4101).

The column prefix is a two-character code used to uniquely identify table columns. The first character must be numeric and the second character must be alpha-numeric.

Indices

List the field as the index name, such as Address Number, if there is only one field in the index.

For coexistence, it is critical that OneWorld indices match logicals on the AS400. When you run the Generate Table command in Table Design, OneWorld automatically looks to the AS400 and checks to see if a matching AS400 file exists. If a matching AS400 file does not exist, then OneWorld creates logical files on the AS400. If a matching AS400 file exists, OneWorld does not create any logicals on the AS400.

If there are two fields in the index, list them consecutively, such as Address Number, Line Number ID.

List the first two followed by an alpha character (A), such as Address Number, Line Number, A if there are more than two fields in the index and the first two

fields are the same as the first two fields of another index. Otherwise list the fields followed by a (+), such as Item Number, Branch, +.

Place a comma (,) and space between each index field and between last index field and the plus sign.

Do not include more than 10 fields in an index.

The total length of the index name cannot exceed 19 characters if the index has two or more fields. If you exceed 19 characters, the compiler will give you a warning of “Re-definition is not identical...”. This will impact fetches using the wrong index ID in business functions.

Field	Explanation
Object Name	The OneWorld architecture is object-based. This means that discrete software objects are the building blocks for all applications, and that developers can reuse the objects in multiple applications. Each object is tracked by the Object Librarian. Examples of OneWorld objects include: <ul style="list-style-type: none"> • Batch Applications (such as reports) • Interactive Applications • Business Views • Business Functions • Business Functions Data Structures • Event Rules • Media Object Data Structures
Description	The description of a record in the Software Versions Repository file. The member description is consistent with the base member description.
Product System Code	A code that designates the system number for reporting and jargon purposes. See UDC 98/SY.
Product Code	A user defined code (98/SY) that identifies a J.D. Edwards system.
Prefix – File	A prefix associated with a particular table. The prefix is placed before the data dictionary data item name to give the field a unique name across J.D. Edward’s systems.
Object Use	Designates the use of the object. For example, the object may be used to create a program, a master file, or a transaction journal. See UDC 98/FU.
Source Language	The source language code indentifies the programming language that a business function is written in.

Field	Explanation
Process Type	The Process Type groups objects by operation, such as report, conversion, or batch process. It is edited on the 98/E4 UDC table.

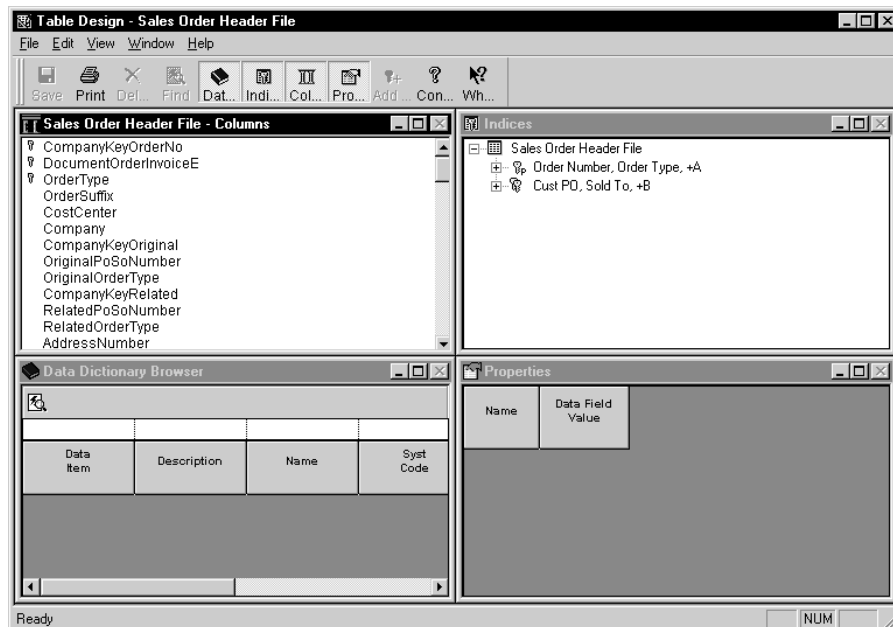
The following fields are typically required for an audit trail.

USER	User ID
PID	Program ID
UPMJ	Date Updated
JOBN	Workstation ID
UPMT	Time Last Updated

Working with Table Design

Table Design presents the following views within a single window:

- Table Columns displays the data items that make up your table.
- Data Dictionary Browser locates data items for selecting and moving to the Column view.
- Indices defines the unique data items for quicker sorting and updating of the table.
- Properties displays data item attributes for a selected data item in the Column view. This is a display view only and is primarily used when defining indices.



When you modify or delete data items or indices, you must reconfigure the table. Changes may affect business views and forms that reference that table.

- You use Generate to generate a newly modified table. The existing data in the table will be lost.

Caution: If you delete a table or delete columns from that table, any business views that reference the table or the deleted table columns will be invalid and will produce error messages when you generate the application.

If you use Table Design to delete a table, it only deletes the specs. It does not delete the physical table.

► **To launch the Table Design Aid**

1. Add a new table, or in Object Management Workbench choose a table and click the Design button in the center column.
2. On Table Design, click the Design Tools tab, and then click Start Table Design Aid.

Use Table Design to complete the following tasks:

- Choosing data items for the table
- Defining indices
- Previewing tables

Choosing Data Items for the Table

Table columns are the data items that store information used by an application.

- A data item must exist in the data dictionary before you can use it in a table.
- Tables can contain data items from multiple system codes.

► **To choose data items for the table**

1. On Table Design, in the Data Dictionary Browser view, use QBE to locate the data dictionary items you want to include in your table.
2. To include a data item in your table, drag it from the Data Dictionary Browser view to the Table Columns view.
3. To remove a column from a table, choose it and choose Delete from the Edit menu.

Defining Indices

You use indices to find specific records and to sort records faster. Table indices are like tabs in a card file. Each index is comprised of one or more keys, which are individual data items. You use indices to access data in the most simple manner so you do not have to read the data sequentially.

OneWorld middleware generates an SQL statement that the native database understands.

A table can have multiple indices; however, every table must have only one primary, unique index. The primary, unique index is the one unique identifier for each record in the table. The database should use the index that returns the most detail. It does not always use the primary index. Additionally the primary index is used to build business views.

See Also

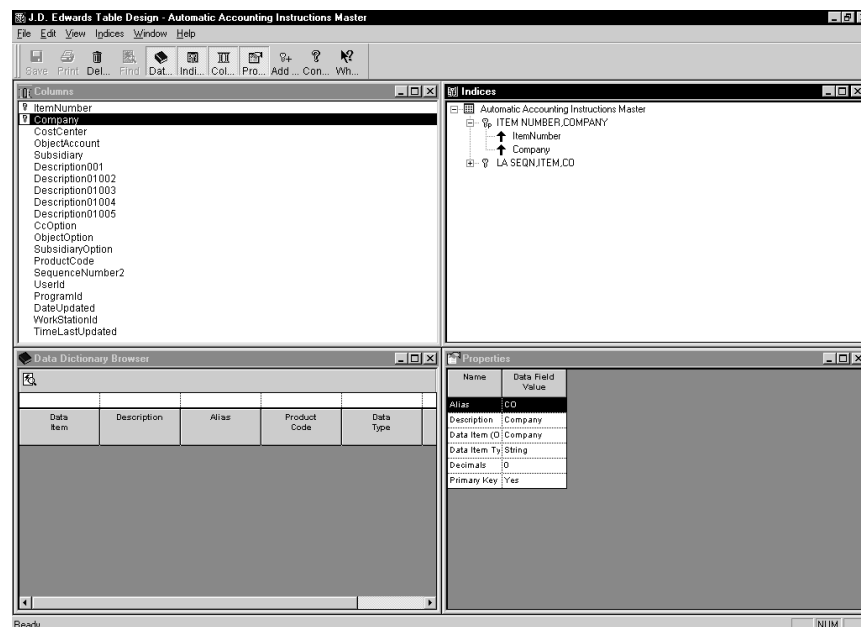
- *Performance* for performance issues regarding indices
- *Business View Design* for how business views use the primary index from a table

► To define indices

1. On Table Design, focus on the Indices form so the Indices menu appears.
2. From the Indices menu, choose Add.

You can also drag indices from the column view into the index view.

The index description is Untitled; it is marked with a key that displays the letter P to indicate a primary index.



3. Double-click the index title to name the index. After you name the index, press Enter.

4. On the Table Columns view, choose one or more columns from the columns view and drag them to the index.

A unique index is marked with a single key. You can toggle the unique/not unique status of a key by clicking your right mouse button and choosing Unique from the Index menu. The Unique Primary Index cannot be changed to nonunique status.

5. Indicate the ascending/descending sort order for an index column. An arrow pointing in the upward direction indicates that the index column is sorted in ascending order.

J.D. Edwards Design Standards

Use the following guidelines to name an index.

List the field as the index name, such as Address Number, if there is only one field in the index.

For coexistence, it is critical that OneWorld indices match logicals on the AS/400. When you run the Generate Table command in Table Design, OneWorld automatically looks to the AS400 and checks to see if a matching AS/400 file exists. If a matching AS/400 file does not exist, then OneWorld creates logical files on the AS/400. If a matching AS/400 file exists, OneWorld does not create any logicals on the AS/400.

If there are two fields in the index, list them consecutively, such as Address Number, Line Number ID.

List the first two fields followed by an alpha character (A), such as Address Number, Line Number, A if there are more than two fields in the index and the first two fields are the same as the first two fields of another index. Otherwise list the fields followed by a (+), such as Item Number, Branch, +.

Place a comma (,) and space between each index field and between the last index field and the plus sign.

Do not include more than 10 fields in an index.

The total length of the index name cannot exceed 19 characters if the index has two or more fields. If you exceed 19 characters, the compiler will give you a warning of “Re-definition is not identical...”. This will impact fetches using the wrong index ID in business functions.

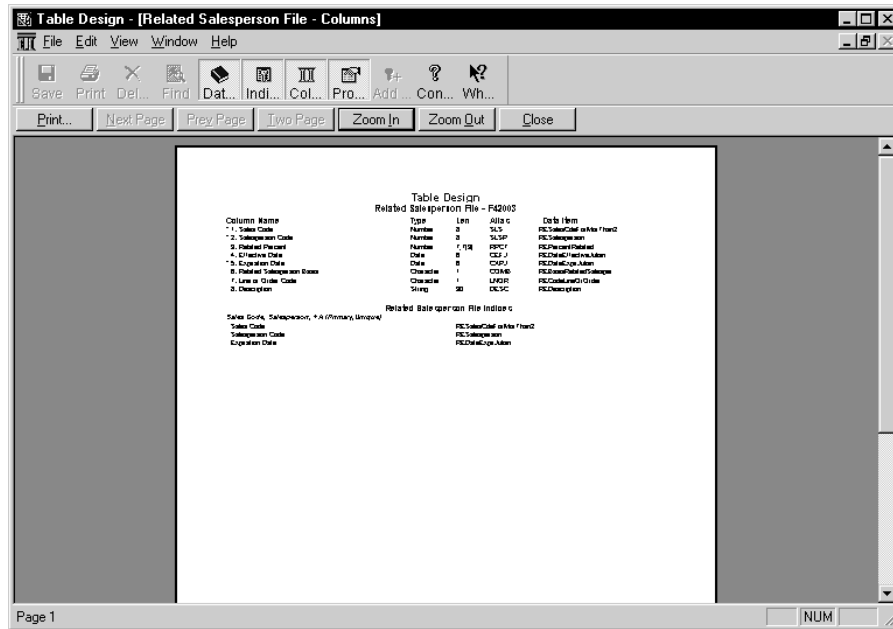
Previewing Tables

You can preview how tables will look when printed.

► **To preview your table**

1. On Table Design, focus on the Columns form and from the File menu, choose Print Preview.

A print preview of your table displays.



Working with Tables

The Object Management Workbench (OMW) provides a central location from which to manage your tables. This topic describes the following tasks:

- Generating tables
- Generating indexes
- Generating header files
- Copying tables
- Removing tables from the database

Generating Tables

After you have selected data items and assigned indices for your table, you are ready to configure the table for a specific data source. If you have not established an index your generation will fail.

You must generate a table to create the physical table. You cannot add or update to the table until it is generated. Table generation also creates a .h file that is used for compiling in business functions and table event rules.

The OMW calls the Object Configuration Manager application (P986101) to configure tables. You can configure the table within any existing data source. If you do not indicate a data source, OneWorld automatically configures the table for the default map. You can change the path code to generate the table in a different location. This actually does a drop statement similar to the remove table, and then the table is recreated. If you regenerate the current table, the data in it will be lost.

You must regenerate a table if you modify it, for example if you delete or add data items. To ensure that data is not lost, you must export your data, generate the table, and copy the data back.

To generate tables

1. On Table Design, click the Table Operations tab, and then click Generate Table.

2. On Generate Table, complete the following fields, and then click OK:
 - Data Source
 - Password

A message appears indicating whether the generation was successful.

Generating Indexes

If you create additional indices or modify existing ones, you must regenerate them. This modifies the .h file, but you will not lose existing data as you do when you regenerate the entire table.

To generate indexes

1. On Table Design, click the Table Operations tab, and then click Generate Indexes.
2. On Generate Indexes, complete the following fields, and then click OK:
 - Data Source
 - Password

Generating Header Files

Occasionally, you might have tables that have no header files. Use this process to generate header files without having to generate the entire table.

To generate header files

1. On Table Design, click the Design Tools tab, and then click Generate Header File.

The system generates the header file.

Copying Tables

You can copy tables from one data source to another. This operation does not copy table specifications. You can also use table conversion to copy tables from one data source to another.

See Also

- *OneWorld Table Conversion* guide for more information on using the Table Conversion tool to copy tables

▶ To copy tables

1. On Table Design, click the Table Operations tab, and then click Copy Table.
2. Complete the following fields, and then click OK:
 - Data Source (Source)
 - Data Source (Destination)
 - Object Owner ID
 - Password

Field	Explanation
Data Source	OneWorld uses this data source if the primary data source or the data item in the primary data source cannot be located. To Data Source: Where users want to move ActivEra data.
Object Owner ID	The database table prefix or owner.

Removing Tables from the Database

To remove a table completely from the system, you must use the OMW function, Remove Table from Database. If you use Table Design to delete a table, it only deletes the specifications. Table Design cannot physically delete a table.

▶ To remove tables from the database

1. On Table Design, click the Table Operations tab, and then click Remove Table from Database.
2. On Remove Table, complete the following fields, and then click OK:
 - Data Source
 - Password

Viewing the Data in Tables

If you want to view the data in tables in different databases, you can use the Universal Table Browser. This tool lets you verify the existence of data in a table as well as determine the table's structure. The Universal Table Browser uses JDEBASE APIs to retrieve data from the database, making it independent of the database you access.

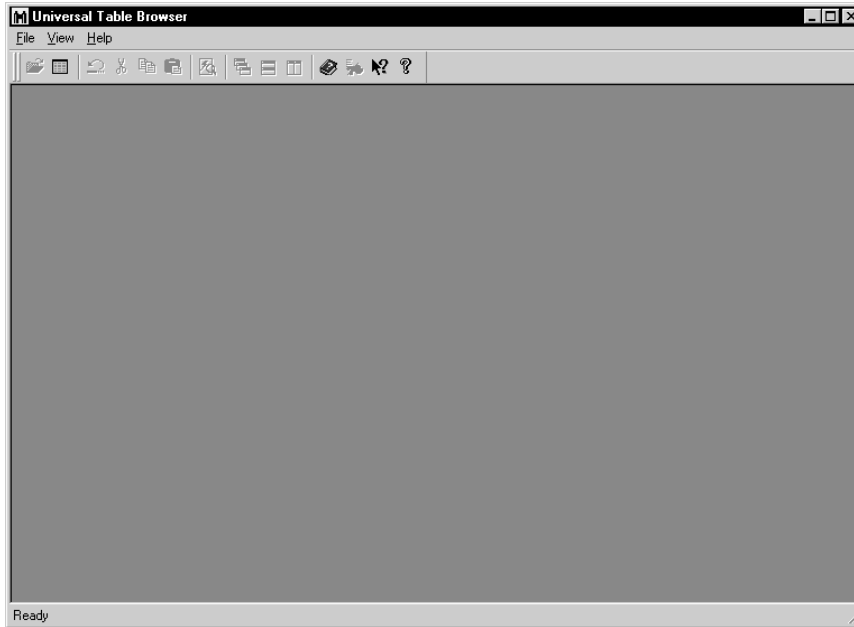
Note: You cannot use OneWorld security to directly secure users from the Universal Table Browser, because it is a Windows executable application, not a OneWorld generated application. However, you can place form security on the Table and Data Source Selection form (W98TAMC). This effectively secures the Universal Table Browser, because the Windows executable cannot function without this OneWorld form. All column and row security that you set up through Security Workbench apply to the Universal Table Browser.

Complete the following tasks:

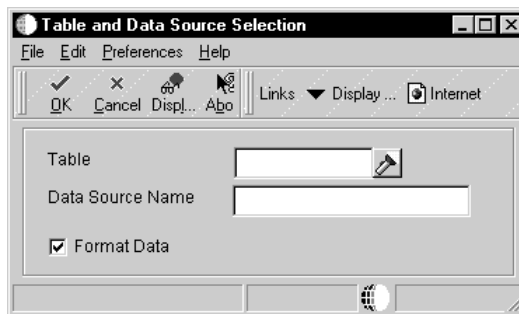
- View the data in tables
- View column properties in a table

To view the data in tables

1. On Cross Application Development Tools (GH902), choose Universal Table Browser.



2. On Universal Table Browser, choose Open Table from the File menu.



3. Complete the following required fields:
 - Table
 - Data Source Name
4. Complete the following optional field:
 - Format Data

Field	Explanation
Object Name	<p>The OneWorld architecture is object based. This means that discrete software objects are the building blocks for all applications, and that developers can reuse the objects in multiple applications. Each object is stored in the Object Librarian. Examples of OneWorld objects include:</p> <ul style="list-style-type: none"> • Batch Applications • Interactive Applications • Business Views • Business Functions • Business Functions Data Structures • Event Rules • Media Object Data Structures <p>..... <i>Form-specific information</i></p> <p>Indicates the name of the OneWorld table. For example, F0101 is the Address Book master. You can use the visual assist to enable the search and select form and locate a table.</p>
Data Source Name	<p>A valid data source in which the table resides. This default value is obtained from the OCM settings in the environment to which the user is signed on. Use the visual assist to enable the data search form and select any OneWorld data source.</p>
Format Data	<p>Indicates whether you want the Universal Table Browser to format portions of the data (default) or whether you want to view raw data.</p> <p>Formatted. The Universal Table Browser displays the data according to the specifications of the OneWorld data dictionary item. For example, assume that the data item PROC is a numeric field of size 15, with 4 display decimals. For a value of 56.2185, the Universal Table Browser displays a formatted value (using the data dictionary editing) as 56.2185, even though this value is stored in the database as 562185.</p> <p>Nonformatted. The Universal Table Browser displays the data according to the specification of the database and the data item type (such as numeric) from which the data came. For example, assume the table data item, PROC, is a numeric field stored in the database. Depending on the database, this field might default to a size of 32 with a precision of 15 being a numeric data type. Because OneWorld does not store the decimals in the database, a value 56.2185 would be stored and displayed in the database as 562185.0000000000000000.</p>

Example: Universal Table Browser (Formatted Data)

In this example, a database table is shown as it was opened with the Format Data option turned on. Notice that the structure of the information in the ABAN8 column of table F0101 is formatted using the data dictionary specifications.

	ABAN8	ABALKY	ABTAX	ABALPH	ABDC	ABMCU	ABSIC	ABL
1			43.078.849/001	Financial/Distrib	FINANCIALDIST	1		
50				Project Managen	PROJECTMANAG	1		
60				Financial Report	FINANCIALREPC	1		
70				French Compan	FRENCHCOMP	1		
77				Canadian Comp	CANADIANCOMF	1		
80				Colombian Com	COLOMBIANCOF	1		
200				Manufacturing/Di	MANUFACTURIN	1		
249				Model Energy &	MODELENERGY	1		
1001			73.058.653/000	Edwards, J.D. &	EDWARDSJDCC	1		
2006			523785321	Walters, Annette	WALTERSANNE	1		
2129			343298761	Jackson, John	JACKSONJOHN	1		
3001				Global Enterpris	GLOBALENTERI	1		
3002				Atlantic Corporat	ATLANTICCORP	1		
3003				CSC Corporatio	CSCCORPORAT	1		
3004				Pacific Company	PACIFICCOMPAI	1		
3005				Technology Syst	TECHNOLOGYS	1		
3333				Continental Inco	CONTINENTALIF	1		
3480				Digger In corpora	DIGGERINCORF	1		
4010				Colorado State T	COLORADOSTA	1		

Grid Rows: 40
Ready

Example: Column Properties

In this example, the column properties are shown for the OneWorld data dictionary item USEQ. The SQL database name for this OneWorld item is DTUSEQ.

The screenshot shows a dialog box titled "Column Properties" with a close button (X) in the top right corner. The dialog is divided into two main sections. The left section contains the following fields:

SQLColumnName:	DTUSEQ
Long Name:	UserDefinedCodeSequence
Alias:	USEQ
Size:	4
ID Dictionary:	USEQ
System Code:	00
Data Type:	EVDT_MATH_NUMERIC
Decimals Stored:	0
Decimals Displayed:	1
Currency Column:	0

Below these fields is an "OK" button. The right section contains the following fields:

Glossary Group	
Can Have Security ?	
Next Number System:	
Search Form Name:	
Edit Rule:	
Display Rule:	CODE:4
C Driver Type:	JDEDDB_C_DOUBLE
Offset in Buffer:	39
Actual Type:	8
Precision:	15
Scale:	0



Business View Design

A business view is a selection of data items from one or more tables. After you create a table, use Business View Design to select only the data items that are required for your application. OneWorld uses the business view to generate the appropriate SQL statements needed to retrieve data from any of the supported OneWorld databases. After you have a business view, you can create a form that updates data (in an interactive application) or design a report that displays data. Because you select only those data items that are required in an application, there is less movement of data over the network.

A business view:

- Can contain data items from more than one table
- Can contain all or some of the data items in a table or tables
- Links a OneWorld application to a table or tables
- Is required to create an application or generate a report
- Defines the data items from multiple tables that an application uses (as in table joins or table unions)

This section discusses the following:

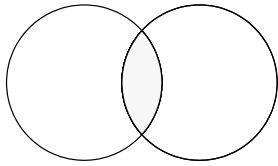
- Adding a Business View
- Working with Business View Design

Table Join

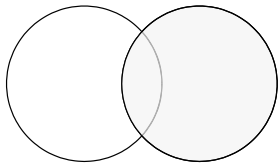
A table join combines data from individual rows of joined tables. The joining columns satisfy a join condition, such as when the rows have the same values in key columns. The primary table is the table you are starting from (usually the table on the left in table design) and the secondary table is the table you are going to (usually the table on the right in table design). There are several types of joins, including the following:



- Simple Joins are also known as inner joins. They include only rows that match both the primary and secondary tables.



- Right Outer Joins include rows common to both the primary and secondary tables, and unmatched rows from the secondary, table.



- Left Outer Joins include rows common to both the primary and secondary tables, and unmatched rows from the primary, table.

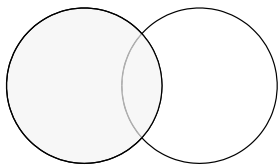
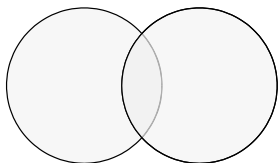


Table Union

A table union appends entire tables. It presents to the application selected rows from the first (primary) table first, then rows with corresponding columns from the second (secondary) table. If the rows from the two tables contain identical data, then only one of the records is retrieved in the union.

- Unions include rows from the first table and corresponding columns from the second table.



Select Distinct

You can use select distinct to help eliminate duplicate rows in the business view query.

Indices

The primary index from a table is used to build business views. Business views then carry information from the table to an application. If you need additional information other than the primary key carried forward to the application, you may need to change the business view index.

Adding a Business View

Before you begin designing a business view, think about the purpose of your application and the data items that it requires. Identify which OneWorld table or tables contain those data items. If you add a new business view, it does not impact performance. However, if you try to use an existing view that contains many more columns than you need, it may affect performance.

Business views usually contain more fields than are used on the form or grid. If you need a field that is in a business view, but not on a form or a grid, you can add the field without changing the business view.

You use different business views for each form type. For forms with grids, the grid should contain fewer fields than the business view to allow for business-specific issues that might affect performance. Typically, Search and Select should have the minimum number of items you need, so it is smaller. It should include only the basic fields needed for filtering and necessary output fields, such as descriptions.

Find/Browse and Parent/Child forms have a few more items and are more medium sized. They include those fields needed for filtering and necessary output fields, such as descriptions.

Input-capable forms have all of the items from the table and tend to be larger. They should include all of the fields necessary to add or update a record, including audit information.

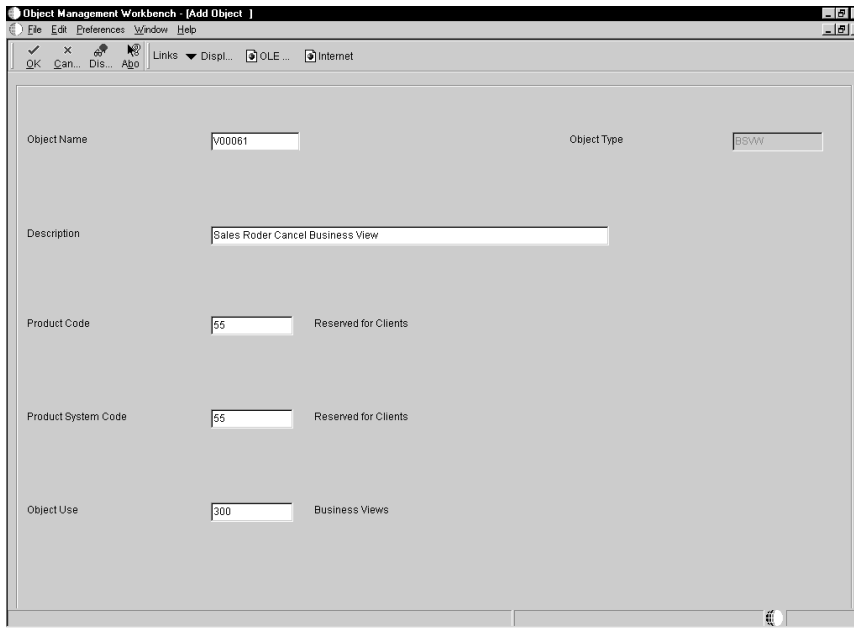
See Also

- *Performance* for performance information about business views



To add a business view

1. On Object Management Workbench, click Add.
2. On Add OneWorld Object to the Project, choose the Business View option, and then click OK.



3. On Add Object, complete the following fields and click OK:

- Object Name

See the J. D. Edwards Naming Standards below.

- Description
- Product Code
- Product System Code
- Object Use

J. D. Edwards Naming Standards

J.D. Edwards recommends you use the following guidelines when you add a business view.

The Object Librarian name for a business view can be a maximum of 10 characters and should be formatted as follows: **VZZZZZZZA**

V = business view

ZZZZZZZZ = should be the characters of the *primary* table

A = letter to designate which view. For example V0101A is the first view over the table F0101, V0101B is the second view over the same table.

External Developer Considerations

External development refers to creation of applications by developers outside of J.D. Edwards, such as Partners in Development and J.D. Edwards consultants that might create custom applications for specific clients. You must use caution when naming a business view to prevent collisions between J.D. Edwards and non-J.D. Edwards objects. When creating a new business view over a standard J.D. Edwards table, format the business view name as follows: **Vssss9999**, where

V = business view

ssss = the system code applicable to the enterprise

9999 = a unique next number or character pattern unique within the enterprise

Provide up to a 60-character description for a business view. It should reflect the application description followed by the form type, such as Item Master Browse and Item Master Revisions.

Primary unique key fields should remain in the business view. Do not reorganize the primary unique key fields.

Note: There should be one business view for each table that includes all columns. Use this business view for the level 01 section in all reports upon which the file is based.

Also, only one business view is allowed for each form type, except in the case of a header/detail form. In this instance, two business views may be selected, one for the header portion of the form and one for the detail portion of the form.

Joined Views

Format the name for joined views using the names of the two tables being joined, separated by a forward slash. Place the primary table first.

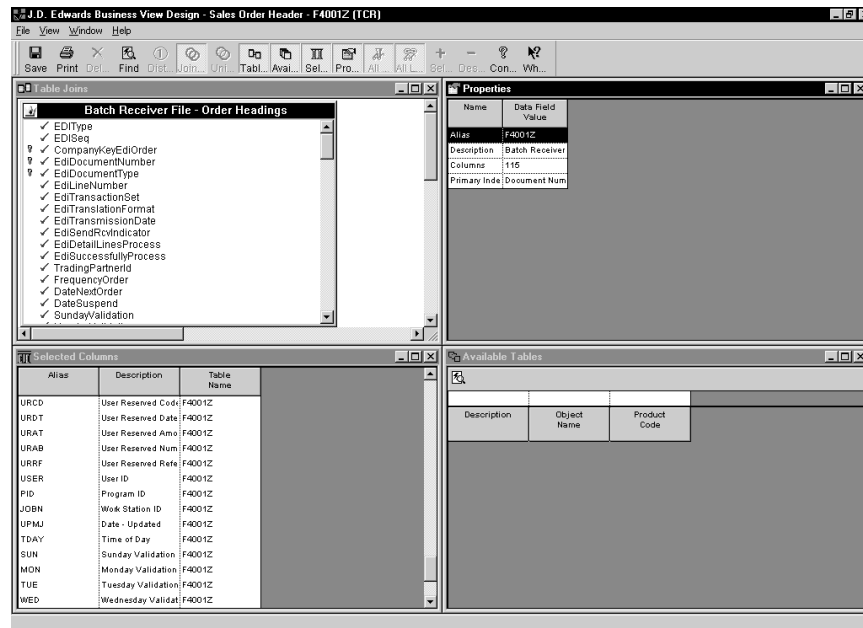
For example, where F4101 is the primary table in the join view between F4101 and F4102, use the following naming standard: F4101/F4102

Field	Explanation
Object Name	<p>The OneWorld architecture is object-based. This means that discrete software objects are the building blocks for all applications, and that developers can reuse the objects in multiple applications. Each object is tracked by the Object Librarian. Examples of OneWorld objects include:</p> <ul style="list-style-type: none">• Batch Applications (such as reports)• Interactive Applications• Business Views• Business Functions• Business Functions Data Structures• Event Rules• Media Object Data Structures
Description	<p>The description of a record in the Software Versions Repository file. The member description is consistent with the base member description.</p>
Product Code	<p>A user defined code (98/SY) that identifies a J.D. Edwards system.</p>
Product System Code	<p>A code that designates the system number for reporting and jargon purposes.</p> <p>See UDC 98/SY.</p>
Object Use	<p>Designates the use of the object. For example, the object may be used to create a program, a master file, or a transaction journal.</p> <p>See UDC 98/FU.</p>

Working with Business View Design

The Business View Design tool presents several views within the same window:

- Table Joins defines the tables over which you create the business view.
- Available Tables locates tables for selecting and moving to the Table Joins view.
- Selected Columns lists the data items from your table that are included in your business view.
- Properties.



You should generate the business view whenever you create a new business view or when you add a data item to the business view.

You should be aware of the following:

- When you delete a data item from a business view, if the database item is used in an application, an error results when you run the application.

If this occurs, you must open the application and delete the item from the application or reselect the column in the current business view to fix the control.

- When you delete an entire table from a business view, any application that uses the business view will not run.

If this occurs, you must fix all items that refer to the business view.

- If you delete a business view any forms that use the business view will fail.

If this occurs, you must connect the forms to a new business view and connect all of the controls.

This topic describes the following tasks:

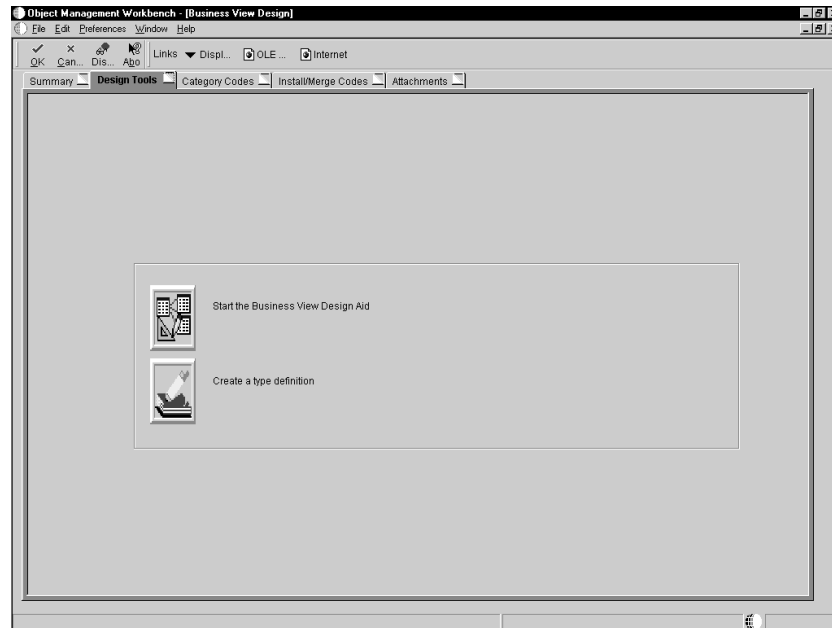
- Launching the Business View Design Aid
- Choosing a table for a business view
- Choosing data items for a business view
- Using Select Distinct
- Creating a table join
- Creating a table union

Launching the Business View Design Aid

You use the Business View Design Aid tool to define a new business view or to modify an existing one.

To launch the Business View Design Aid

1. On Object Management Workbench, create a new business view or choose an existing business view, and then click the Design button in the center column.
2. Click the Design Tools tab, and then click Start the Business View Design Aid.



Choosing a Table for a Business View

Choose one or more tables from which to create the business view.

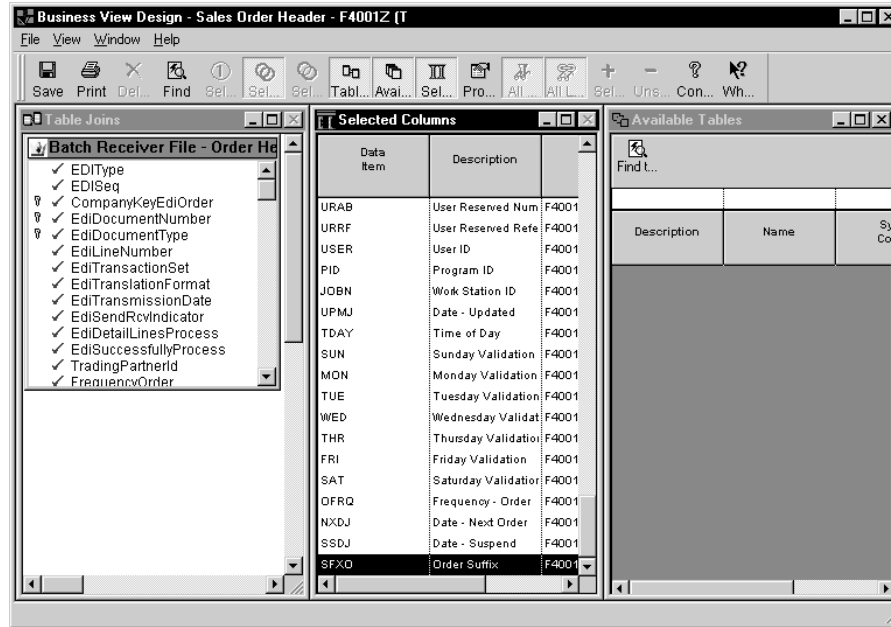
You can create a business view over one large table to retrieve and update only those columns needed by the application. However, performance is diminished when retrieving and updating a very large table. Consider joining two smaller tables rather than creating one large table that contains many data items.

► To choose a table for a business view

1. On the Available Tables view of Business View Design, locate the table for the business view using the query-by-example line. You can search by Description, Name, and System Code.
2. Choose one or more tables and drag them to the Table Joins view. The view is called Table Join regardless of whether you are joining multiple tables or not.

The Table Joins view displays selected tables, along with the columns that comprise each. Key symbols appear next to columns that are index keys in the table. The primary table supplies the key to the business view. This is where an application begins a search.

Note: To ensure maximum performance in your application, OneWorld limits the number of tables to five if all joins are simple joins or to three tables if any of the joins is an outer join or if there is a union.



3. Designate a primary table by double-clicking the title bar of the desired table (optional).

If the business view contains multiple tables, the first table added is automatically designated as the primary table. A crown symbol appears in the upper left corner of the window to indicate the primary table. If a business view contains only one table, that table is automatically the primary table.

4. To delete a table from a business view, choose it and choose Delete from the Table menu, or click the right mouse button and choose the Delete menu item.

Choosing Data Items for a Business View

After you choose one or more tables for the business view and indicate the primary table, you must choose the data items to include in the business view. All data items in the primary table, along with any tables to which the primary table is joined, are available for the business view.

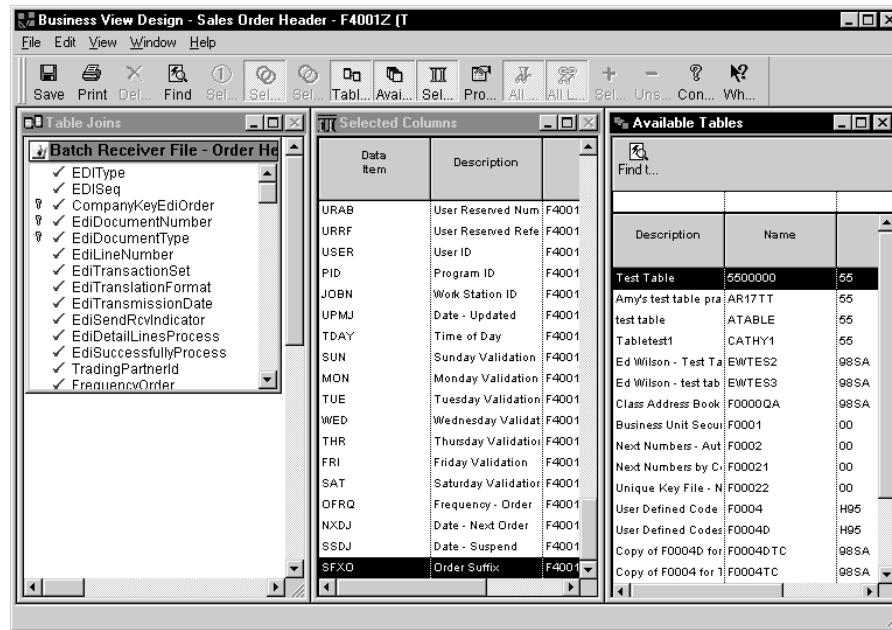
Choose the data items that you want to use in an interactive application or batch job. When you create an application, you do not have to use every item in the business view.

► To choose data items for a business view

1. On the Table Joins view of Business View Design, double-click the data items to include in the business view.

Selected data items appear highlighted in the Table Joins window. As you select each item, it is added to the Selected Columns window.

Do not select more than 256 columns for your business view. An application that uses a business view containing more than 256 columns will fail at runtime.



2. Double-click a selected item in the Table Joins window to remove it from the business view.

If you have two tables in a join, the primary index is automatically selected for both. Index keys of selected tables are automatically highlighted for use in the business view. Except for the primary table index keys, you can remove index keys if they are not going to be used in the business view. You cannot remove index keys for the primary table from the business view.

If you do a join, items from both tables are automatically selected. If the same data item appears in multiple tables, you only need to select the data item from one table.

You cannot join unlike items.

Do not select the same data item from two different tables; otherwise it will appear twice in the business view.

Using Select Distinct

If a business view includes the primary key of the primary table (default implementation), then every row of the business view query will be unique. The primary key will have a different value in each row of the primary table. If the business view does not have a primary key in the primary table, then duplicate rows can occur during the business view query. You can eliminate duplicate rows in the business view query by using the Select Distinct feature while designing a business view.

For example, Journal Entry is unique by line number within a document. You only need the first document number with Line 1, not all line numbers within the document. Select Distinct will fetch only the first occurrence of the document number, not all the line numbers within it.

Any business view with a primary table containing one or more of the following columns, which are used for currency support or security features, may cause the Select Distinct feature to output duplicate values:

CO	Company
CRCO	Currency Code – From
CRDC	Currency Code – To
CRCX	Currency Code – Denominated In
CRCA	Currency Code – A/B Amounts
LT	Ledger Type
AID	Account ID
MCU	Business Unit
KCOO	Order Company (Company Code)
EMCU	Business Unit Header
MMCU	Branch
AN8	Address Number

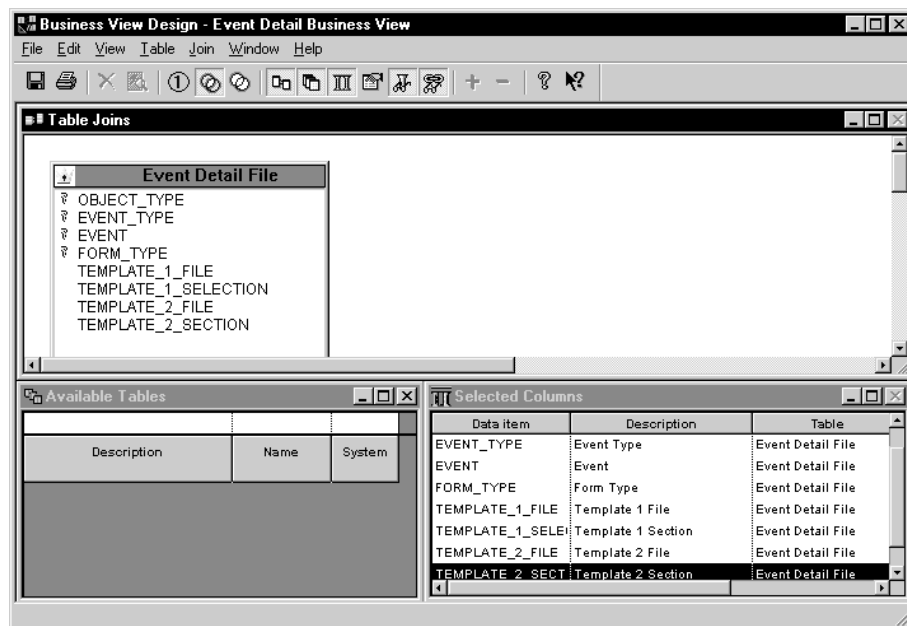
► To use select distinct

1. On Business View Design Aid, choose Distinct from the File menu.
2. Choose the primary table of your view.
3. From the Table menu, choose Change Index to change the index of the primary table to a nonunique index.

Example: Select Distinct

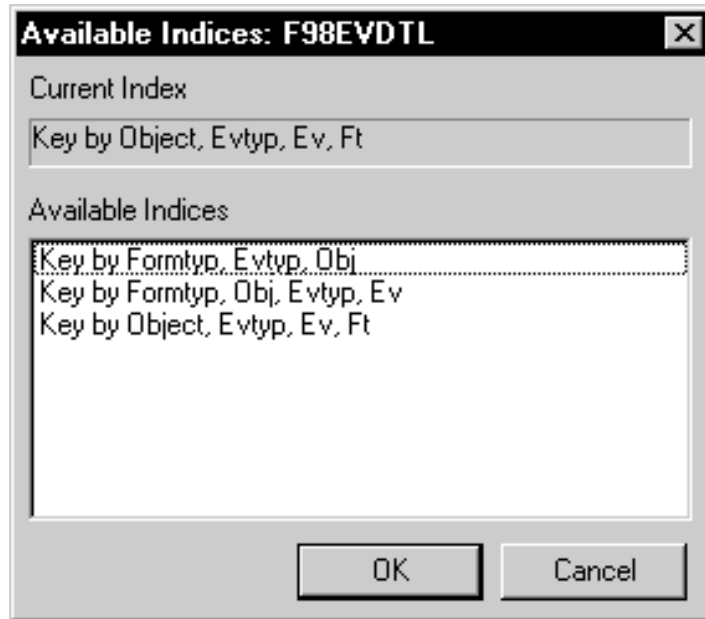
This example illustrates Select Distinct. The business view used for this example, V98EVDTL, uses the primary index of the primary table (in this example it is F98EVDTL) by default. The primary index of a OneWorld table must be unique. Because of this, duplicate values are not returned when the business view query is generated. Business View Design Aid also allows you to use any other index of the primary table when processing the business view.

Choose the primary table in Business View Design Aid. From the Table menu, choose Change Index to change the primary table index. Change Index is only available for the primary table.

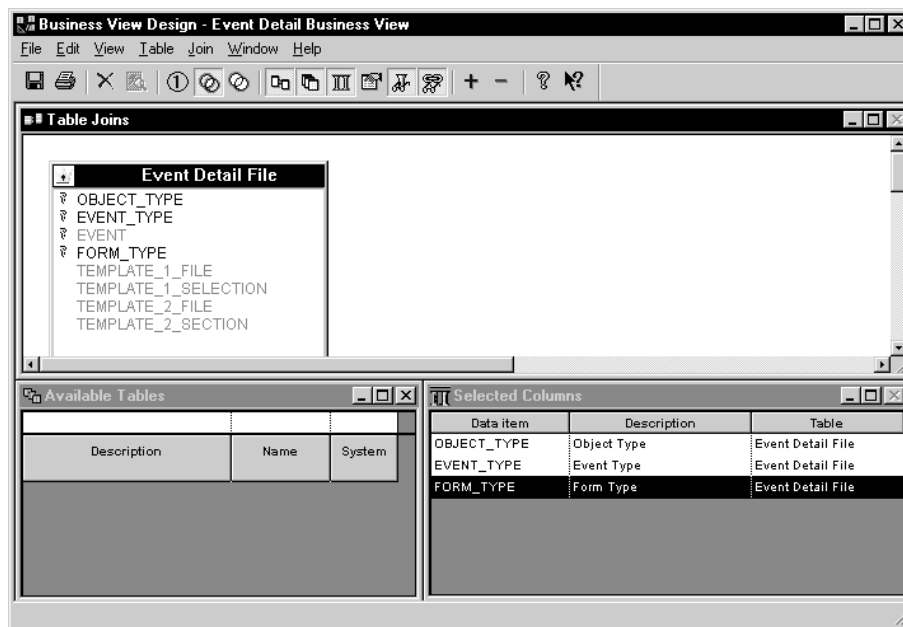


A warning, indicating that your selected column list will be changed, appears. Click Yes to continue.

The Available Indices form appears. The first edit field on the form displays the current index of the table used by the business view. The default is the primary index.



For this example, choose Key by Formtyp, Evtype, Obj from Available Indices and click OK.



The Table Joins list and Selected Columns list change to reflect the keys of the new index.

Save the changes and exit Business View Design Aid.

Now if you run an application that uses the V98EVDTL business view, with Select Distinct off, and the changed business view index, Key by Formtyp, Evtyp, Obj, the generated SQL statement is:

```
SELECT EDOBJTYPE, EDEVTYPE, EDFORMTYPE FROM PVC. F98EVDTL
```

Using this example, you might now have 281 rows of data from table F98EVDTL.

Next you reopen the V98EVDTL business view. Choose Select Distinct from the File menu. Choose Change Index to reselect the Key by Formtyp, Evtyp, Obj index from Available Indices and click OK. Save the business view and exit Business View Design Aid.

At this point you may need to exit and restart OneWorld. OneWorld caches the business view in memory, so even though you have changed the business view, the previous cached business view will run until it is cleared out.

When you generate and rerun the same application using the V98EVDTL business view you just completed (with Select Distinct on and the changed business view index Key by Formtyp, Evtyp, Obj), the generated SQL statement is:

```
SELECT DISTINCT EDOBJTYPE, EDEVTYPE, EDFORMTYPE FROM PVC.
F98EVDTL
```

Using this example, you might now have 53 rows of data from table F98EVDTL.

Creating a Table Join

Create table joins to access multiple tables in a single application.

Joins are typically used for non-input-capable forms, such as Find/Browse forms, and reports. They are not usually used for forms that update and add to the database because the relationship between the records must be precise. Use joins for input capable forms when the relationship between two tables is simple. Joins are faster than using individual fetches to retrieve data.

If a business view uses multiple tables, they must be linked by establishing joins between columns in those tables. The links instruct the system on how rows between a table correspond to rows in another table.

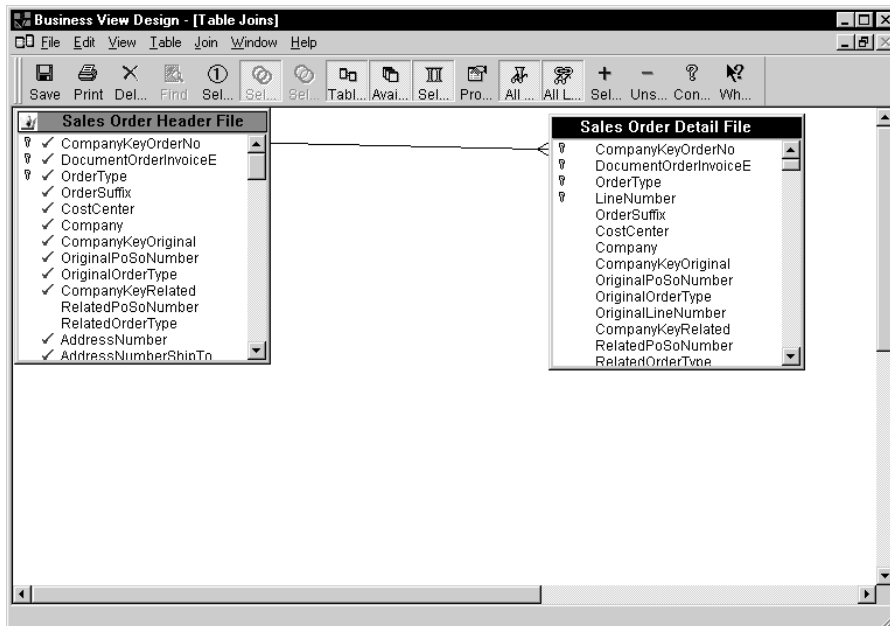
Data item attributes are defined in the data dictionary. Use the Properties window to view attributes for a column when determining whether you can use it in a join. The highlighted data item in the Selected Columns window displays the properties of that data item in the Properties window.

Look at each table and decide how the data in each table is related to the data in other tables for your application or report. You might need to add columns, build indexes, or even create new tables. If you build new indexes be sure your needs have been well thought out before you do so.

► To create a table join

1. Click and draw a line that links a column in the primary table to a column in a related table.

You can create table joins only between like columns. The name of the columns can be different, but the attributes for Data Type and Decimals must be identical. To determine whether data items are candidates for a join, view the data item attributes that are displayed in the Hover Hints for each data item.



2. To delete a join, choose it and choose Delete from the Join menu, or click the right mouse button and choose Delete from the pop-up menu.
3. From the Join menu, choose Types and choose one of the following join types (the default is Simple):
 - Simple
 - Left Outer
 - Right Outer
4. From the Join menu, choose Operators and choose one of the following operators (the default is Equal)
 - Equal (=)
 - Less than (<)
 - Greater than (>)
 - Less or equal (<=)
 - Greater or equal (>=)

- Click the Selected Columns view to sequence columns.

Creating a Table Union

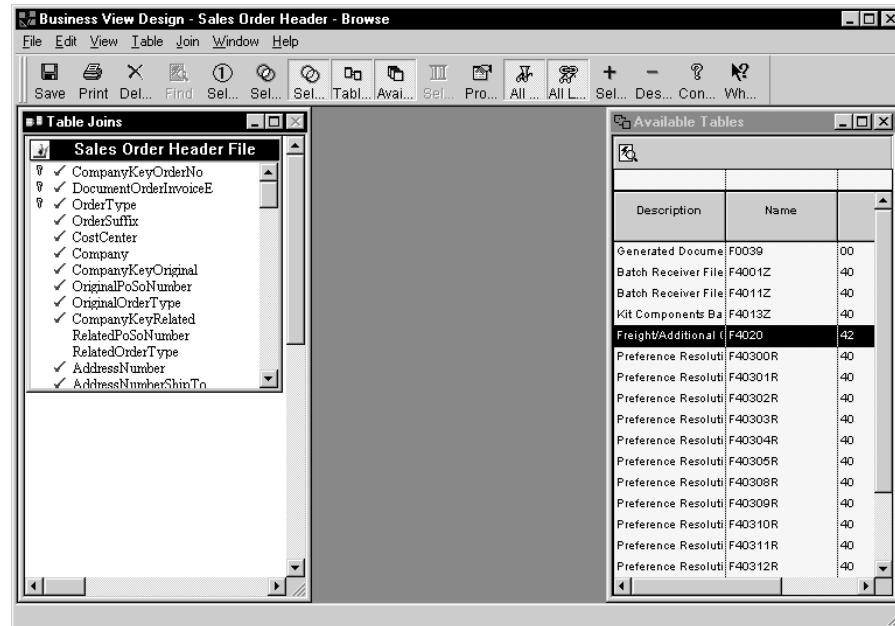
Unions are used to pull rows from tables that have the same structure. Unions pull rows that exist in either table.

► To create a table union

- On Business View Design, from the Table menu, select Unions (or use the appropriate icon on the toolbar).

The view will still say Table Joins.

- Choose the tables you wish to unite.





Data Structures

Data structures are a key element of any programming language or environment. A data structure is a list of parameters that is used to pass data between applications and tables or forms. OneWorld uses data structures in the following instances:

- The system generates a data structure.
- You create a data structure.

System Generated Data Structures

Form Data Structures

Each form with an attached business view has a default data structure. Maintain the data structure using the Form/Data Structure menu option in Forms Design. This data structure is used to receive parameters from or send parameters to other forms during Form Interconnects.

Report Data Structures

A batch application with an attached business view can receive parameters from or send parameters to a data structure. Create and maintain the data structure from the Report/Data Structure menu option in Report Design. Unlike a form data structure, this data structure is not automatically populated with data items.

User Generated Data Structures

You can create three types of data structures.

Media Object Data Structures

To enable an application for media objects, you must create a data structure to pass arguments from the application table to the media object table. To work with a data structure for media objects, create a GT object type, or select an existing one to modify in Object Librarian.



Processing Options Data Structures

Processing options are used to create an “input property sheet.” A parameter list is used to pass processing options to an application. You can create a processing option data structure template or modify an existing one in Object Librarian.

Business Function Data Structures

Any business function, whether it uses C or Business Function Event Rules as its source language, must have a defined data structure to send or receive parameters to or from applications. You can create a DSTR object type, or select an existing one to work with in Object Librarian.

You can also create data structures for text substitution messages (refer to the Messaging section in this guide).

You can attach notes, such as an explanation of use, to any data structure or data item within the structure.

This section describes the following topics:

- Working with interconnection data structures
- Creating a data structure

Working with Interconnection Data Structures

Interconnection data structures are maintained by the Form Design or Report Design tool, not by Object Librarian. The data structure enables an application or report to pass values between interconnection forms or sections.

For interactive applications, the default data structure contains only the keys from the business view selected for a form. If you want to pass a value for a data item that is not in the default data structure, you must add the data item to the data structure.

A default data structure is created at the report level for a report with an attached business view. The default data structure is empty. If data items are added to the data structure, the structure is maintained by Report Design, as is the report.

This topic describes the following:

- Changing form data structures
- Changing report data structures
- Displaying type definitions

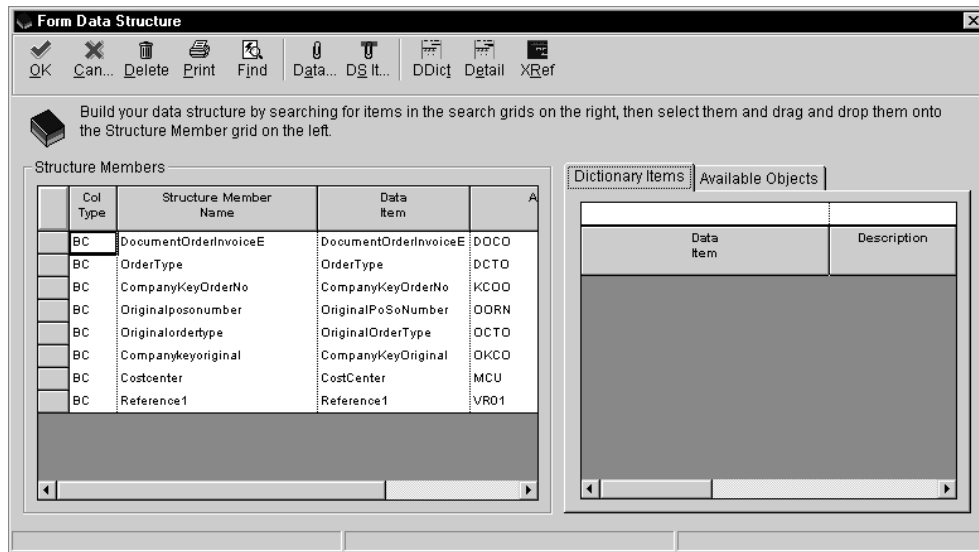
Changing Form Data Structures

You create the default data structure by using keys from the business view selected for the form. You can access the form data structure from Forms Design. From the Form menu, choose Data Structure. The Form Data Structures form is displayed.

The items in the existing data structure are displayed in the Structure Members list. Two tabs appear for modifying the data structure. The Available Objects Tab lists data items in the attached business view. If you want to add an item to the data structure that is not in the Available Objects list, then you can select it from the Dictionary Items list. The Dictionary Items tab accesses items from the Data Dictionary.

To change form data structures

1. On Form Design Aid, focus on the form with the data structure you want to modify, and then select Data Structures from the Form menu.



2. To add an object from the attached business view or grid column, click the Available Objects tab, then drag the object into Structure Members on the left of the form.
3. To add a specific data dictionary item, perform these steps:
 - First, click the Dictionary Items tab.
 - Second, enter data to search for a data dictionary item in the QBE row and then click find.
 - Third, drag the desired data dictionary item into Structure Members on the left of the form.
4. To remove an object from the data structure, select the item in Structure Members and then click Delete.
5. When finished, click OK.

Changing Report Data Structures

An empty default data structure is created for any report section with an attached business view. This data structure is used to receive values from another report in a report interconnection. If a required item is not in the existing data structure, you can select it from the data dictionary and include it in the Structure Members list. To modify a report data structure use Report Design and follow the same procedures as those for modifying a form data structure.

Example: Changing Interconnection Data Structures

The following example describes a situation in which you might want to change an interconnect data structure. Suppose you create two Fix/Inspect forms that use the same business view. The second form shows more detail, for example, additional columns from the same record. If you do not want to refetch the record in the second form, you should change the data structure for the second form.

The data structure for the second form should contain all the information (fields on the form)/data items in the data structure, in addition to the keys in the business view. In this case you should set the Form Options to No fetch on form business view for that form. This also applies if you are not updating the record with new information on the second form, but you are passing that information back to the first form and letting the first form update the database. In this case, you should set the form options for updating the database to No Update on Form Business View.



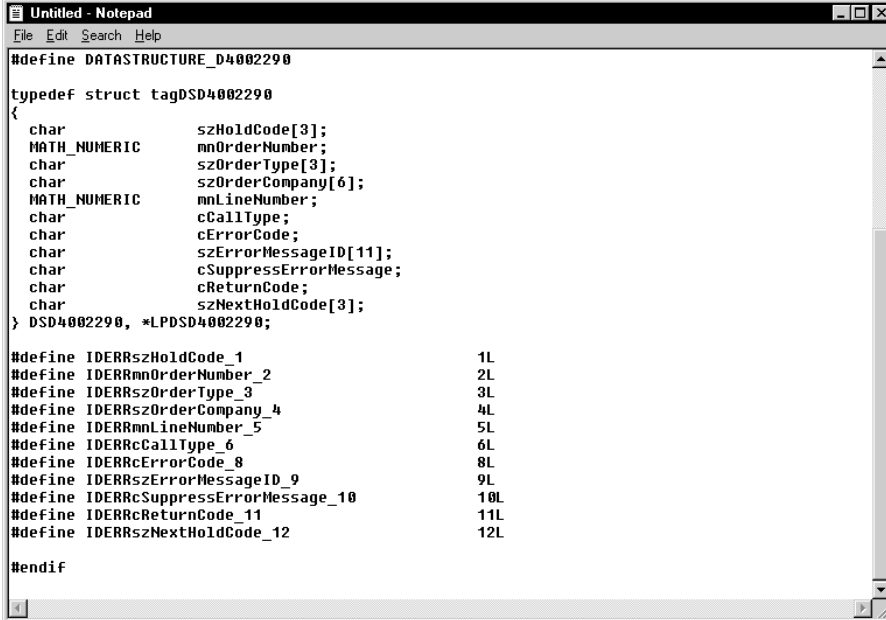
Depending on your configuration, you might want to fetch the entire record once in the first form and pass it to the second form, or let the second form refetch the record. In this case, you will have two separate business views, with different columns from the same record. If the second form is a form that will not be accessed often, do not fetch the fields in the first form's business view. Use a business view for the first form that has only the columns needed for the first form. If you fetch all columns needed for both forms you do only one Select from the database and all the data is passed over the network once, as opposed to having two separate Select statements generated by two different business views traversing the network. If hardware or memory for the workstation is not a problem, then fetch all columns for that record for both forms and let that information reside in the workstation's memory.

Displaying Type Definitions

You can display the typedef for a data structure.

► To display typedef

The type definition is stored in the clipboard. You can paste it into another application, such a word processing application, to save or display the definition.



```
Untitled - Notepad
File Edit Search Help
#define DATASTRUCTURE_D4002290

typedef struct tagDSD4002290
{
    char          szHoldCode[3];
    MATH_NUMERIC  mnOrderNumber;
    char          szOrderType[3];
    char          szOrderCompany[6];
    MATH_NUMERIC  mnLineNumber;
    char          cCallType;
    char          cErrorCode;
    char          szErrorMessageID[11];
    char          cSuppressErrorMessage;
    char          cReturnCode;
    char          szNextHoldCode[3];
} DSD4002290, *LPDSD4002290;

#define IDERRszHoldCode_1          1L
#define IDERRmnOrderNumber_2      2L
#define IDERRszOrderType_3        3L
#define IDERRszOrderCompany_4     4L
#define IDERRmnLineNumber_5       5L
#define IDERRcCallType_6          6L
#define IDERRcErrorCode_8         8L
#define IDERRszErrorMessageID_9   9L
#define IDERRcSuppressErrorMessage_10 10L
#define IDERRcReturnCode_11      11L
#define IDERRszNextHoldCode_12   12L

#endif
```

Creating a Data Structure

A data structure object is required to create a business function. A data structure is used to pass data between an application and a business function. OneWorld provides the ability to create an encapsulated data structure object.

You can create several types of data structures that are maintained independently as objects in Object Librarian. You can:

- Creating a media object data structure
- Creating a processing options data structure
- Creating a business function data structure

Refer to the *Development Standards: Application Design Guide* for J.D. Edwards naming conventions.

Creating a Media Object Data Structure

A special data structure object is required to work with media objects. The F00165 table contains information used to determine how to access specific media object attachments. This information is stored in the Object Name and Media Object Key (GDTXKY) fields. For example, in the Sales Order application, GDTXKY stores the key for each sales order entered. The F00165 table contains the following:

- Object Name (for example GT4201A)
- GDTXKY (Key) (for example, sales order number | order type | company)
- GDTXVC (Media Object Attachment)

The media object key contains the actual key to where a specific record in the application is stored. This typically is the same as the keys to the table used by the grid or form. The media object key must have a unique value for each media object attachment so that the system will know which attachment to retrieve.



To create a media object data structure

1. On Object Management Workbench, click Add.

2. On Add OneWorld Object to the Project, choose the Media Object Data Structure option, and then click OK.

The Add Object form appears.

3. On Add Object, complete the following fields and click OK:

- Object Name

The J.D. Edwards standard format for naming a media object data structure is GTxxxxyyA.

GT = media object

xxxx = the file name excluding the letter F

yy = a next number

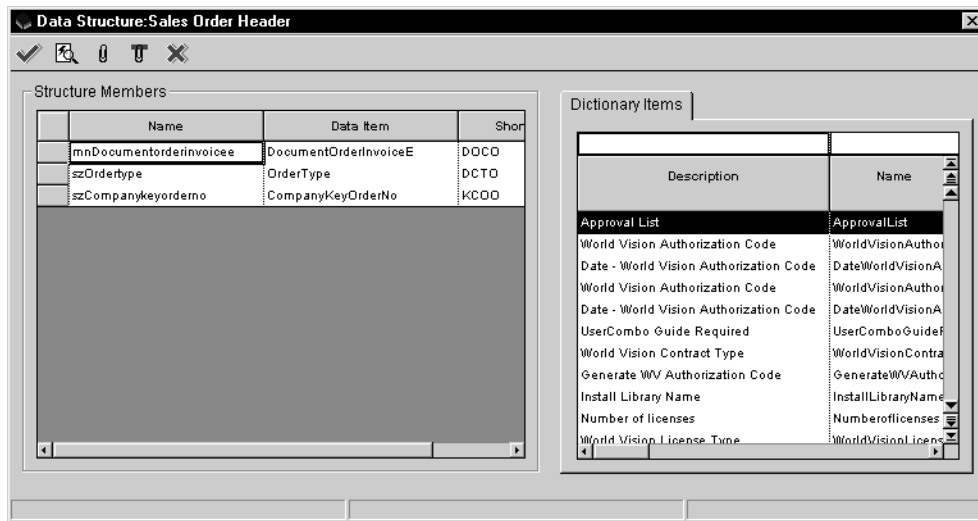
A = If a file has multiple media objects, differentiate them by placing a letter (such as A, B, or C) at the end of the name.

- Description

Provide up to a 60-character description that reflects the subject of the media object.

- Product Code
- Product System Code
- Object Use

4. On Data Structure Design, click the Design Tools tab, and then click Data Structure Design.



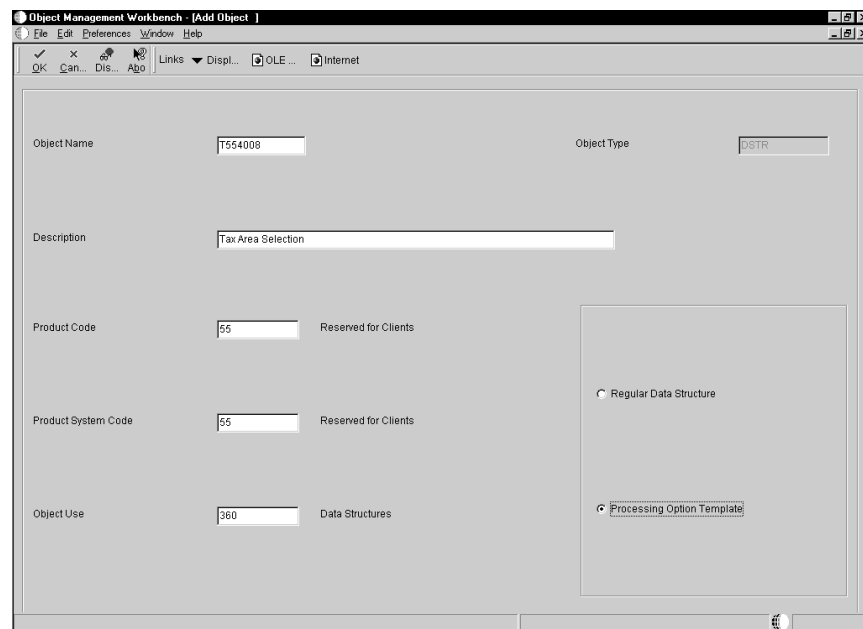
5. Choose data items from Dictionary Items list.
 - Enter a Name, Alias, System Code or combination in the QBE fields and click Find to more easily locate an item in the list.
6. To change the name double-click on the name and type a new name. Use Hungarian notation rules.
7. Drag the desired data item to Structure Members.
8. Click OK.

Creating a Processing Options Data Structure

Processing options are used to create an input property sheet.

▶ To create a processing options data structure

1. On Object Management Workbench, click Add.
2. On Add OneWorld Object to the Project, choose the Data Structure option, and then click OK.



3. On Add Object, complete the following fields:
 - Object Name

The J.D. Edwards naming convention for processing option data structures is Txxxxyyy where:

- T = processing option data structure

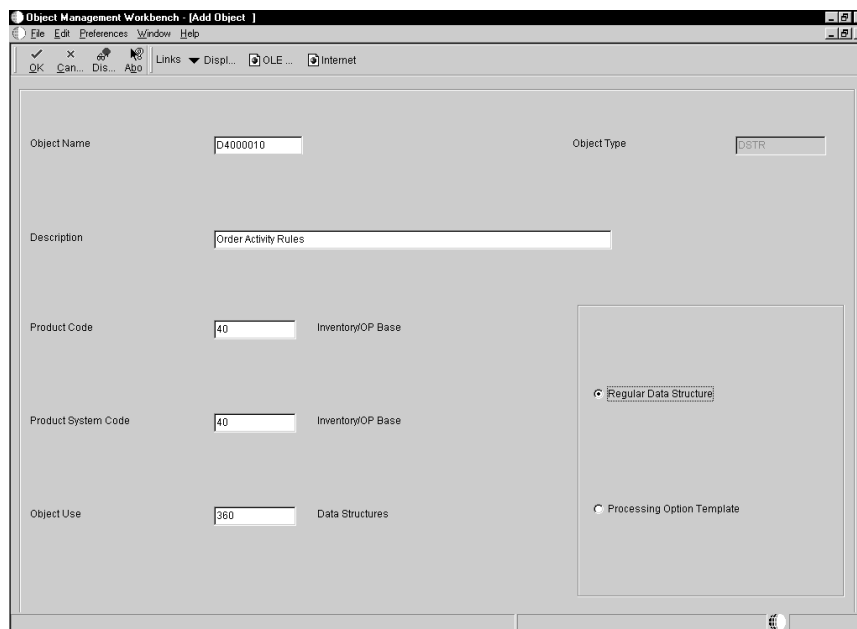
- xxxxyyyy = the program number for the application or report
 - Description
 - Product Code
 - Product System Code
 - Object Use
4. Choose the Processing Option Template option, and then click OK.

Creating a Business Function Data Structure

Business function data structures are used by C business functions and business function event rules to pass data between the business functions or application event rules.

► To create a business function data structure

1. On Object Management Workbench, click Add.
2. On Add OneWorld Object to the Project, choose the Data Structure option, and then click OK.



3. On Add Object, complete the following fields.
 - Object Name

The J.D. Edwards naming convention for processing option data structures is DxxxxxyyyyA where:

- D = data structure
 - xxxx = the system code
 - yyyy = a next number
 - A = a multiple data structure differentiator (A, B, C, and so forth), used when a function has more than one data structure
- Description
 - Product Code
 - Product System Code
 - Object Use
4. Choose the Regular Data Structure option, and then click OK.

Defining a Data Structure

Once created, you must define what data objects to include in the data structure. You can modify existing data structures by adding new data objects to it or deleting data objects from it.

Defining a data structure includes:

- Launching data structure design
- Designing a data structure
- Creating a type definition

See Also

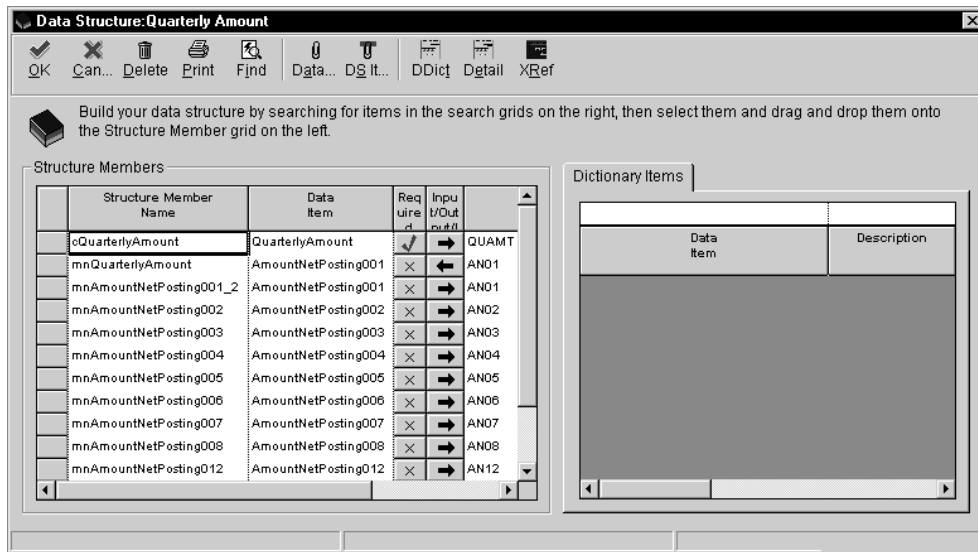
- See *Processing Options* for information about working with processing options and their data structures and templates.

Lanching Data Structure Design

If you have just created a new data structure and are viewing the Object Librarian Data Structure Design form, skip to step 3.

▶ **To launching data structure design**

1. On Object Management Workbench, check out the data structure you want to work with.
2. Select the data structure, and then click the design button in the center column.
3. On Data Structure Design, click the Design Tools tab, and then click Data Structure Design.



Designing a Data Structure

After creating a data structure with the OMW, use the following process to define how it will function.

► To design a data structure

1. To add a data dictionary item to the data structure perform the following steps:
 - First, enter data to search for a data dictionary item in the QBE row and then click find.
 - Second drag the desired data dictionary item into Structure Members on the left of the form.
 - To change the name, double-click the name and enter a new name. Use Hungarian notation rules.

If you are unsure about which data dictionary item you want, click Data Dictionary to search for data dictionary items and then to view the data items you select in detail.

You can view detailed information about a data item in the data structure by selecting it and then clicking Data Dictionary Detail.

2. You can indicate the required status by leaving the required field blank or choosing one of the following:
 - An X to indicate the field is optional.

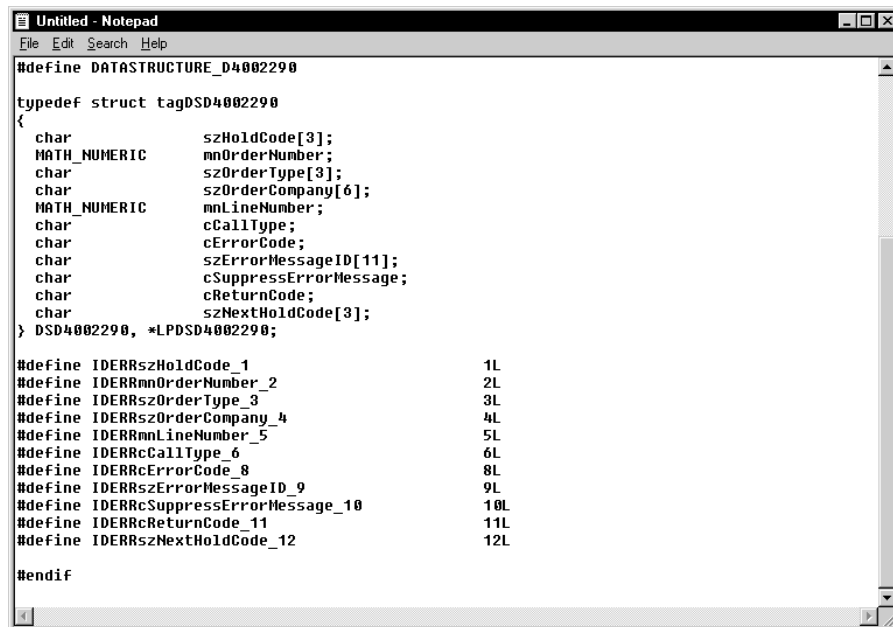
- A checkmark to indicate the field is required.
3. Choose a direction for the data flow by clicking on the arrow until it changes to the appropriate direction and type.
 4. To remove an object from the data structure, select the item in Structure Members and then click Delete.
 5. To define attachments for the data structure, click Data Structure Attachments.
 6. To define attachments for a data structure item, click Data Structure Item Attachments.
 7. To launch the Cross Reference utility, click Cross Reference.
 8. When finished, click OK.

Creating a Type Definition

The type definition is stored in the clipboard. You can paste it into another application, such a word processing application, to save or display the definition.

► To create a type definition

1. On Object Librarian Data Structure Design, click the Design Tools tab.
2. Click *Create a type definition*.



```

Untitled - Notepad
File Edit Search Help
#define DATASTRUCTURE_D4002290

typedef struct tagDSD4002290
{
    char          szHoldCode[3];
    MATH_NUMERIC  mnOrderNumber;
    char          szOrderType[3];
    char          szOrderCompany[6];
    MATH_NUMERIC  mnLineNumber;
    char          cCallType;
    char          cErrorCode;
    char          szErrorMessageID[11];
    char          cSuppressErrorMessage;
    char          cReturnCode;
    char          szNextHoldCode[3];
} DSD4002290, *LPDSD4002290;

#define IDERRszHoldCode_1           1L
#define IDERRmnOrderNumber_2       2L
#define IDERRszOrderType_3         3L
#define IDERRszOrderCompany_4      4L
#define IDERRmnLineNumber_5        5L
#define IDERRcCallType_6           6L
#define IDERRcErrorCode_8          8L
#define IDERRszErrorMessageID_9    9L
#define IDERRcSuppressErrorMessage 10L
#define IDERRcReturnCode_11        11L
#define IDERRszNextHoldCode_12     12L

#endif

```




Cross Reference Facility

You can use the Cross Reference Facility to find out information about where specific kinds of objects are used and how they are used. You can also view relationships between objects and their components. For example, you can:

- Identify each instance where a business function is used
- View a list of forms in an application
- Display all fields within a business view or form interconnection
- Cross-reference all applications where a specific field, event rule, or control is used

You can rebuild cross reference relationships if you change something.

This section contains the following:

- Working with the Cross Reference facility



Working with the Cross Reference Facility

The Cross Reference Facility (P980011), is located on the Cross Application Development Tools menu.

Use Cross Reference for:

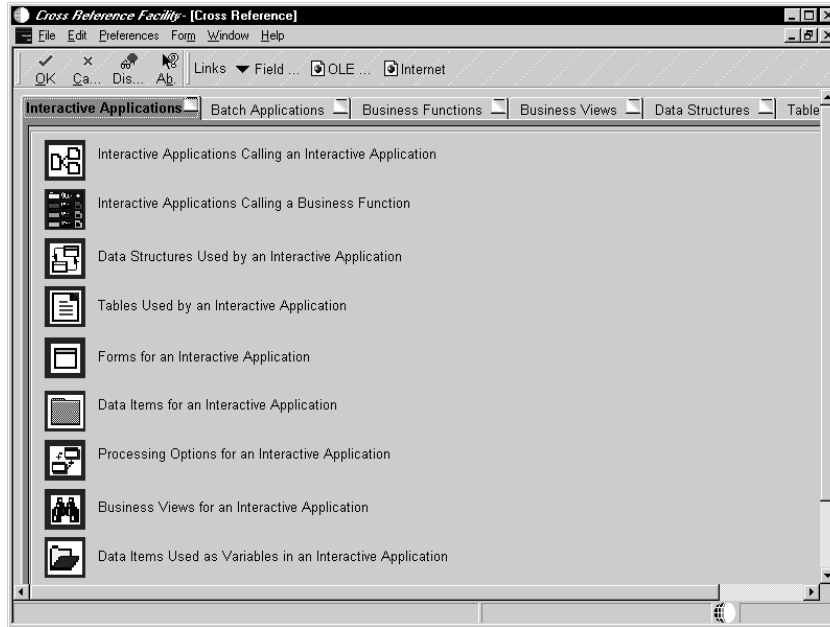
- Searching for objects
- Searching for data items
- Searching for interactive applications
- Searching for batch applications
- Searching for business functions
- Searching for business views
- Searching for data structures
- Searching for tables
- Searching for forms
- Searching for event rules
- Viewing field relationships
- Rebuilding cross reference information

Searching for Objects

You can search for objects by search type and object name.

To search for objects

1. From the Cross Application Development Tools menu (GH902), choose Cross Reference.



Click one of the following tabs:

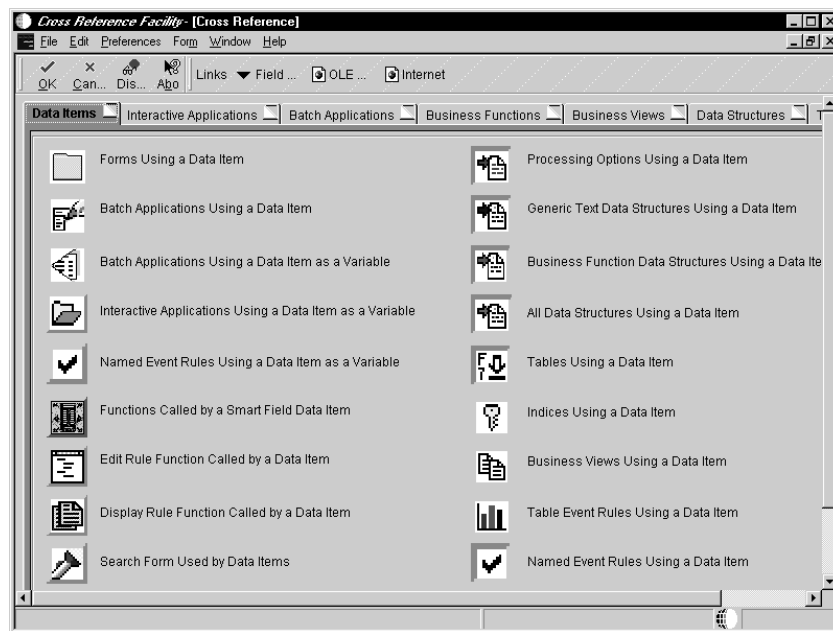
- Data Items
- Interactive Applications
- Batch Applications (the batch application object type in Cross Reference is UBE)
- Business Functions
- Business Views
- Data Structures
- Tables
- Forms

Searching for Data Items

You can locate where data items are used. You can search for the following:

- Forms using a data item
- UBEs using a data item
- UBEs event rules using a data item as a variable
- Applications using a data item as a variable
- Named event rules using a data item as a variable
- Functions called by smart field data items
- Edit rule functions called by a data item

- Display rule functions called by a data item
- Search forms used by data items
- Processing options using a data item
- Generic text data structures using a data item
- Business function data structures using a data item
- All data structures using a data item
- Tables using a data item
- Indices using a data item
- Business views using a data item
- Table event rules using a data item

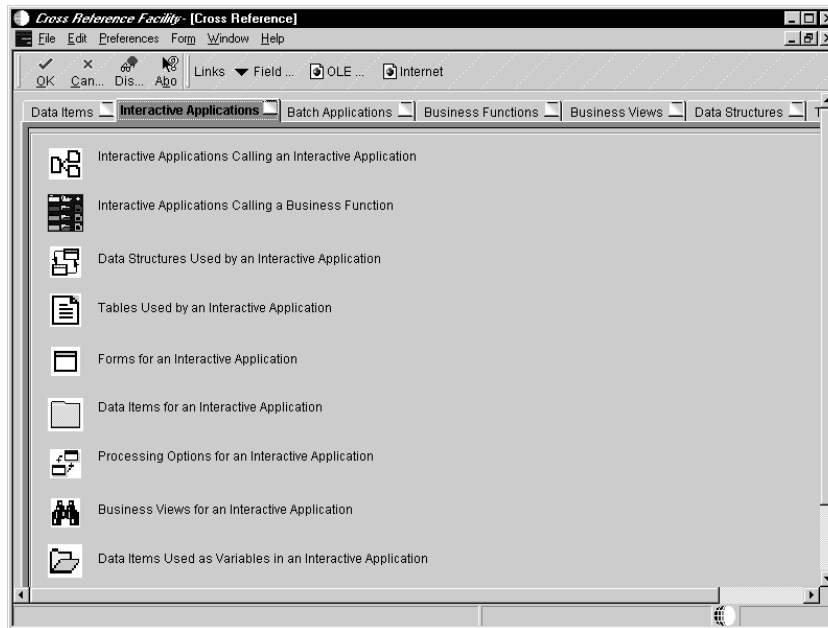


Searching for Interactive Applications

You can locate information about interactive applications. You can search for the following:

- Applications calling an application
- Applications calling a business function
- Data structures used by an application
- Tables used by an application
- Forms for an application
- Data items for an application

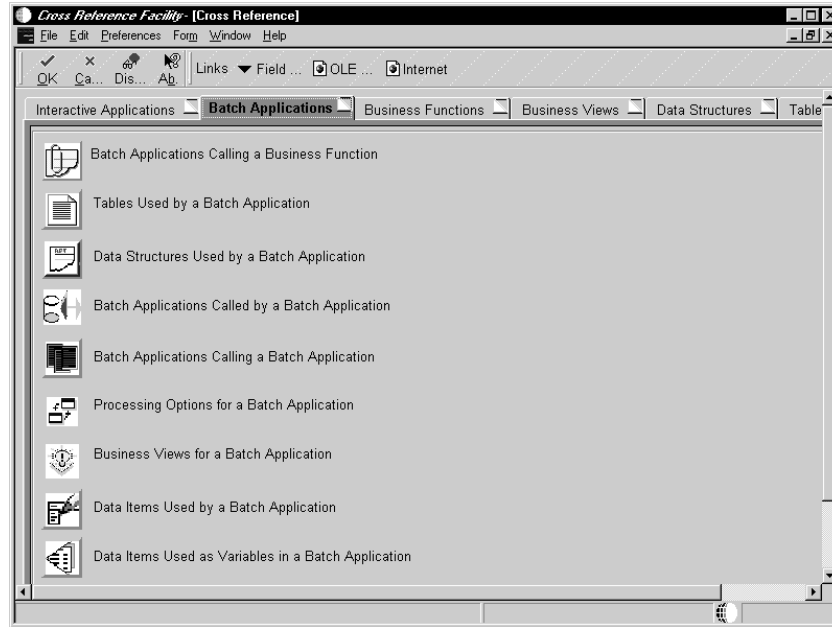
- Processing options for an application
- Business views for an application
- Data items used as a variable in an application



Searching for Batch Applications

You can locate information about batch applications and how they are used. You can search for the following:

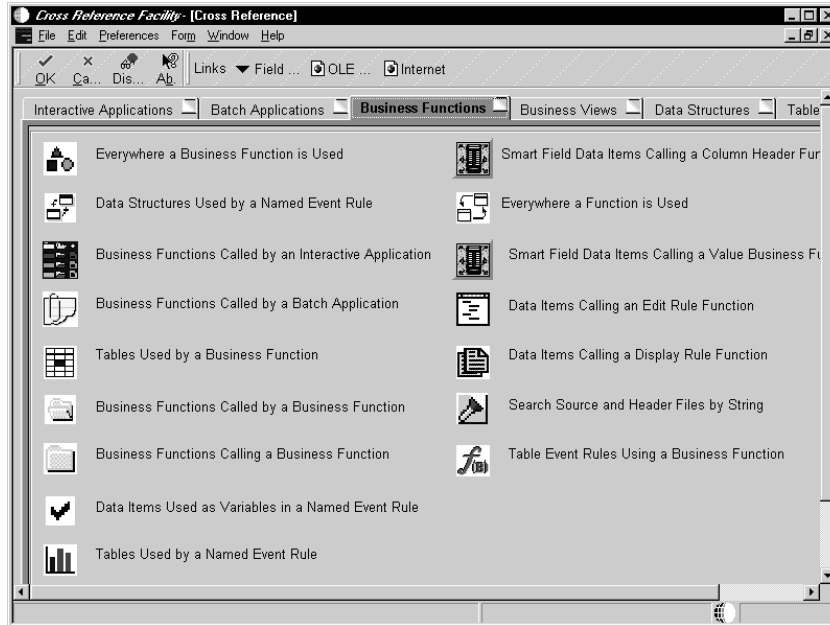
- UBEs calling a business function
- Tables used by a UBE
- Data structures used by a UBE
- UBEs called by a UBE
- UBEs calling a UBE
- Processing options for a UBE
- Business views for a UBE
- Data items used by a UBE
- Data items used as variables in a UBE



Searching for Business Functions

You can locate information about business functions and how they are used. You can search for the following:

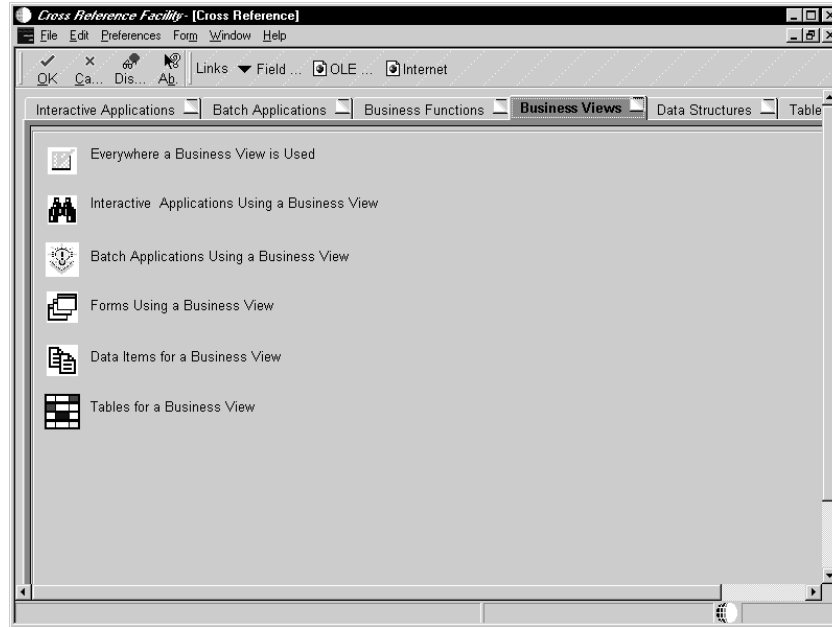
- Everywhere a business function is used
- Data structures used by a named event rule
- Business functions called by an application
- Business functions called by a UBE
- Tables used by a business function
- Business functions called by a business function
- Business functions calling a business function
- Data items used as variables in a named event rule
- Tables used by a named event rule
- Business functions called by a named event rule
- Smart field data items calling a column header function
- Everywhere a function is used
- Smart field data items calling a value business function
- Data items calling edit rule functions
- Data items calling display rule functions
- Search source and header files by string
- Table event rules using a business function



Searching for Business Views

You can locate information about business views and how they are used. You can search for the following:

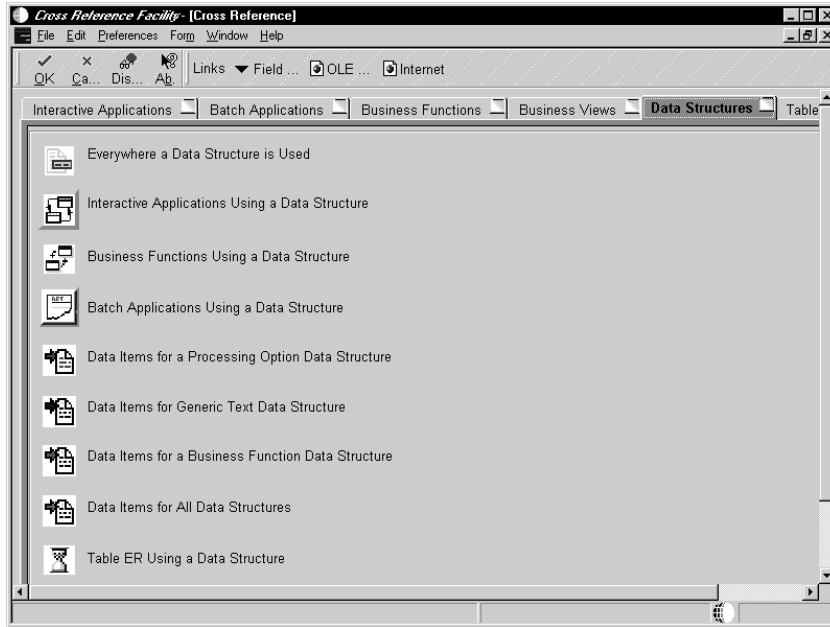
- Everywhere a business view is used
- Applications using a business view
- UBEs using a business view
- Forms using a business view
- Data items for a business view
- Tables for a business view



Searching for Data Structures

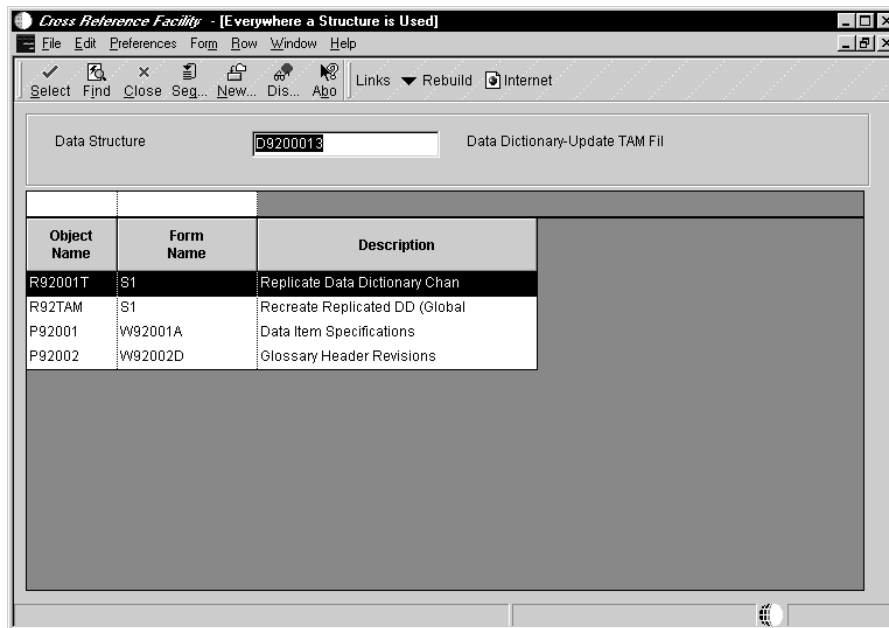
You can locate information about data structures and how they are used. You can search for the following:

- Everywhere a data structure is used
- Applications using a data structure
- Business functions using a data structure
- UBEs using a data structure
- Data items for a processing option
- Data items for generic text data structures
- Data items for a business function data structure
- Data items for all data structures
- Table event rules using a data structure



Everywhere a Data Structure is Used

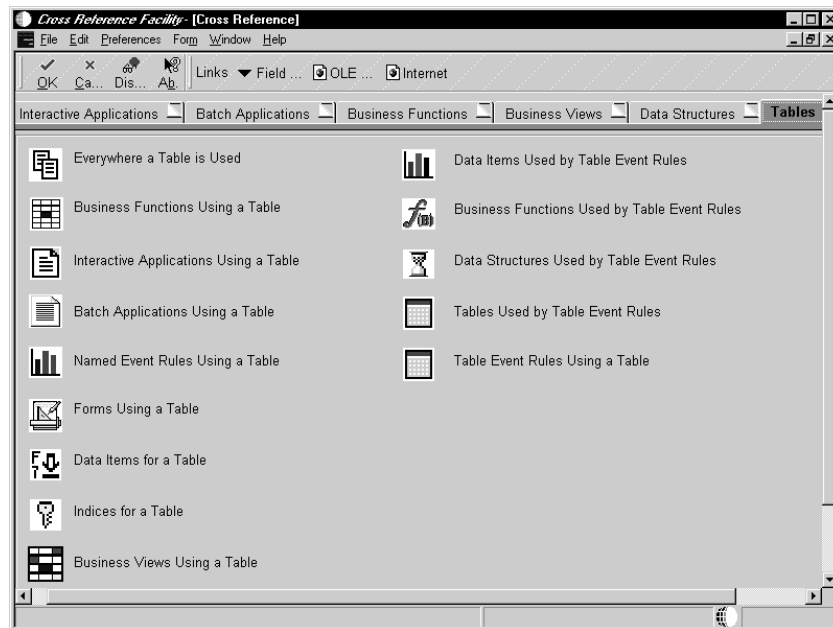
You can determine all the places that a specific data structure is used.



Searching for Tables

You can locate information about tables and how they are used. You can search for the following:

- Everywhere a table is used
- Business functions using a table
- Applications using a table
- UBEs using a table
- Named event rules using a table
- Forms using a table
- Data items for a table
- Indices for a table
- Business views using a table
- Data items used by table event rules
- Business functions used by table event rules
- Data structures used by table event rules
- Tables used by table event rules
- Table event rules using a table

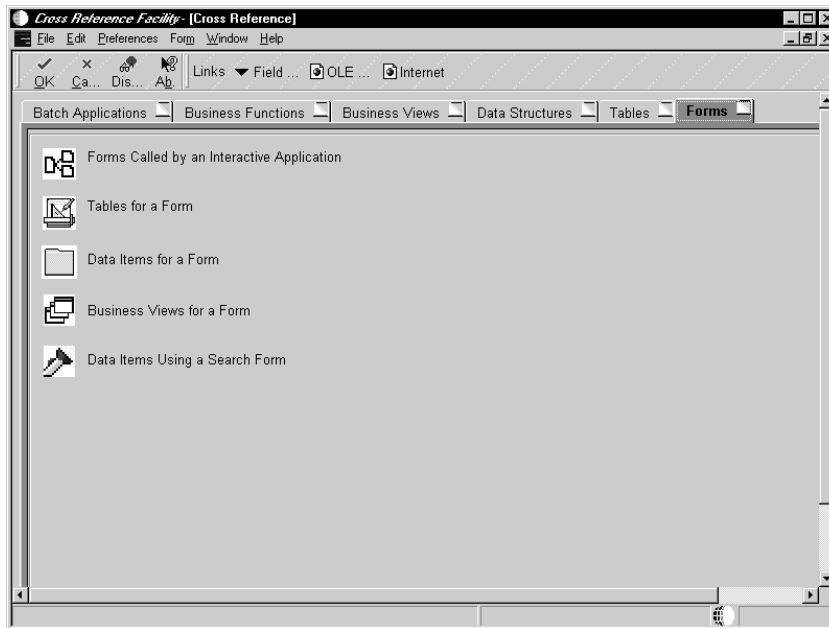


Searching for Forms

You can locate information about forms and how they are used. You can search for the following:

- Forms called by an application
- Tables for a form

- Data items for a form
- Business views for a form
- Data items using a search form

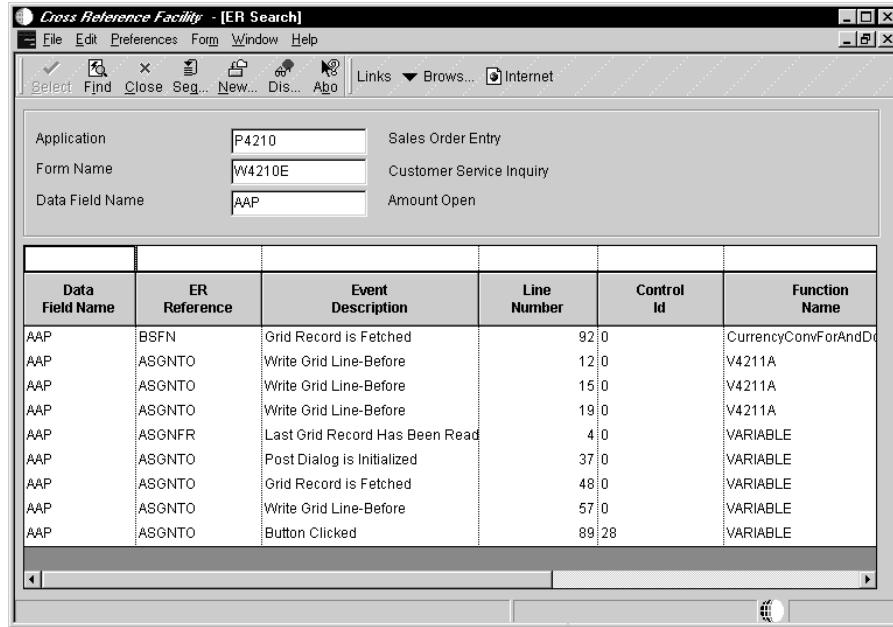


Searching for Event Rules

You search for event rules to help you find an existing event you can use.

► To search for event rules

1. On the search form you have chosen, choose Event Rules from the Form menu.



2. Complete the following fields

- Application
- Form Name
- Data Field Name

Field	Explanation
Application	The OneWorld architecture is object based. This means that discrete software objects are the building blocks for all applications, and that developers can reuse the objects in multiple applications. Each object is stored in the Object Librarian. Examples of OneWorld objects include: <ul style="list-style-type: none"> • Batch Applications • Interactive Applications • Business Views • Business Functions • Business Functions Data Structures • Event Rules • Media Object Data Structures
Data Field Name	The Data Field Name field simply provides the name of a data field being used in either a data base file, a form or a report. This field name is the six character RPG name being referenced in the Data Description Specifications.
Form Name	The name given to a record format for a form, report, or database table.

You can also choose BrowsER from the form menu to browse through event rules. Refer to BrowsER in this guide for more information.

Viewing Field Relationships

The Field Relationships form is meaningful only for the following Cross Reference search types:

DA	Data items used by an application
FA	Forms for an application
FI	Forms using a data item
SA	Data structure for an application

In these instances the Field Relationships form displays the control type for a field, such as:

- BC for a business view column
- FI for a form interconnect control
- GC for a grid control
- FC for a form control

To view field relationships

On the Cross Reference search form you have chosen choose Field Relationships from the Form menu.

Object Name	Form Name	Data Field Name	Description	Event Rules Control	Control Id
P4210	W4210E	AAP	Amount Open	GCW	281
P4210	W4210E	AAP	Amount Open	GCW	421
P4210	W4210E	AAP	Amount Open	VAR	0
P4210	W4210A	ABAS	Override Price (Y/N)	VAR	0
P4210	W4210A	ACNT	Adjustment Control Code	VAR	0
P4210	W4210A	ACOM	Apply Commission (Y/N)	BC	0
P4210	W4210B	ACOM	Apply Commission (Y/N)	BC	4
P4210	W4210B	ACOM	Apply Commission (Y/N)	FC	102

Rebuilding Cross Reference Information

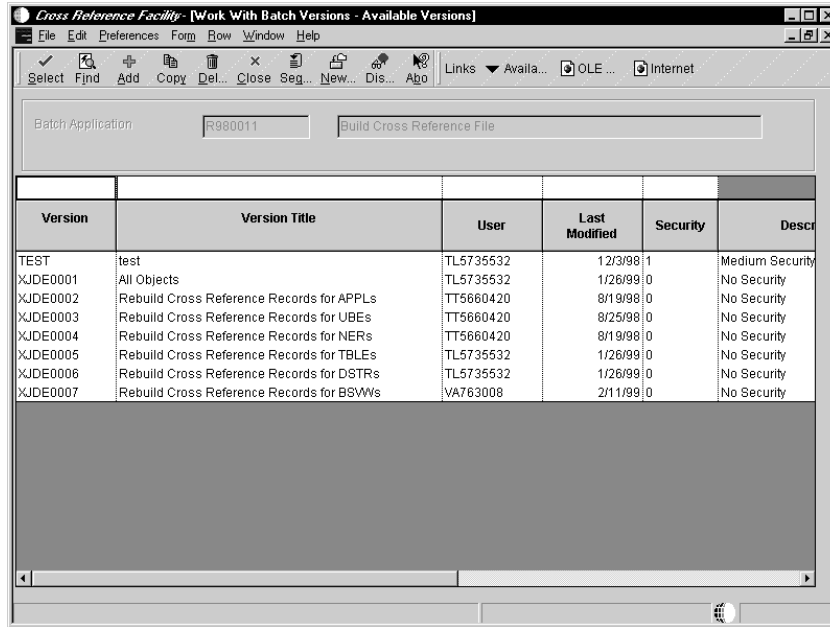
Whenever objects are modified, cross reference information is out of sync with objects in the system. Because the cross reference files are not automatically rebuilt when objects are modified, it is necessary to do so at the time of inquiry. You may also do regularly scheduled builds to ensure that the cross reference information is put in sync on a regular basis.

View the date a record was built in the far right column of the grid on each Cross Reference form of the Cross Reference utility. If information is out of date, use the Rebuild option from any of the Cross Reference forms.

Cross Reference builds using relational database tables, not local specifications.

► To rebuild cross reference information

1. On Cross Reference, choose Rebuild from the Form menu.



2. Choose the objects you want to rebuild.

This process can take several minutes.

Application Design



Application Design

Application Design is the entry point to several tools for creating, generating, running, maintaining, and securing applications. Application Design includes Forms Design for creating forms and Event Rules Design for attaching business logic through event rules. Use Application Design to:

- Access Forms Design for creating forms
- Define processing options
- Run an application
- Access BrowsER
- Create text overrides
- Browse forms in an application

You can use the OneWorld toolset to create applications in different modes, including:

- Windows client
- Java client
- HTML client

See Also

- *Forms Design* for complete instructions on creating a form
- *Processing Options* for creating and attaching a processing options template to an application
- *BrowsER* for instructions on viewing, enabling, and disabling event rules used within forms
- *Developing Web Applications* for more information about developing Web applications
- *Vocabulary Overrides* in the *OneWorld System Administration* guide for information about defining vocabulary overrides for an application

This section describes the following:

- Understanding application design
- Adding an interactive application



Understanding Application Design

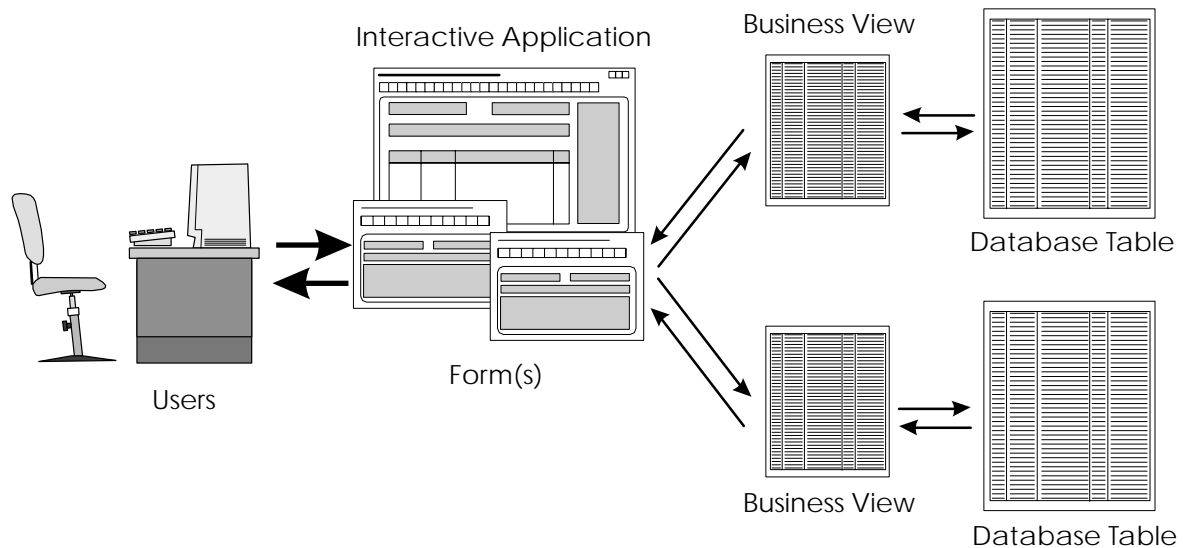
An application is a collection of computer programs that perform a specific task. An application retrieves and updates data within a database table. Accounts Payable and Sales Order Processing are examples of applications.

In OneWorld, there are three types of applications:

- Interactive applications
- Batch applications
- Web applications

Understanding Interactive Applications

An interactive application is the interface between a user and a database table (based on a business view). A user uses an interactive application to add, modify, or view data using a form.



Understanding Batch Applications

A batch process is an application that processes automatically without user interaction. Table conversions and reports are examples of batch processes. See *Batch Processing* for detailed information about batch processes.

Understanding Web Applications

To create Web applications, use the OneWorld toolset in the same manner as you would for standard applications. You can then customize your windows applications for the Web.

In addition to installing the Web server, you may need to install the Java generator on the OneWorld developer workstation. Install the Java generator if you are planning to develop new applications or change existing applications. The Java Generator is usually automatically installed when you do a OneWorld client install.

Refer to *Developing Web Applications* in this guide for more information about developing web applications.

Refer to the *Web-Based Solutions* guide for more information about Web products.

Adding an Interactive Application

An application is an object. You must add an application so it exists as an object before you can begin developing it. You can add a new application, or you can add a version of an existing application.

After the application exists as an object, you can start building the components of the application. You can use Forms Design to design the first form in your application. Instructions for using Forms Design are contained in the next section.

Adding an interactive application describes the following tasks:

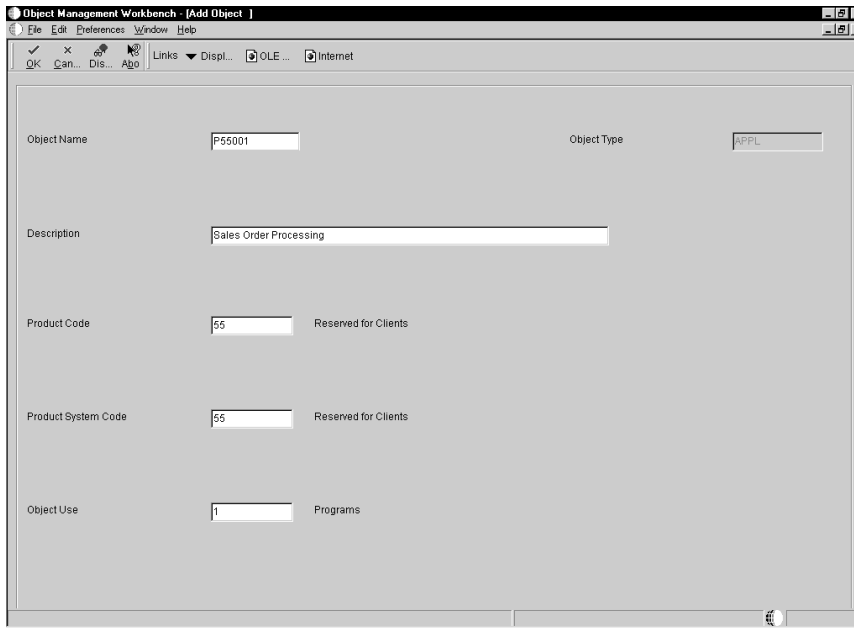
- Creating an interactive application object
- Creating an interactive version object

Create an Interactive Application Object

Use the following process to create an interactive application object.

► To create an interactive application object

1. On Object Management Workbench, click Add.
2. On Add OneWorld Object to the Project, choose the Interactive Application option, and then click OK.



3. On Add Object, complete the following fields, and then click OK:

- Object Name

This field accepts up to 10 characters; however, if you enter more than 8 characters, the entry will be truncated. The J.D. Edwards naming standard for applications is formatted as Pxxxxyyy.

P = application

xxxx = the system code

yyy = a next number, such as 001 and 002

- Description

Provide up to a 60-character description. It should reflect the subject of the forms within the application, such as Companies and Constants.

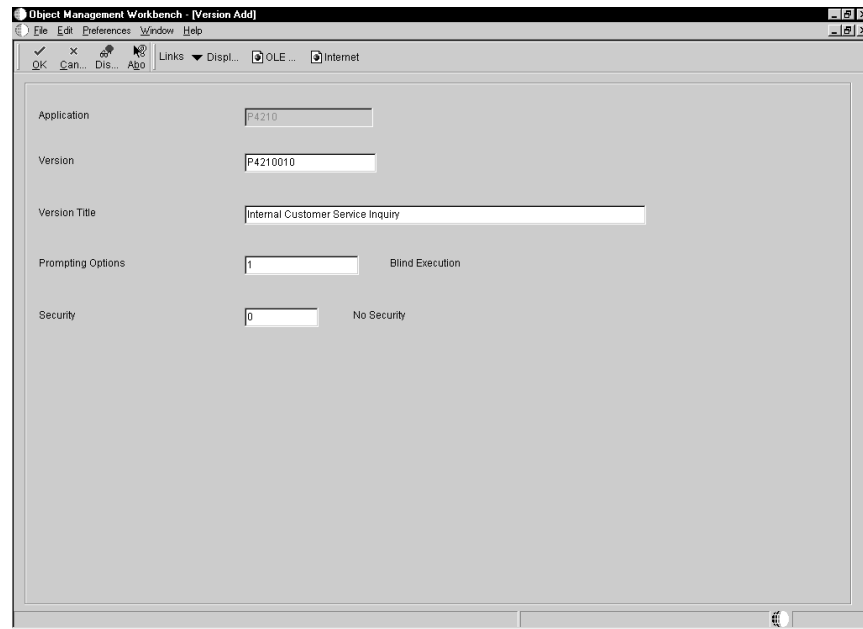
- Product Code
- Product System Code
- Object Use

Creating an Interactive Version Object

Use the following process to create a version of an interactive application.

▶ **To create an interactive version object**

1. On Object Management Workbench, click Add.
2. On Add OneWorld Object to the Project, choose the Interactive Version option, and then click OK.
3. On Adding a Version, enter the object name of the application upon which you want to base this version.



4. On Version Add, complete the following fields, and then click OK:

- Version

This field accepts up to 10 characters; however, if you enter more than 8 characters, the entry will be truncated. The J.D. Edwards naming standard for applications is formatted as Pxxxxyyy.

P = application

xxxx = the system code

yyy = a next number, such as 001 and 002

- Version Title

Provide up to a 60-character description. It should reflect the subject of the forms within the application, such as Companies and Constants.

- Prompting Options
- Security



Forms Design

Use Forms Design to create one or more forms for an application.

This section describes the following

- Understanding form types
- Creating a form
- Designing a form layout
- Working with menu/toolbar exits
- Working with controls
- Overriding data dictionary items at design time
- Using text variables
- Using Quick Form
- Processing media objects
- Testing a form

Before You Begin

Before working with Forms Design, you should have a working knowledge of Windows and have an understanding of the OneWorld application design process. Additionally, you must:

- Think about which data items are required for each form in the application.
- Determine if there are any existing tables that suit your needs. If none exist, create a table.
- Create a business view on which to base a form.

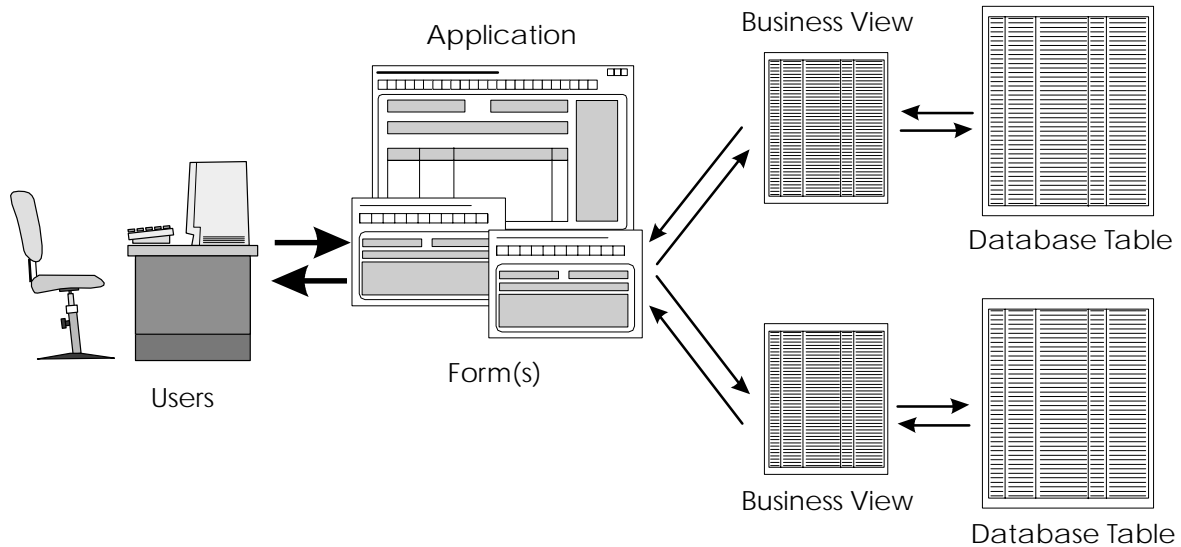
See *About Table Design* and *About Business View Design* in this guide for information about creating tables and business views.



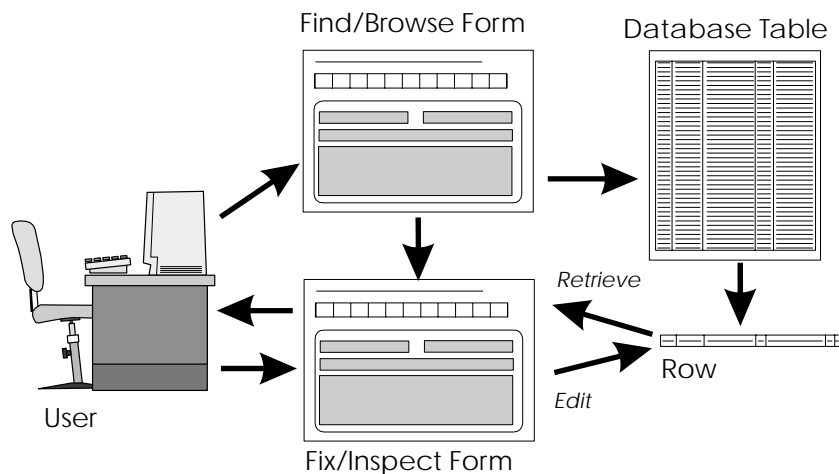
What is a Form?

A form is the interface between a user and a table. This interface should:

- Present data logically
- Contain the functionality necessary to enter and manipulate data



A single application can contain one or more forms. Usually, a Find/Browse form is the first form displayed in the application. It enables the user to locate a specific record with which he or she wants to work. Upon selecting a record, a subsequent form, such as a Fix/Inspect form, is automatically displayed, where the user can view or modify data for that record.



Elements of a Form

Form Type	The form type establishes the basic functionality of a form. Each form type has default controls and processes.
Business Views	In an application, business views link forms and tables. You must associate all forms except the Message form with a business view.
Controls	All objects on a form are controls. Controls include grids, check boxes, radio buttons, push buttons, and more.
Properties	In application design, there are several types of properties: application, form, control, and grid. Properties define appearance and functionality.
Data Structure	A data structure defines the data that can be passed between forms within an application or between applications. Once the data structure of a form is defined, use form interconnections to indicate the direction of flow of data between forms.
Event Rules	<p>Event Rules can contain processing instructions for specific events. Events are actions that occur on a form, such as clicking a button or tabbing out of a field. Use event rules to attach business logic to any event.</p> <p>Events are triggered either as a result from user interaction with a control (for example, click button) or as a result of a system controlled process (for example, loading a grid).</p>

See *Understanding Form Types* for a discussion of each form type.

Understanding Form Types

When you create a new form, you must initially specify a form type. It is important to understand the available form types, because each form has characteristics that accommodate different tasks. For example, the Find/Browse form includes a Find option to search for records across the table and a grid to display those records.

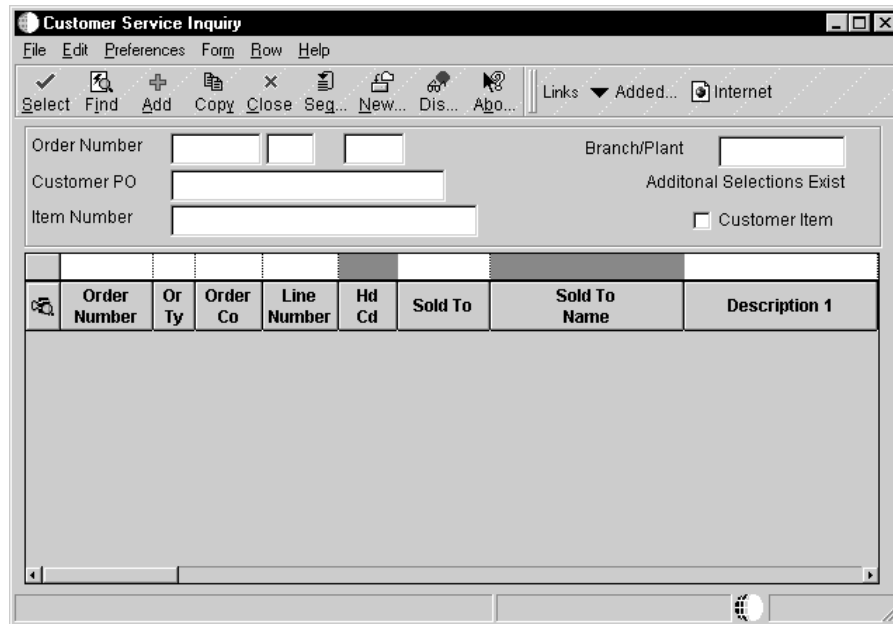
The OneWorld Forms Design tool supplies the following form types:

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

About J.D. Edwards Design Standards

J.D. Edwards design standards ensure a consistent approach across all applications. While many of these standards apply to all form types, there are also specific standards for each form type. See the *Development Standards: Application Design Guide* for a complete list of standards.

Find/Browse Forms



Find/Browse is the entry point to an application. It contains an optional query-by-example (QBE) line so you can search on any database field in the grid. QBE columns that are grayed out do not have QBE capability (for example, Sold To Name in the example above).

Use a Find/Browse form to:

- Search, view, and select multiple records in a grid
- Delete records
- Exit to another form to add, update, or view a table record

Find fetches records; the filter fields and QBE line provide the WHERE clause of a SELECT statement. Delete removes records from the table.

You cannot add new or update existing records on a Find/Browse form.

A Find/Browse form displays data contained in one table. Therefore, you can attach only one business view to a Find/Browse form.

The following table describes toolbar functionality for this form.

Select	Standard toolbar button that comes with the form. You must add the appropriate form interconnections to provide functionality.
---------------	--

Find	Standard toolbar button that comes with the form
Close	Standard toolbar button that comes with the form
Add	You can add this button to the toolbar. You must also provide the functionality for the button, which typically includes a form interconnect.
Copy	You can add this button to the toolbar. You must also provide the functionality for the button.
Delete	You can add this button to the toolbar. You must also provide the functionality for the button.
OK	This button does not apply to this form type.
Cancel	This button does not apply to this form type.

Fix/Inspect Forms

The Fix/Inspect form allows you to add a new record to a table or to update an existing record. The Fix/Inspect form includes OK and Cancel buttons. When you click OK, updates or additions are written to the table. When you click Cancel, any changes you have made are lost and no database changes are made. Because the Fix/Inspect form only allows you to add or update one record at a time, the form does not contain a grid.

Use this form type to:

- View a single record per form
- Add new records
- Update existing records

Because the Fix/Inspect form only contains one record, you can attach only one business view to a Fix/Inspect form.

If a record was selected on a previous form, the Fix/Inspect form displays data for that record. If no record was selected on the previous form, the Fix/Inspect form is empty, except for any default values.

The following table describes toolbar functionality for this form.

OK	Standard toolbar button that comes with the form.
Cancel	Standard toolbar button that comes with the form.
Select	This button does not apply to this form type.
Find	This button does not apply to this form type.
Close	This button does not apply to this form type.
Copy	This button does not apply to this form type.
Delete	This button does not apply to this form type.
Add	This button does not apply to this form type.

Header Detail Forms

Item Number	Quantity Ordered	Tr. UoM	Unit Cost	Extended Cost	Pu. UoM	Ln Ty	Descript
				0.00			

The Header Detail form allows you to work with data from two separate tables. You can use this form to add or update a single header record. You can also add, update, or delete multiple detail records from the same form.

The Header Detail form includes an input-capable grid so you can add or update detail records. Click OK to perform updates or adds to both tables on the form. When you click Cancel, any changes are lost and no database changes are made.

Because the Header Detail form allows you to update or add records from two different tables, you can attach two business views to a Header Detail form. Attach one business view to the grid and the other to the form, updating both tables from a single form. You can use the Header Detail form for one to many relationships.

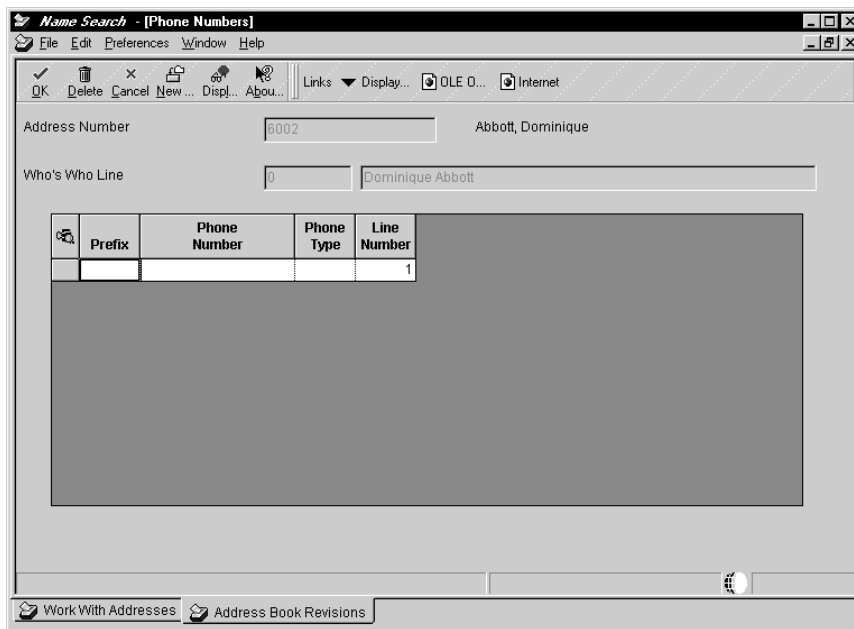
The following table describes toolbar functionality for this form.

OK	Standard toolbar button that comes with the form.
Cancel	Standard toolbar button that comes with the form.

- Find** You can add this button to the toolbar. You must also provide the functionality for the button.

The engine will perform a standard fetch just like it does when the form was entered.
- Delete** You can add this button to the toolbar. Even though the tool will delete the record, you may wish to add other logic here.
- Select** This button does not apply to this form type.
- Close** This button does not apply to this form type.
- Copy** This button does not apply to this form type.
- Add** This button does not apply to this form type.

Headerless Detail Forms



The Headerless Detail form is used to display multiple records from a single table that is not normalized. Because this form is used to update only one table, you can attach only one business view to the form.

The Headerless Detail form contains an input-capable grid, where you can add or update detail information. The header portion of the form displays data

common to all the detail records in the grid. Both header and detail information come from the same business view.

Click OK to perform updates or additions to the table. When the user clicks cancel, any changes you have made are lost, and no database changes are made.

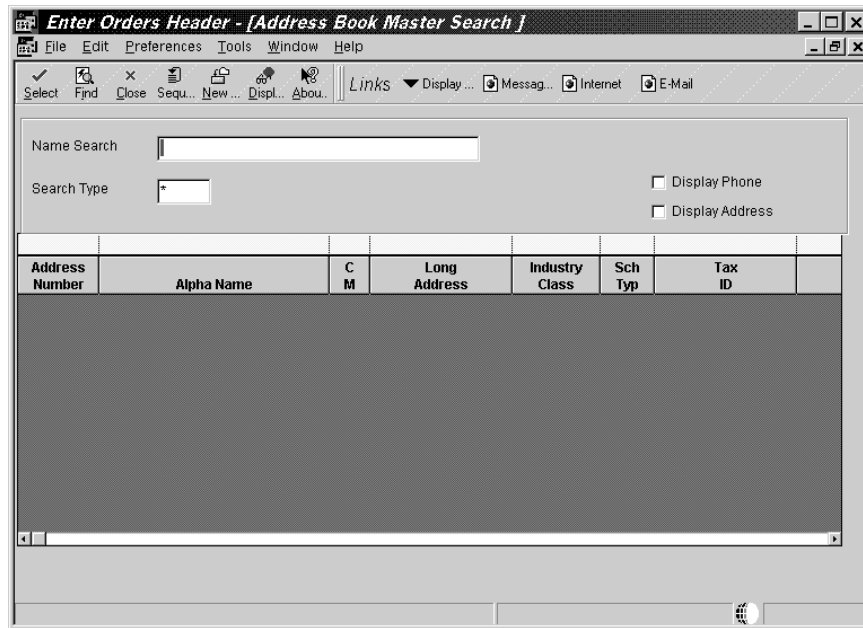
Use this form type to:

- Display multiple records in a grid
- View, add, change, and delete records

The following table describes toolbar functionality for this form.

OK	Standard toolbar button that comes with the form.
Cancel	Standard toolbar button that comes with the form.
Find	<p>You can add this button to the toolbar. You must also provide the functionality for the button.</p> <p>The engine will perform a standard fetch just like it does when the form was entered.</p>
Delete	You can add this button to the toolbar. You must also provide the functionality for the button.
Select	This button does not apply to this form type.
Close	This button does not apply to this form type.
Copy	This button does not apply to this form type.
Add	This button does not apply to this form type.

Search and Select Forms



Use this form to locate a value and return it to the calling field. The Search and Select form is called using a visual assist (flashlight) or hyper-control.

This form only displays information; you cannot edit fields. Therefore, the form contains Select and Close options.

The Search and Select form includes a non-input capable grid where you can view multiple records in one table. The grid displays valid values. When a user chooses a value from the grid and clicks Select, that value is automatically returned to the calling field.

Because this form is used to view records from only one table, you can attach only one business view to a Search and Select form. You use a separate application for this form. One of the benefits of this form is that it improves performance by fetching only necessary fields and it exists in its own application.

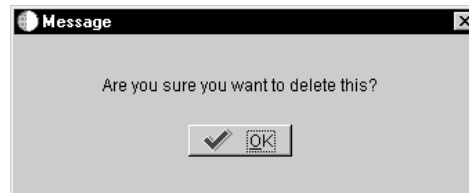
The data structure for this form should contain only one element.

After you create a Search and Select application, you must attach the Search and Select form to the specific data item for which it was created. You do this using the visual assist trigger in the data dictionary or the overrides in the property sheet for a particular control. You use the data dictionary for all instances and override for just one instance.

The following table describes toolbar functionality for this form.

Select	Standard toolbar button that comes with the form. The Select action will automatically return to the calling form. You do not need to add form interconnects.
Find	Standard toolbar button that comes with the form.
Close	Standard toolbar button that comes with the form.
Copy	This button does not apply to this form type.
Delete	This button does not apply to this form type.
OK	This button does not apply to this form type.
Cancel	This button does not apply to this form type.
Add	This button does not apply to this form type.

Message Forms



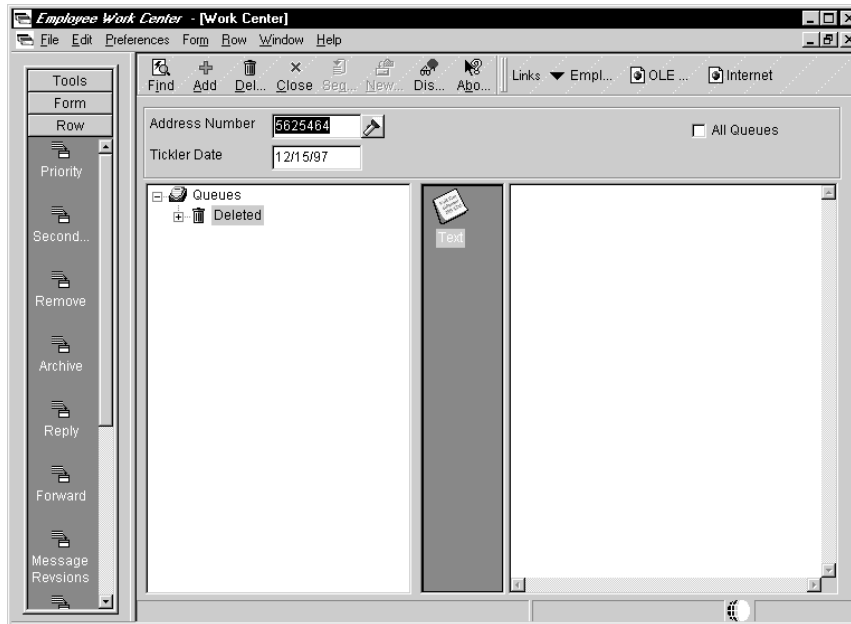
Use the Message form to display messages or request action from the user. The form is modal and is also not sizable. You can only add static text and push buttons to this form. This form is the only one that allows standard push buttons, including OK, Cancel, Yes, and No buttons. Do not use form interconnections on this form.

A delete confirmation is a good example of how you can use the Message form.

There is no business view for this form type.

There is no toolbar functionality for this form.

Parent/Child Forms



You can use the parent/child form to represent parent/child relationships in an application. The form has a parent/child control placed where the grid resides in a Find/Browse form. This control presents a tree view in the left portion of the control, which displays a visual representation of the parent/child relationship. The right portion of the composite control displays a grid in browse mode. A movable bar separates the tree view and grid. Use it to resize either view horizontally. The grid displays the detail information for the nodes of the tree. The parent/child form uses one business view.

There are two modes available on the parent/child form. You can choose the default mode or show all details. The default mode displays the detail for all nodes at the same level in the tree. As you switch levels in the tree, the grid lines change to reflect the level you are in. If you choose to show all details, there is a one-to-one correspondence between the nodes in the tree and the grid lines. You can also load pre-expanded trees so that if you click Find the tree will display all levels instead of just loading one level of nodes at a time. If you choose this option, it shows up as an option on the right mouse menu. You can also allow multiple selects.

Use the Parent/Child form to:

- Represent Parent/Child relationships.
- Perform normal Find/Browse functions (including FETCH and SELECT WHERE statements) and present information in a tree format.

The following table describes toolbar functionality for this form.

Select	Standard toolbar button that comes with the form. You must add the appropriate form interconnections to provide functionality.
Find	Standard toolbar button that comes with the form
Close	Standard toolbar button that comes with the form
Add	You can add this button to the toolbar. You must also provide the functionality for the button, which typically includes a form interconnect.
Copy	You can add this button to the toolbar. You must also provide the functionality for the button.
Delete	You can add this button to the toolbar. You must also provide the functionality for the button.
OK	This button does not apply to this form type.
Cancel	This button does not apply to this form type.

Parent/Child form processing is based on the concept of a parent/child relationship between two data items either within a table or across different tables. If the parent/child relationship is within a table or business view it is an inherent relationship. If the parent/child relationship is between two different tables or business views, then the relationship is an established one.

To display explicit parent/child data, the business view underlying the parent/child form should have a parent column and a child column. When any node on the tree for the form is expanded, a fetch is performed by querying the database for all the expanded node's child records. For example:

- Business unit 10 consists of business units 5 and 3.
- Business unit 5 consists of business units 2 and 1.

There is an inherent parent/child relationship between business unit 10 and business units 2 and 1.

If you simply wish to display data in a hierarchical fashion, the business view does not need to have a parent column and a child column. You must develop the logic to perform a fetch when a tree node is expanded.

The Delete hyperitem will delete the currently selected node record from the table. You must delete child records if needed.

The steps involved in creating a Parent/Child form are different than other form types. To create a Parent/Child form:

- Create the form in Form Design Aid
- Select a Business View
- Highlight the grid and add columns to the grid
- Add filter fields to the form

Choose the Parent field from the business view as the filter because the tool uses this for fetches from the database

Until this point, the form setup for Parent/Child is almost identical to the setup for a Find./Browse form. From this point on, there are unique steps to complete the Parent/Child form. See *Working with Controls* for information about completing the Parent/Child form design.

Creating a Form

You use Forms Design to create one or more forms that display in an application. These forms are the visual interface for the end-user of your application and enable that user to view, add, or modify data stored in one or more tables.

To create a form, you must:

- Select a form type
- Define form properties
- Select a business view

After you create a form, you can also:

- Revise information about a form
- Delete a form

Before You Begin

Before creating a form you must:

- Create a table if there is not an existing one you can use
- Create a business view upon which to base a form
- Add an application in the Object Librarian
- Understand the different form types
- Think about which data items are required for each form in the application

Select a Form Type

Select a form type to support how you want users to interact with the form and what functions you want to be available to users. See *Understanding Form Types* for more information about the types of forms and their underlying functionality.

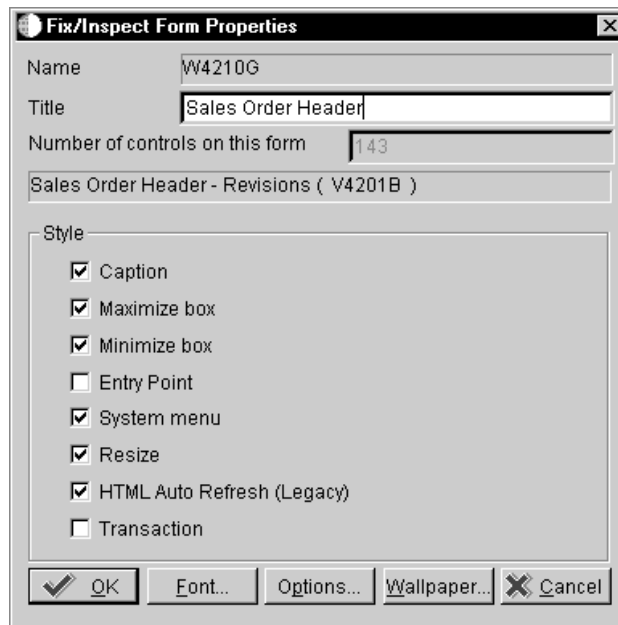
► **To select a form type**

1. On Interactive Application Design, click the Design Tools tab, and then click *Start Form Design Aid*.
2. From the Layout menu, choose Size to Guide to make your forms a standard size.

The form guide appears as a blue box.

3. From the Form menu, choose Create, and select one of the following form types:
 - Find/Browse
 - Fix/Inspect
 - Header Detail
 - Headerless Detail
 - Search and Select
 - Parent/Child
 - Message Form

An appropriate Properties form appears.



The business view name only appears in the gray area below the Title on Form Properties after you define your form properties and select a business view for your form.

Forms Design automatically assigns a name using the following format:

WzzzzzzzA

W = form

zzzzzzzz = is the application name

A = the first form created in the application. It is usually, but not always, the entry point to the application; subsequent forms are assigned sequential letters, such as B for the second form, C for the third and so on.

For example, the application P0101 has two forms. The first form, Work with Addresses, is the entry point and is assigned the name, W0101**A**. The second form, Address Book Revisions, is assigned the name W0101**B**.

Define Form Properties

Use the following process to set the functionality of the form.

► To define form properties

1. On the Properties form, complete the following field:

- Title

Provide a form description based on the form type:

Find/Browse - *Work With* followed by the subject of the application, such as Work With Companies, Work With Constants.

Fix/Inspect, Header Detail and Headerless Detail - should reflect the topic they cover, such as Supplier Information, Item Master Revisions, Purchase Order Entry.

For lower level forms, identify the form that called the form by appending the calling form's title, such as Enter Voucher - G/L Distribution.

When the title of a form includes a verb, use an active verb instead of a nominalization, such as Work With Vouchers.

2. Click one or more of the following Style options:

- Caption
- Maximize/Minimize

- Entry Point

Entry Point indicates that the form is the first form displayed when the application is accessed.

- System Menu
- Resize
- Transaction Processing

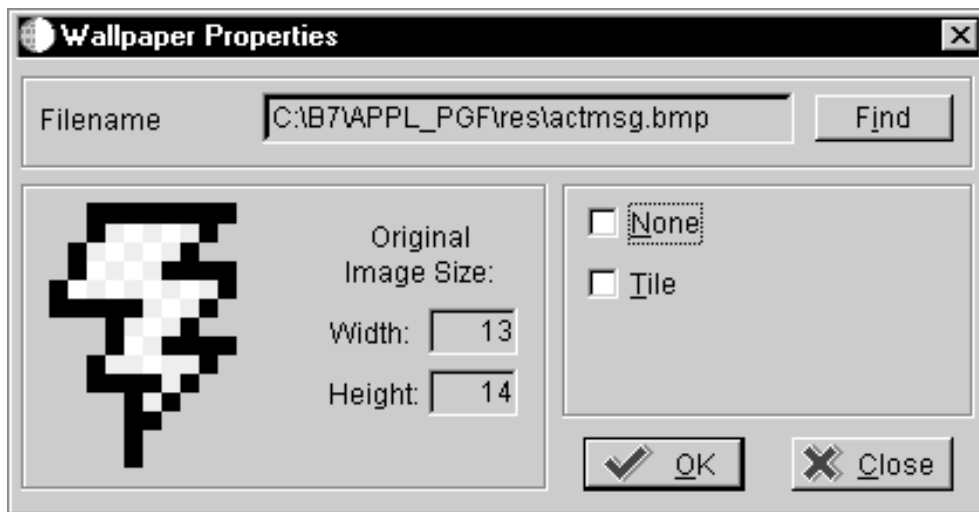
See *Transaction Processing* for instructions about defining boundaries for transaction processing.

- HTML Auto Refresh (Legacy)

See *Understanding the HTML Client* for information about turning on and off HTML post for all non-critical events on a form.

3. Click the following buttons to further alter or define the form:

- Font
- Options
- Wallpaper



If you turn on the Wallpaper option, you can designate a bitmap to be used as background wallpaper for the form.

- None

If you click the None option, the bitmap is no longer used.

- Tile

If you turn on the Tile option, multiple smaller images of the bitmap are used to fill the background instead of one larger image.

Field	Explanation
Title	<p>The title for the form, field, control, or button.</p> <ul style="list-style-type: none"> • For a form: This will appear on the form at runtime. • For a field, control, or button: If the title is longer than the size of the control or button, it will be truncated. Resize the control to show the complete title.
Caption	Displays the form title at runtime.
Maximize/Minimize	Allows the user to minimize or maximize the form. When you turn on these options, the minimize and maximize buttons are displayed in the upper right corner of the form.
Entry Point	Designates the form as the initial form displayed in the application. There is only one entry point per application. If you are starting your application from a Fix/Inspect form, it does not need an entry point designated. Use this option to make the entry point form accessible from a menu.
System Menu	Includes the system menu button in the upper left corner of the form.
Resize	Allows the user to resize your form at runtime. Use this feature along with the modeless option.
Transaction	Enables transaction processing for this form. Transaction processing occurs within the specified transaction boundary so that database operations are stored in a queue until a commit command is issued.

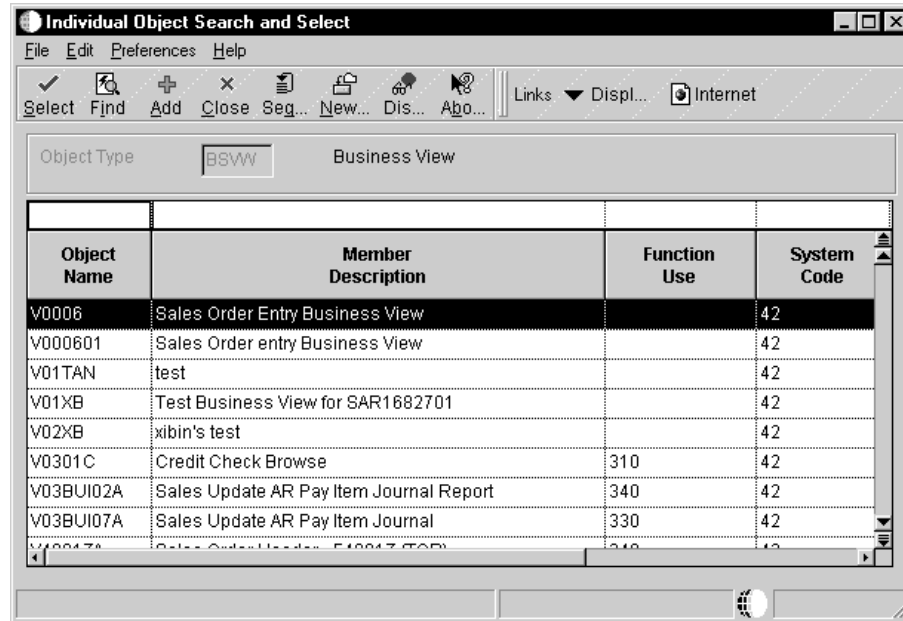
Field	Explanation
HTML Automatic Refresh	<p>The HTML client refreshes during the following events:</p> <ul style="list-style-type: none"> Set Focus on Grid, Push Button Clicked, Check Box (Only if ER exists), Radio Button (Only if ER exists), Bitmap clicked, URL clicked, Tabs selected, Tree Control Actions: (Node expanded, Drag and drop), Events marked as HTML Post events. <p>If HTML Automatic Refresh is enabled, the HTML client will also refresh for every event that contains the following ER:</p> <ul style="list-style-type: none"> Hide/Show Object, Hide/Show Column (both Grid and Parent/Child), Enable/Disable Grid <p>If the Automatic Refresh is disabled the event level attribute for HTML Post may be used to force a refresh on any or all of the following events:</p> <ul style="list-style-type: none"> Edit Control (CONTROL_IS_EXITED, CONTROL_IS_EXITED_CHANGED_INLINE, CONTROL_IS_EXITED_CHANGED_ASYNC) Grid Control (COLUMN_IS_EXITED, COLUMN_IS_EXITED_CHANGED_INLINE, COLUMN_IS_EXITED_CHANGED_ASYNC) Parent Child Form Grid (TREE_NODE_LEVEL_CHANGED) Tree Control (TREE_NODE_SELECTED) <p>Enabling HTML Automatic Refresh may cause the HTML client to refresh very often and result in poor performance. If this occurs, disable the HTML Automatic Refresh and enable an HTML refresh for each appropriate event using the HTML Post event flag.</p>

Select a Business View

Forms require a business view from which to read and write data. Use the following process to associate a business view with a form.

► To select a business view

1. On the form with which you are working, from the Form menu, choose Business View.



2. Choose the business view to attach to the form.

Revise Information about a Form

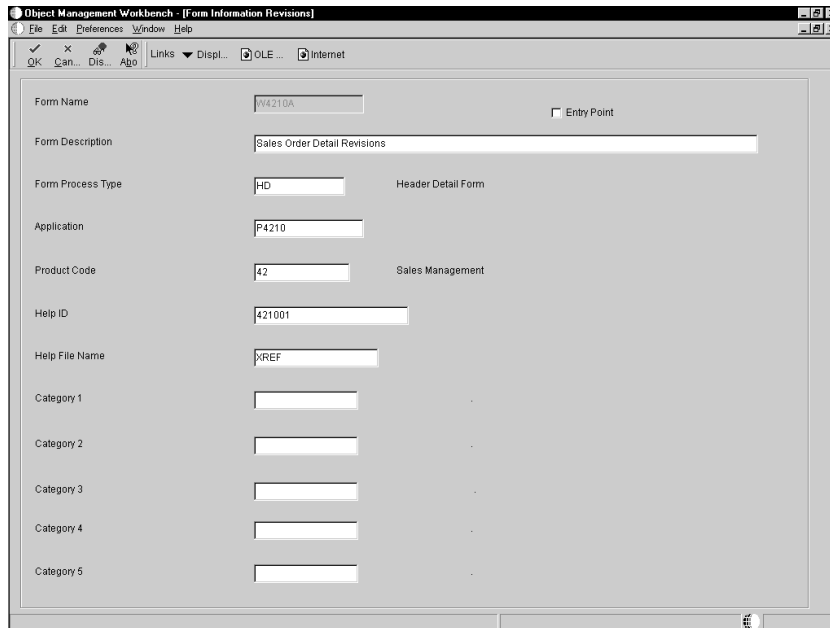
You can assign and change data such as a form's description and its help ID as well as metadata such as its product code.

► To revise information about a form

1. On Interactive Application Design, click the Design Tools tab, and then click View Forms.
2. On Work with Forms, choose one of the following options:
 - Local Forms
 - Checked-in Forms

When you create a form, it is local to your machine until you check the form in. When you check in the form, the F9865 file is updated. You can view forms that have been checked in or forms that are local to your machine.

3. Choose one of the available forms to view or revise more detailed information about the form.



Delete a Form

► To delete a form

1. On Form Design Aid, choose Destroy from the Form menu.
2. Confirm the deletion.

Caution: Once you delete a form, you cannot retrieve it.

Alternately, you can delete a form on Work with Forms by choosing the form to delete and then clicking Delete.

Designing a Form Layout

Forms Design includes features that help you to define your form layout. This section highlights these features and provides you with information about:

- Selecting and moving controls
- Changing the size of grids, and controls
- Using the cut, copy, and paste commands in forms
- Aligning controls
- Using undo and redo

Selecting and Moving Controls

Select and move controls as you design the form. You can:

- Move a control
- Select and move a group of controls
- Move a control and static text (and also any associated text)

▶ **To move a control**

1. In Forms Design, on the form with which you are working, position the mouse pointer over the control.

The pointer changes to the hollow square design tool.

2. Click and hold the mouse button as you drag the control to where you want it. Then release the mouse button.

▶ **To select and move a group of controls**

1. On the form with which you are working, place the mouse pointer outside the group of controls you want to select.
2. Press and hold the mouse button. The pointer changes to the pencil design tool.

3. Drag the mouse to the opposite corner of the group of controls.

As you drag the mouse, a rectangle is drawn from the point you first placed the arrow design tool. Make sure the rectangle completely surrounds all the controls you want to select.

4. Release the mouse button. The selected group of controls is surrounded by a box.
5. Position the mouse pointer over the selected group.

The pointer changes to the hollow square design tool.

6. Click and hold the mouse button as you drag the group to where you want it. Then release the mouse button.

To move a control and static text

1. On the form with which you are working, position the mouse pointer over the shaded area between the static text and the control box.
2. When the pointer turns into the hollow square design tool, press and hold the mouse button. When the crosswire appears, you can move both the static text and the control box together.
3. Drag the controls to where you want them. Release the mouse button.

If the static text and the control box are too close together, the hollow square design tool will not appear in the shaded area. If this occurs, move the text and the control further apart. You can also use the pencil tool to draw a box around just one control and move it independently.

Changing the Size of Grids, and Controls

There are two methods you can use to change the size of a control:

- Size controls using the mouse
- Size and place a control using the Size command

The Size command allows you to specify placement on the form. You can also specify size and placement for only one control at a time.

J.D. Edwards Design Standards

- The length of string fields must be at least the minimum size of the string of characters.

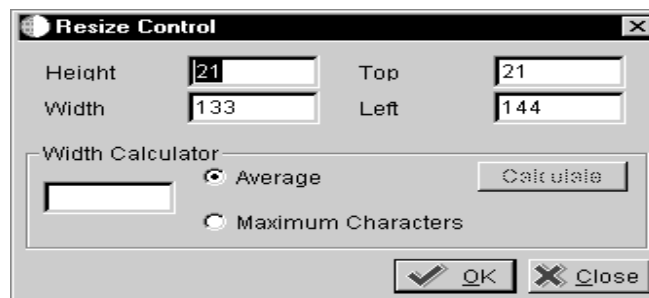
► To size controls using the mouse

1. On the form with which you are working, choose the control.
Handles appear on the control.
2. Move the mouse pointer over one of the handles.
The mouse pointer changes to the filled-square design tool.
3. To change the width, drag a handle at the left or right of the control.
To change the height, drag a handle at the top or bottom of the control.
To change both the width and the height, drag a handle at the corner of the control.

Resize the form and grid using the same method.

► To size and place a control using the Size command

1. On the form with which you are working, choose the control.
2. From the Layout menu, choose Size.



3. To specify the exact control size in pixels, complete the following fields:
 - Height/Width
4. To have the tool automatically calculate the width of a control, type the number of characters in the Width Calculator field, and click one of the following options:
 - Average
 - Maximum Characters

Average width is about the size of a lowercase x. Maximum width is about the size of an uppercase W.
5. Click the Calculate button.

The new calculated width, in pixels, appears in the Width field.

6. Click OK to accept the width.

The Resize Control window remains open for additional changes.

7. To specify the exact position of the control on the form, complete the following fields in pixels:
 - Top
 - Left

Pixels are calculated from the top and left margins of the form.

8. Click Close when you are finished.

Field	Explanation
Height/Width	Specifies the control measurements in pixels.
Top/Left	Designates the placement, in pixels, of a control on the form.
Width Calculator	Calculates the size of a control. To calculate the size, indicate whether the value in the Width Calculator is an average or maximum number of characters. Click on Calculate and the width is automatically adjusted. The new width in pixels appears in the width field.
Average	Used to calculate and resize the width of a control. The new width is calculated using the value you enter in the Width Calculator field.
Maximum Characters	Resizes the width of a control based on the value you enter in the Width Calculator field.
Calculate	Calculates the size of a control. To calculate the size, indicate whether the value in the Width Calculator is an average or maximum number of characters. Click on Calculate and the width is automatically adjusted. The new width in pixels appears in the width field.

Using the Cut, Copy, and Paste Commands in Forms

You use the Cut, Copy, and Paste commands from the Edit menu to move and copy controls within the same form or on another form within the same application. You can also create a new form or application into which to paste the controls.

▶ To cut, copy, and paste controls

1. On the form with which you are working, choose the control or group of controls.
2. From the Edit menu, choose Copy or Cut.
3. Select the form into which you want to paste the control or group of controls.
4. From the Edit menu, choose Paste.
5. Move the outline of the controls to the desired location on the form.
6. Click once to place the controls.

Event rules attached to a control are copied, cut, or pasted when you copy, cut, or paste the control. You cannot copy or paste complex controls such as a tab control, grid control, or grid column.

Aligning Controls

Use the Layout menu to:

- Align a group of controls
- Use an alignment grid to align controls

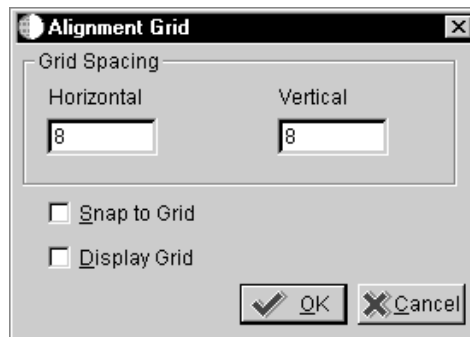
▶ To align a group of controls

1. On the form with which you are working, select the group of controls you want to align.
2. Click the control within the group to which you want to align the group of controls.
3. From the Layout menu, choose one of the following commands:
 - Align Left
 - Align Center
 - Align Right
 - Align Top
 - Align Middle
 - Align Bottom
 - Align Database

Field	Explanation
Align Left	Aligns a group of controls vertically with the left edge of the selected control.
Align Center	Centers a group of controls vertically in relation to the selected control.
Align Right	Aligns a group of controls vertically with the right edge of the selected control.
Align Top	Aligns a group of controls horizontally with the top edge of the selected control.
Align Middle	Centers a group of controls horizontally in relation to the selected control.
Align Bottom	Aligns a group of controls horizontally with the bottom edge of the selected control.
Align Database	Aligns a group of database or dictionary items. This left aligns the static text controls with the selected control and automatically aligns the associated control boxes as well.

► **To use an alignment grid to align controls**

1. On the form with which you are working, from the Layout menu, choose Grid Alignment.

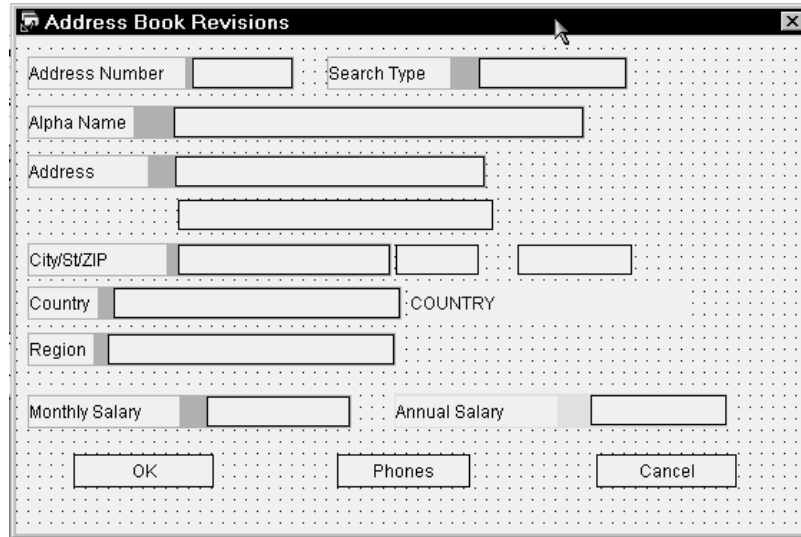


2. To determine grid spacing, complete the following fields:
 - Vertical/Horizontal Placement
3. Click one or both of the following:
 - Snap to Grid
 - Display Alignment Grid

Form Design retains the grid alignment settings from session to session. Both of these settings are on by default. It is important to have the grid alignment setting on when you are developing web applications because

it is quite noticeable if controls are not properly aligned. Use the snap to grid for placing, moving, or resizing controls or grid columns.

The following example shows a form displaying the alignment grid.



Field	Explanation
Vertical/Horizontal Placement	Specifies the space between the horizontal and vertical grid coordinates. The default spacing is 8 pixels between each horizontal and vertical gridline.
Snap to Grid	Places the control to the closest vertical or horizontal grid line when you move the control on the form.
Display Alignment Grid	Displays the alignment grid on the form in design mode.

Using Undo and Redo

You use undo to undo the last change you made. You can set the depth on the undo feature to undo many (over 100) changes back. However, a large setting requires a large buffer size and may lead to performance degradation.

Using undo and redo contains the following tasks:

- Setting the undo depth
- Undoing and redoing changes

▶ **To set the undo depth**

1. On Form Design, from the edit menu, choose Set Undo Depth.
2. Enter a number to indicate how many steps back you can undo.

▶ **To undo and redo changes**

1. On Form Design, to undo a change (that is to move backward in the stack of changes), from the Edit menu, choose Undo.

Each time you choose Undo, the tool reverts back by one change.

2. To reapply a change (that is, to move forward in the stack of changes), from the Edit menu, choose Redo.

Each time you choose Redo, the tool reapplies one change.

Working with Menu/Toolbar Exits

Menu/Toolbar Exits can be items that are only in the menu or items that appear in the menu and on the toolbar. An item must be placed in the menu in order to appear on the toolbar. The toolbar is a visual shortcut to menu items. You can display a bitmap next to an item in a form or row menu or on the toolbar. You can create custom exits, or you can use any of the standard exits OneWorld provides for a form. You can also modify these standard exits.

To create a Menu/Toolbar exit, you:

- Designate the exit category
- Define the exit properties
- Attach event rule logic if needed
- Select a bitmap for the exit if you are going to show the item on the toolbar

Working with menu/toolbar exits describes the following tasks:

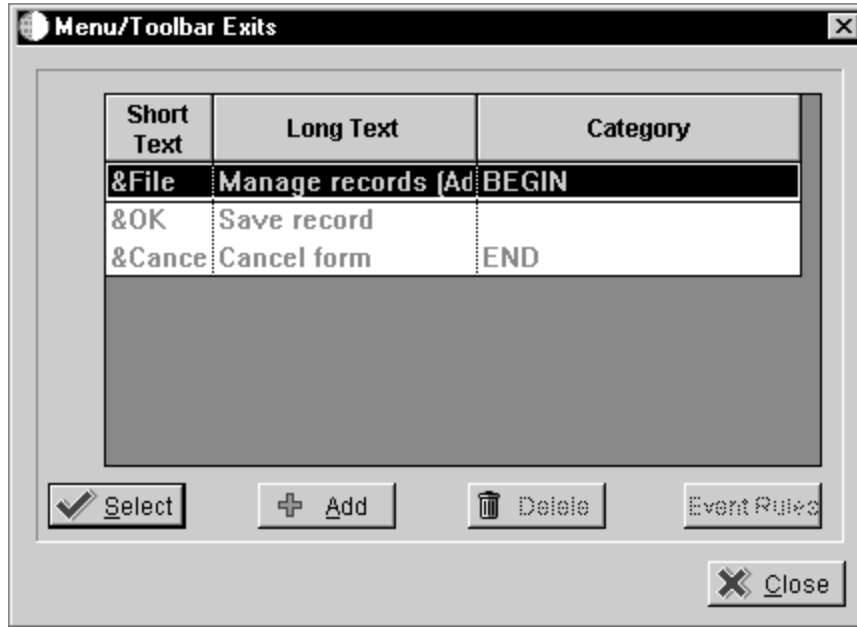
- Creating, editing, or deleting a menu/toolbar exit
- Adding bitmaps
- Attaching event rules to a menu/toolbar exit
- Creating subcategories in menus

Creating, Editing, or Deleting a Menu/Toolbar Exit

Use the following process to create, edit, or delete menu and toolbar exits.

▶ **To create, edit, or delete a Menu/Toolbar exit**

1. In Form Design, on the form with which you are working, choose Menu/Toolbar Exits from the Form menu.



Depending on the form type you are working with, several standard exits will appear. These might include:

- OK** Accepts the data in the form, clears fields, and remains in the form. However, in update mode OK closes the form. In add mode OK may or may not close the form.
- Cancel** Closes the form and returns to the previous one. Any additions, revisions, or deletions the user made in this form are ignored.

2. If you do not want one of the listed exits on your form, choose the exit and click Delete.
3. Choose an exit and click select to edit its properties.
4. To create a new exit, click Add.
5. On Exit Properties, choose the appropriate class.

Classes available depend on the form type. The classes and choices available on the Exit Properties form will also vary depending on the type of exit you select.

Form, row, and view are standard exits and are the only Begin Categories you use. All exits under these categories are user defined, and the last one in the list requires an End Category. The exits you create appear on a

drop-down menu on the toolbar. If you select a Row Category, all exits under it need the Grid option turned on.

6. Complete the following fields:

- Short Text

The short text includes the access key.

- Long Text

The long text appears in the status bar.

7. Complete the Exit Properties form and click OK.

Every Begin category must have a corresponding End category. There can be many exits between the Beginning and End categories.

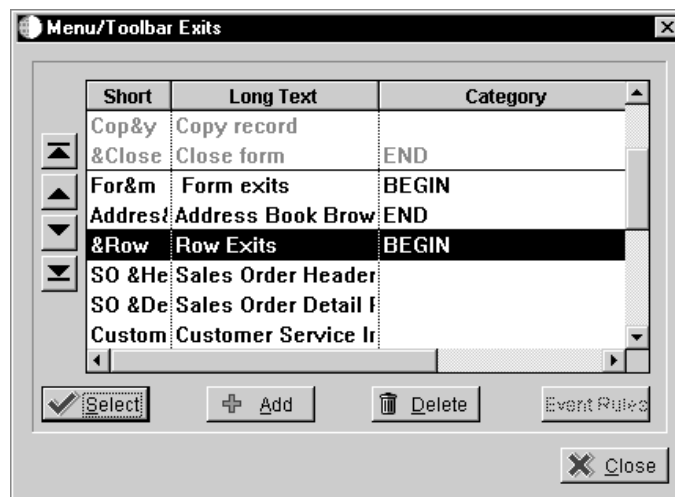
Field	Explanation
Short Text	The short description of the item that appears on the menu and exit bar.
Long Text	The long description of the hyper-item that appears on the menu.
Begin Category	Indicates a category which will appear on the menu bar, such as Row, Form, Report, and View or as a title of cascading (sub) menu.
Toolbar	Specifies that the hyper-item should be placed on the toolbar.
End Category	Specifies that the hyper-item is the last item in this category.
Close Category	Nests hyper-exits for multiple hyper-categories. Closes all open Begin Categories above it.
Enabled	Specifies that the item is initially enabled. (You can then use a system function to disable the item.)
Disabled	Specifies that the item is initially disabled.
Command	Defines the item as a command button. Leave it unselected to define the hyper-item as a menu item.
Toolbar Separator	Specifies that a separator should be placed on the toolbar before this hyper-item.
Menubar Separator	Specifies that a separator should be placed on the menubar after this hyper-item.

Adding Bitmaps

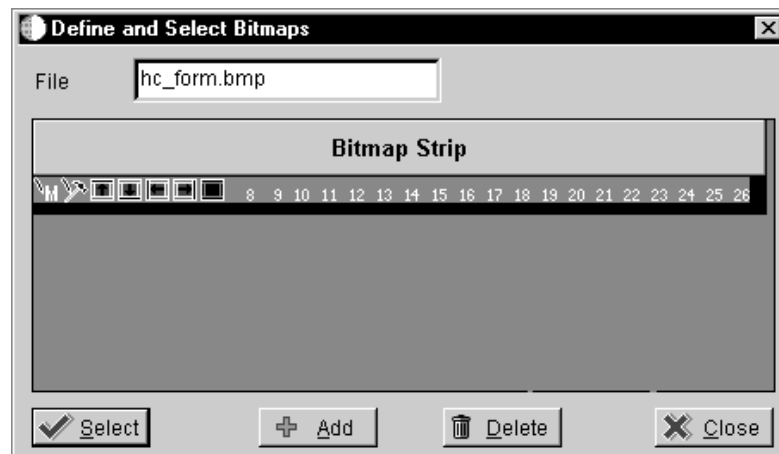
No bitmap strips initially appear for you to choose. Bitmap strips are stored in the `\package name\res` directory, for example, the `b7\appl_pgf\res` directory. Each bitmap strip is named starting with `hc_`. After you have set that strip on an exit for a Begin category it appears for any item listed under that category. You can also use third party tools to modify and initially create your bitmaps.

► To add a bitmap

1. On Menu/Toolbar Exits, choose a Begin Category.

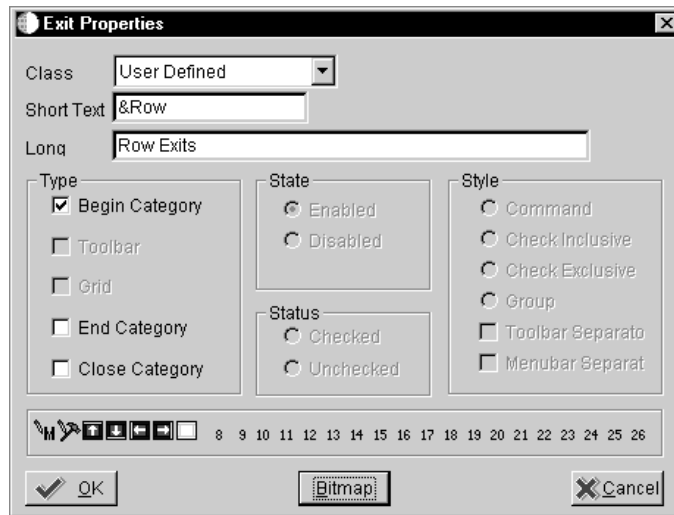


2. On Exit Properties, click the Bitmap button.



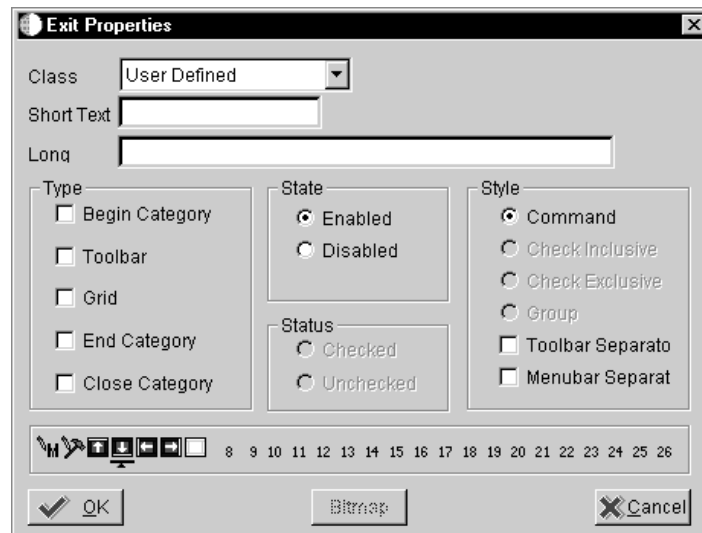
3. Type the file name of your bitmap strip in the File field.
4. Click Add.

- Choose the strip you wish to use and click Select to apply to all the exits under that category.



- On the Exit Properties form click on the specific bitmap that you want to use for each exit.

The bitmap strip you chose contains individuals bitmaps that you can select.

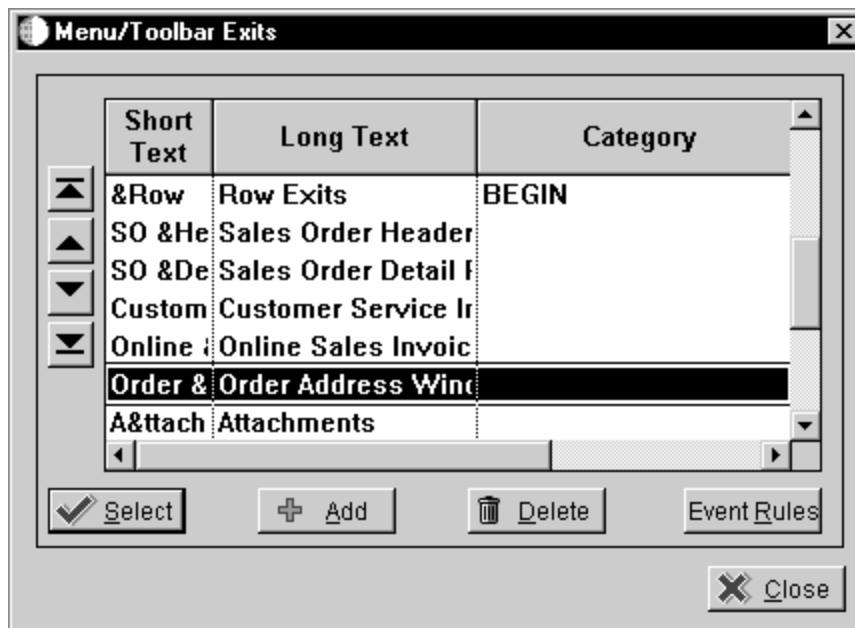


Attaching Event Rules to a Menu/Toolbar Exit

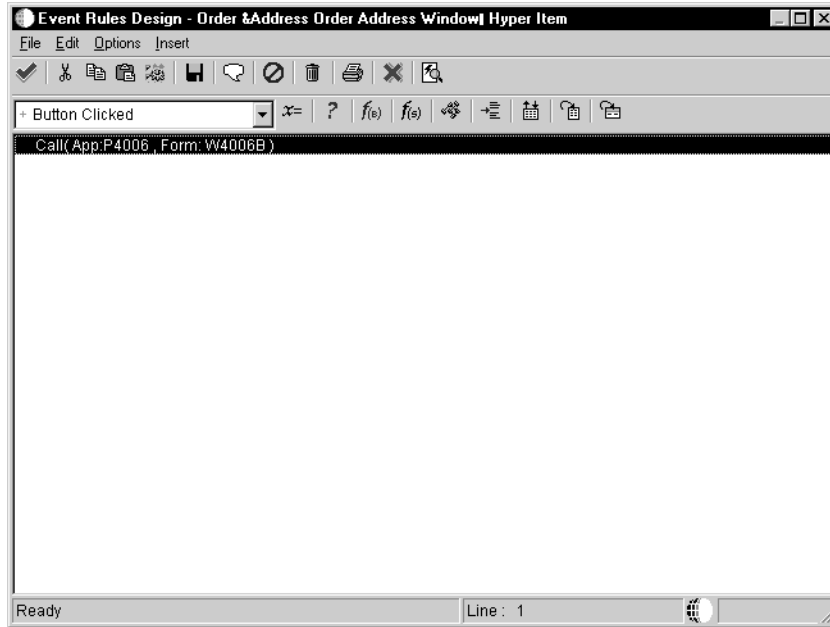
You can attach event rules to an exit. For example, you can attach a form interconnection so that when you choose a menu option another form is called. You cannot attach event rules to Begin Categories. See *Event Rules Design* for more information about creating event rules.

► To attach event rules to a Menu/Toolbar Exit

1. On the Menu/Toolbar Exits form, select the exit you wish to use and click the Event Rules button.



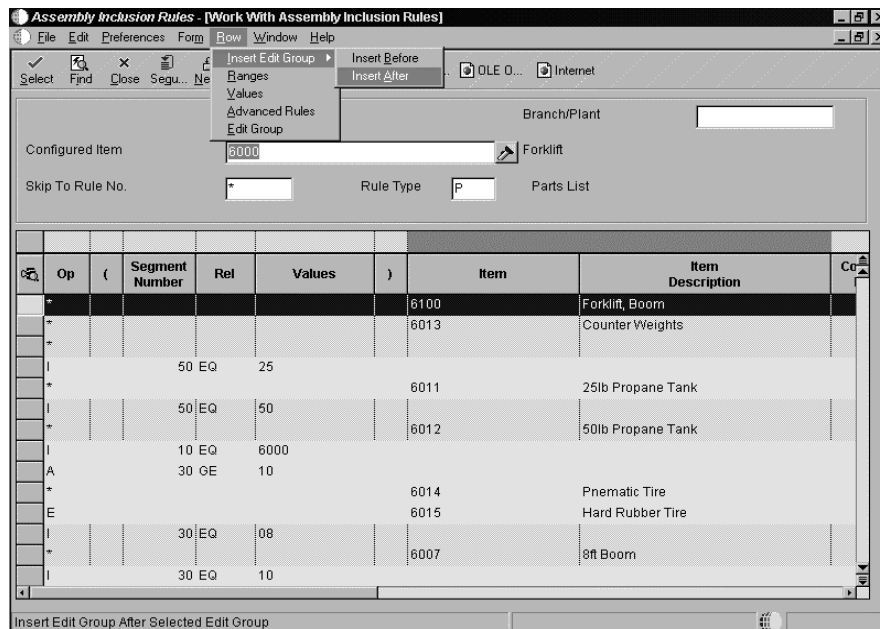
2. On Event Rules Design, choose the Button Clicked event.
3. Add event rule logic you wish to attach.



4. Save, and exit Event Rules Design.

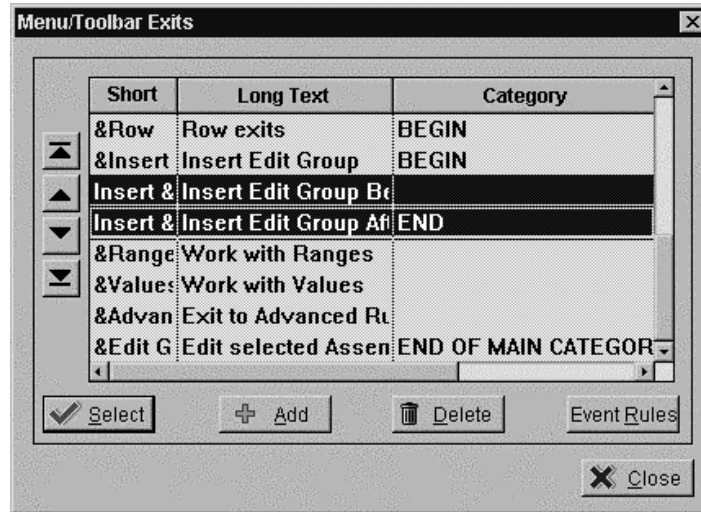
Creating Subcategories in Menus

OneWorld allows you to create subcategories for drop down items (cascading menus):



► **To create a subcategory in a Menu**

1. Choose the Menu/ToolBar exits within Form Design Aid and nest the Begin and End for that form exit.



Insert Edit Group Before and Insert Edit Group After are further subcategories of the Insert Edit Group menu item.

2. Define event rules for each of the exits.

Working with Controls

Use form controls to provide specific functions within an application. For example:

- Insert field controls to display data, enter data, calculate data, store data permanently or temporarily, or pass data between other fields and forms.
- Place check boxes to provide for multiple selections, or radio buttons to indicate mutually exclusive selections.

You can have a maximum of 250 controls on a form. You can use Form Properties to check the number of controls on a form. You will also receive a warning if you are near the 250 control limit.

Working with controls includes the following tasks:

- Understanding form controls
- Creating push buttons
- Creating check boxes
- Creating radio buttons
- Adding data items to a form
- Creating static text controls
- Creating edit controls
- Creating UDC edit controls
- Creating media object controls
- Creating bitmap controls
- Creating a tree control
- Creating combo boxes
- Creating text boxes
- Creating group boxes
- Defining grid properties
- Defining grid column properties
- Defining properties for parent/child controls
- Defining options for forms, grid and edit controls
- Associating database items, dictionary items, or descriptions

- Changing tab sequence
- Creating tab controls
- Designing forms using multiple modes

Before You Begin

Before you work with form controls, you must:

- Create a form.

Understanding Form Controls

Each form automatically includes certain default controls, depending on the type of form you are creating. However, you might need to add additional controls as you design the form. Choose from standard Windows graphical controls as well as J.D. Edwards custom controls. Available controls include the following:

- Push Button
- Check Box
- Radio Button
- Edit
- Static Text
- UDC Edit
- Grid
- Parent/Child
- Media Object
- Group Box
- Bitmap
- Tree Control
- Tab Control
- Business View Field
- Data Dictionary Field

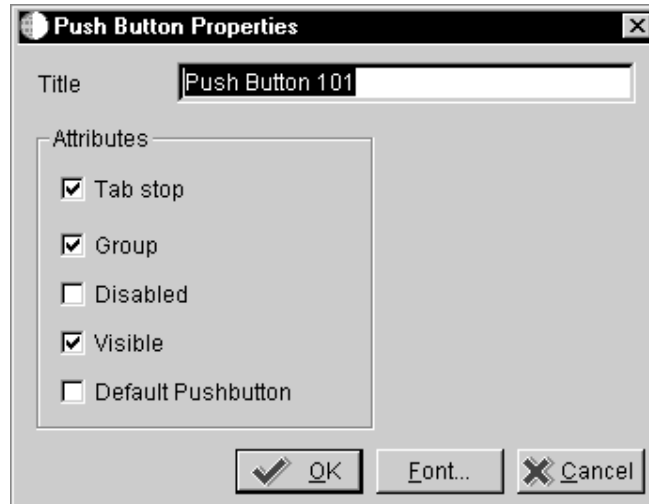
You can disable controls at design time and then enable them at runtime using event rules. If a control is disabled, it is grayed out. If a control is read-only, you can set focus on it or tab into it, but you cannot enter values.

Creating Push Buttons

You can use a push button to initiate an action or a set of actions.

► To create a push button

1. On the form with which you are working, choose Push Button from the Insert menu.
2. Position the control outline on the form, and then click.
3. To define the push button properties do one of the following:
 - Double-click the push button control
 - Choose the push button and choose Item Properties from the Edit menu.



4. Complete the following field:
 - Title
5. Click one or more of the following Attributes:
 - Tab Stop
 - Group
 - Disabled
 - Visible
 - Default Push Button
6. Assign an access key by inserting the ampersand symbol (&) in front of the particular letter in the push button name (optional).

An access key provides a quick way to execute a push button using a combination of the Alt key and the specific alpha key. An access key is identified by any underlined letter within the push button name. Pressing the Alt key and that letter on your keyboard simultaneously is the same as clicking the button with your mouse.

Field	Explanation
Title	<p>The title for the form or control. If the title is longer than the size of the control it will be truncated. Resize the control to show the complete title.</p> <p>For event rules title, this is the default name of the control for use in an event rule. The event rules title defaults to the static text associated with an item. If there is no associated static text and you want to use this control in an event rule, enter a name.</p>
Tab Stop	Allows the user to tab into this control.
Group	Defines this control as the first in a series of related controls. A group is defined as this control plus all subsequent controls without the Group option turned on. The next control with Group turned on signals the start of a new group (a single control can form a group). Within a group of more than one control, use the arrow keys to move from one control to another instead of pressing Tab.
Disabled	Disables a control. A disabled control is grayed out at runtime.
Visible	Makes a control visible.
Default Push Button	Executes the selected push button upon pressing the Enter key.

Creating Check Boxes

You use check boxes to indicate choices. A check mark in a box indicates that you have made a choice. Check boxes are not mutually exclusive.

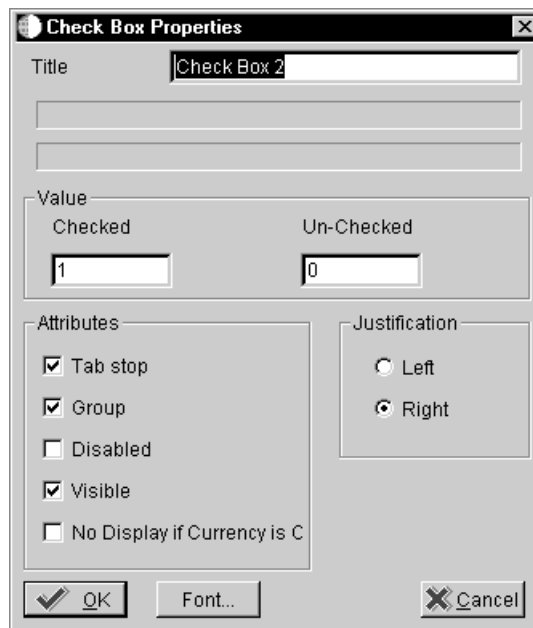
- You must associate a check box with a database or dictionary item.
- You can use a check box to pass a value to an event rule.

See Also

- *Associating Database Items, Dictionary Items, or Descriptions*

► To create a check box

1. On the form with which you are working, choose Check Box from the Insert menu.
2. Position the control outline on the form, and then click.
3. To define the check box properties do one of the following:
 - Double-click the check box control
 - Choose the check box control and choose Properties from the Settings menu.



4. Complete the following field:
 - Title
5. Complete the following Value fields:
 - Value Checked/Unchecked
6. Click one or more of the following attributes:
 - Tab Stop
 - Group
 - Disabled
 - Visible
 - No display if currency is OFF
7. Click one of the following Justification options:
 - Left

- Right

Field	Explanation
Value Checked/Unchecked	Specifies the value used if this check box is selected. If a check box is associated with a database item, then the table is updated with this value. Otherwise, the value can only be used in an event rule.
Justification	Determines how the values entered at runtime will be justified within the edit box.
No display if currency is OFF	Hides this option if the Multicurrency flag = No in System Setup, General Accounting Constants (P0000).

Creating Radio Buttons

You use radio buttons to indicate choices. A filled radio button indicates that you have made a choice. If radio buttons are in a group box, they should always be mutually exclusive.

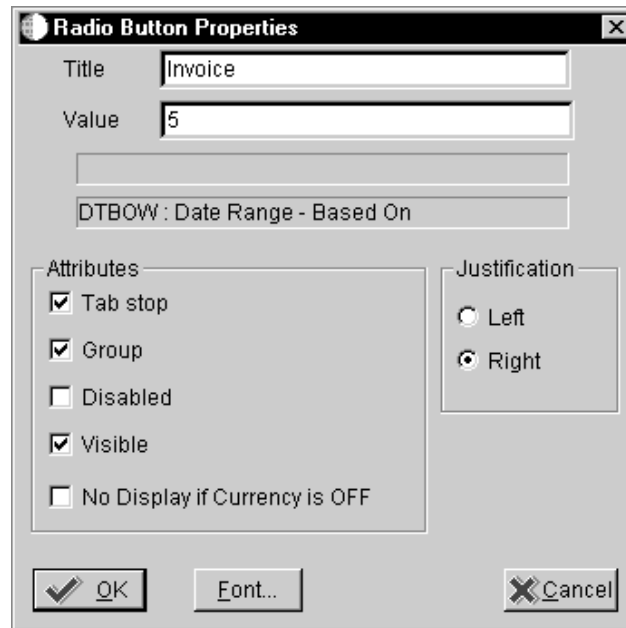
- You can associate radio buttons with a user defined code field, where each button has a value from the User Defined Code table.
- You associate a radio button with a database or dictionary item.

See Also

- *Associating Database Items, Dictionary Items, or Descriptions*

To create a radio button

1. On the form with which you are working, choose Radio Button from the Insert menu.
2. Position the control outline on the form and click.
3. Repeat steps 1 and 2 for as many radio buttons as you need, then select all of the radio buttons to group them and associate each of them to make them mutually exclusive.
4. To define the radio button properties do one of the following:
 - Double-click on the radio button control
 - Choose the radio button control and choose Properties from the Settings menu.



5. Complete the following fields:
 - Title
 - Value
6. Click one or more of the following Attributes:
 - Tab Stop
 - Group
 - Disabled
 - Visible
 - No display if currency is OFF
7. Click one of the following Justification options:
 - Left
 - Right

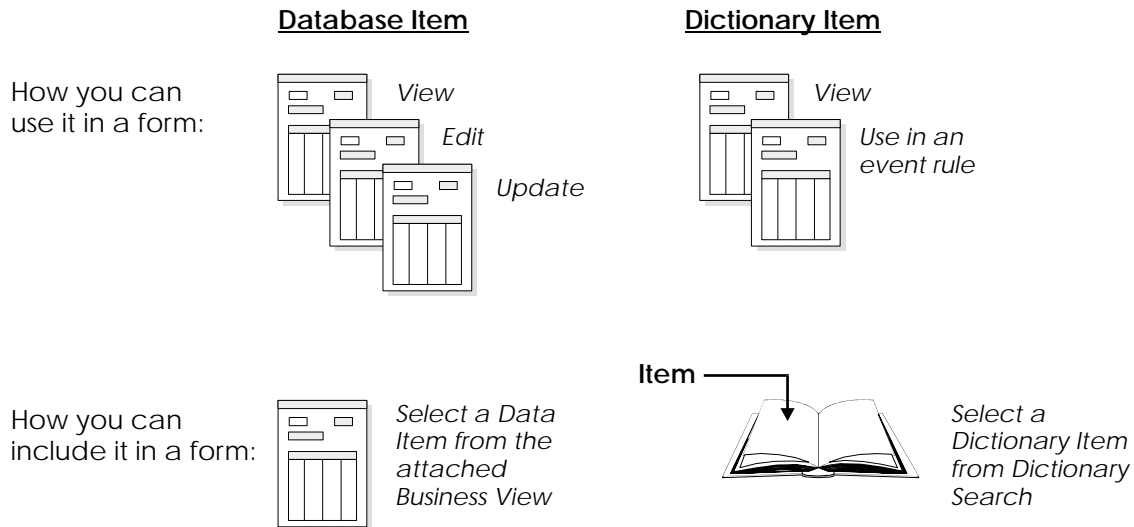
Field	Explanation
Value	Specifies a literal value that is used when the operator selects this radio button.

Adding Data Items to a Form

On a form, you can use any data item that is in the business view and any data item from the data dictionary. Database items are in the business view associated with the form. Data dictionary items are not in the business view.

Use a data dictionary item to store information not contained in a database field. Only database items are updated to the database. On a form, database items appear with a blue box in the left corner and dictionary items appear with a yellow box in the left corner.

The following illustration compares the use of database items and data dictionary items in an application.



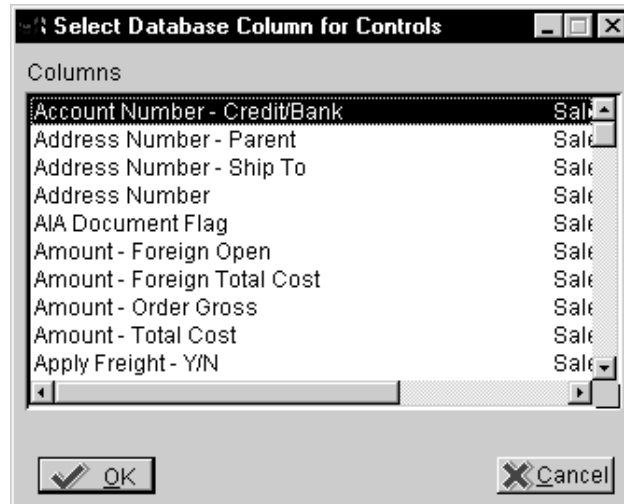
Inserting Database Items Onto a Form

You can insert any data item from the business view to include on the form as a database item. When you insert a database item, it always includes both:

- A static text control that displays the row or column description from the data dictionary
- The associated edit control or UDC edit control

► To insert database items onto a form

1. On the form with which you are working, choose Business View Field from the Insert menu.



2. Choose the database item.
3. Position the control outline on the form and click.

If the form has a grid, you can place a database item on a blank area of the form or in the grid. If you want to put the item in the grid, focus on the grid before you choose the database menu option.

The database control is marked with blue to distinguish it from a dictionary item, which is marked with yellow.

4. Continue to choose database items from the list. Click Cancel when you are finished.

See Also

- *Using Quick Form* in this guide

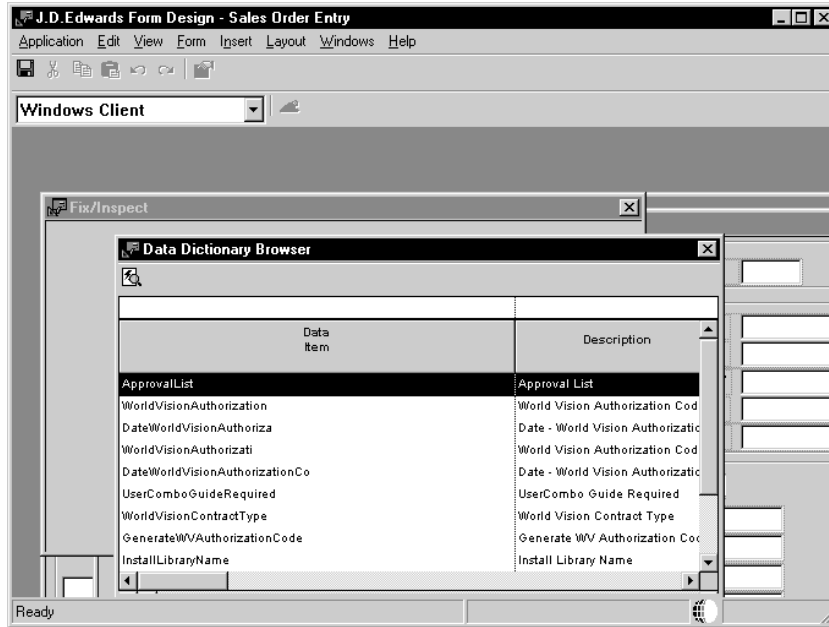
Inserting Data Dictionary Items Onto a Form

You insert a dictionary item when it is not already included in the business view for the form or when you do not want to update the field in a table. Data dictionary items appear with a yellow box in the left corner on a form.

Use data dictionary items as display-only fields in a form or as a reference for an event rule. Data dictionary items do not retrieve or update data in a table.

► To insert data dictionary items onto a form

1. On the form with which you are working, choose Data Dictionary Field from the Insert menu.



2. On Data Dictionary Browse Control, specify search criteria and press enter.
3. Choose a data dictionary item from the available items and drag it to your form.
4. Position the control outline on the form and click.

If the form has a grid, you can place a data dictionary item either on a blank area of the form or in the grid. If you put the item in the grid, focus on the grid before you choose the dictionary menu option.

The data dictionary control has a yellow box in the left corner to distinguish it from a database item, which has a blue box in the left corner.

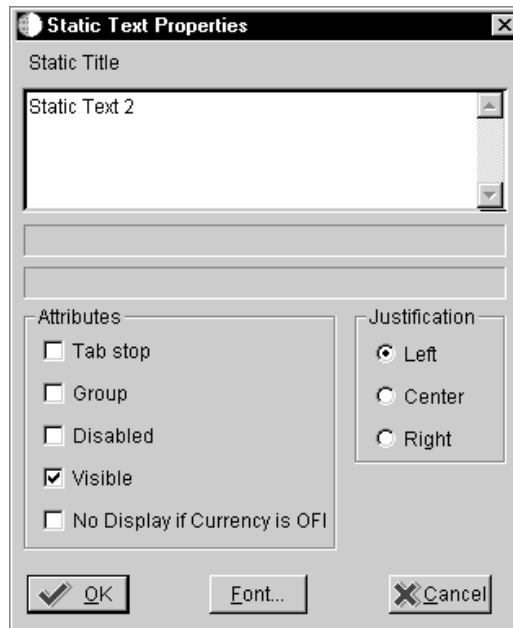
Creating Static Text Controls

Use a static text control to display descriptive text such as a titles or instructions. This text is not associated with a control, and the user cannot change it. It can be changed using the *Set Control Text* system function in event rules.

► To create a static text control

1. On the form with which you are working, choose Static Text from the Insert menu.
2. Position the control outline on the form and click.
3. To define the static text control properties do one of the following:

- Double-click the static text control.
- Choose the static text control and choose Properties from the Settings menu.



4. On Static Text Properties, complete the following field by typing in the text you want to appear on the form:
 - Title
5. Click one or more of the following Attributes:
 - Tab Stop
 - Group
 - Disabled
 - Visible
 - No display if currency is OFF
6. Click one of the following Justification options:
 - Left
 - Center
 - Right

Creating Edit Controls

Edit controls are generic input fields and have no associated text. You should associate edit controls with database or dictionary items.

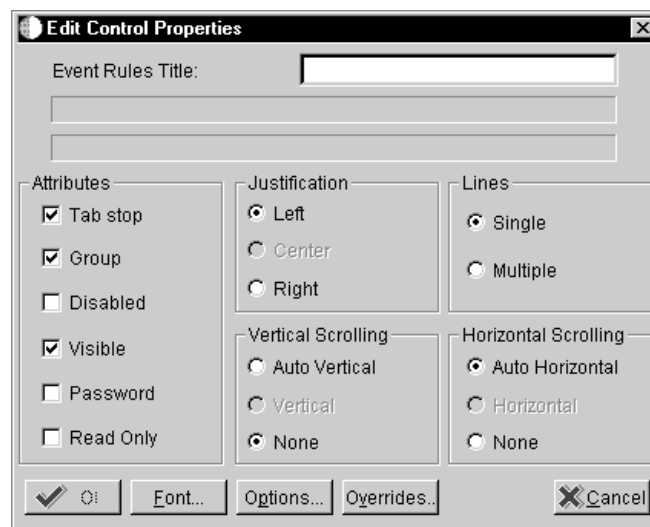
- If you associate an edit control with a database item, then the value entered by a user at runtime updates the table.
- If you associate an edit control with a dictionary item, then the value entered at runtime is for display only.
- Edit Controls have type ahead functionality. Once a user enters a character in the field, a history list is searched for a match. If there is a match, it is displayed in the field with highlighted text. This is particularly useful for data entry work because it can reduce the amount of typing required. The history list is stored in alphabetical order in a local file in the windows root directory. The user can define the length of the history list in the jde.ini file. Type ahead editing can also be enabled or disabled in User Preferences. Type ahead functionality is disabled for double-byte languages and multi-line edit controls.

Creating edit controls contains the following tasks:

- Creating an edit control
- Filtering database items

► To create an edit control

1. On the form with which you are working, choose Edit from the Insert menu.
2. Position the control outline on the form and click.
3. To define the edit control properties do one of the following:
 - Double-click on the edit control
 - Choose the edit control and choose Properties from the Settings menu.

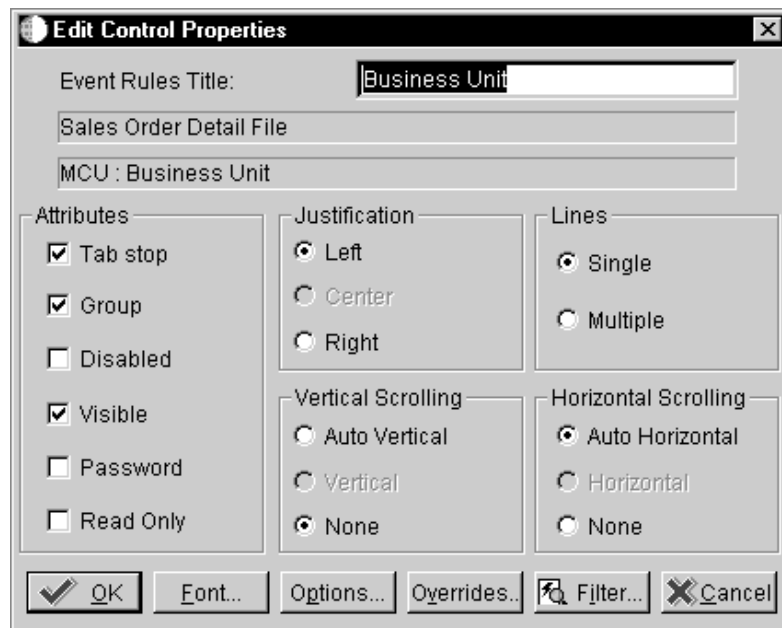


4. On Edit Control Properties, complete the following field:
 - Event Rules
5. Click one or more of the following Attributes:
 - Tab Stop
 - Group
 - Disabled
 - Visible
 - Password
 - Read Only
6. Click one of the following Justification options:
 - Left
 - Center
 - Right
7. Click one of the following Lines options:
 - Lines Single
 - Lines Multiple
8. Click one of the following Vertical Scrolling options:
 - Auto Vertical
 - Vertical
 - None
9. Click one of the following Horizontal Scrolling options:
 - Auto Horizontal
 - Horizontal
 - None
10. Click the following buttons to further alter the control:
 - Font
 - Options
 - Overrides

Field	Explanation
Event Rules Title	The default name of the control for use in an event rule. It defaults to the static text associated with an item. If there is no associated static text and you are planning to use this control in an event rule, enter a name.

Field	Explanation
Password	Allows you to enter an asterisk (*) so you don't see the actual password.
Read Only	Prevents a user from changing the value in a field.
Justification	Determines how the values entered at runtime will be justified within the edit box.
Lines Single	Designates the edit control to be for a single line only.
Lines Multiple	You can make an edit control multi-line only if both automatic scroll options are off.
Scroll Bar Horizontal/Vertical	<p>Displays the horizontal and/or vertical scroll bar.</p> <p>The Auto Vertical option scrolls the text vertically. The text scrolls up a full page when the user presses Enter on the last line of the edit text control. For this option to have any effect, you must also turn on the Multiple Lines radio button.</p> <p>The Auto Horizontal option scrolls the text horizontally. The text automatically scrolls ten characters to the right when the user types a character at the right edge of the edit text boundary. When the user presses Enter, the text scrolls back to the zero position.</p>

If you have associated a database item with the edit control, you will have an additional filtering option available.



► **To filter database items**

1. On Edit Control Properties, click Filter.



2. On Edit Control, choose one of the Edit Control Filter options.
3. Turn on the Wildcard Display if desired. The Wildcard Display displays an asterisk.

As you add controls to the form, you can tell the runtime engine how to filter the incoming records from the database. For example, if you have two controls on your Find/Browse form that you want to filter on, the resulting SQL statement generated will be an “AND” condition for each condition. For example, if you have Search Type and Alpha Description as the controls on the form, the filter criteria for Search Description should be “>=” and the Search Type control should be “=”. If a user types in “D” and puts a “V” in the Search Type field the resulting SQL statement looks like the following:

```
SELECT * FROM F0101 WHERE (ABAT1 = "V" AND ABDC LIKE "D%") ORDER BY ABAN8 ASC
```

Another situation where you might want to use filter fields is when records need to fall between two values. In this case you use a Range Filter. For example, in distribution a status is assigned to each line of the Order. One status is the current status and the other status is the Next Status. In this example you would filter records greater than or equal to the Present Status and less than or equal to the Next Status. You would drop one, filter, and then drop the next one.

Creating UDC Edit Controls

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 data dictionary item.

The visual assist flashlight automatically appears adjacent to the UDC edit control field. When you click 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.

Creating UDC edit controls contains the following topics:

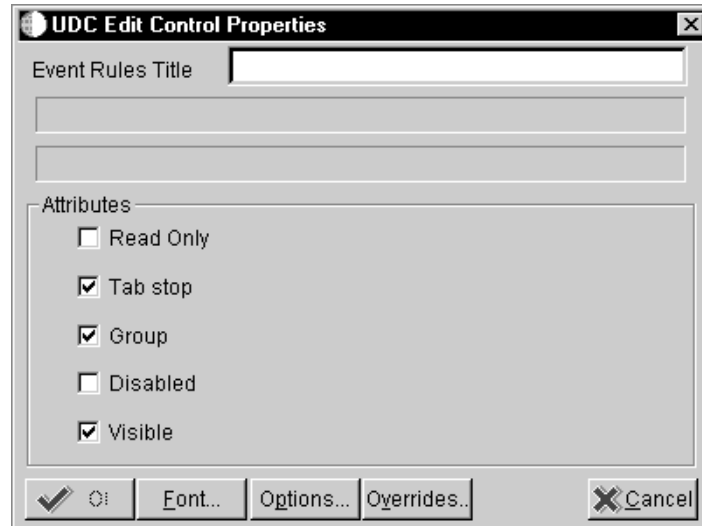
- Creating a UDC edit control
- Disconnecting static text from edit controls

Creating a UDC Edit Control

Use the following process to create a UDC edit control.

 **To create a UDC edit control**

1. On the form with which you are working, choose UDC Edit from the Insert menu.
2. Position the control outline on the form and click.
3. To define the UDC edit control properties do one of the following:
 - Double-click the UDC edit control
 - Choose the UDC edit control and choose Properties from the Settings menu.



4. Complete the following field:
 - Event Rules Title
5. Click any of the following options to alter the behavior of this control:
 - Read Only
 - Tab Stop
 - Group
 - Disabled
 - Visible
6. Click one of the following to further edit the control:
 - Font
 - Options
 - Overrides

See Also

- *Associating Database Items, Dictionary Items, or Descriptions*

Disconnecting Static Text from Controls

You can disconnect the static text from an edit control or a UDC edit control so you can move the static text and edit/UDC control separately or to delete either the text or the control.

If you disconnect the static text of the data item for the edit box, and then delete the static text from the form it will have the same effect as associating an edit control with a database item.

In some cases, the text or description may not be necessary. For example, in Purchase Order Entry, the order type follows the order number, but the description “Order Type” is not necessary.

► **To disconnect static text from an edit control or UDC control**

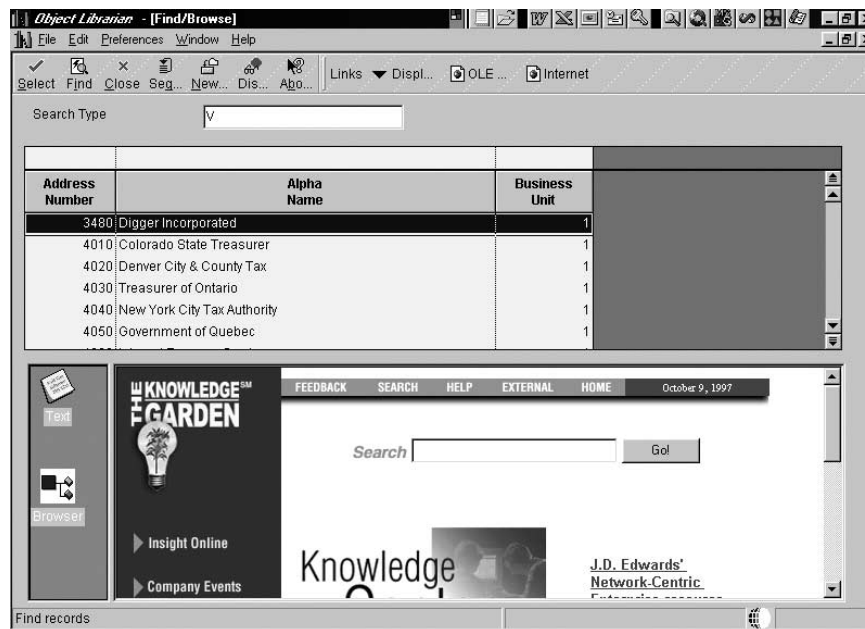
1. On the form with which you are working, choose the item for which you want to disconnect the static text from the control.
2. From the Edit menu, choose Disconnect.
3. Now you can move the static text and edit/UDC box separately. You can use the Cut or Clear option to delete either one.

Creating Media Object Controls

The media object control is a very specialized control. You can use this control to allow the user to enter text or attach objects. You can place this control on any form type except the message box.

You can use the media object control in a variety of ways. You can add images, shortcuts to other applications, and text. You can add multiple media objects to a form. You can add multiple text objects to a single media object. You can also add generic files or URLs.

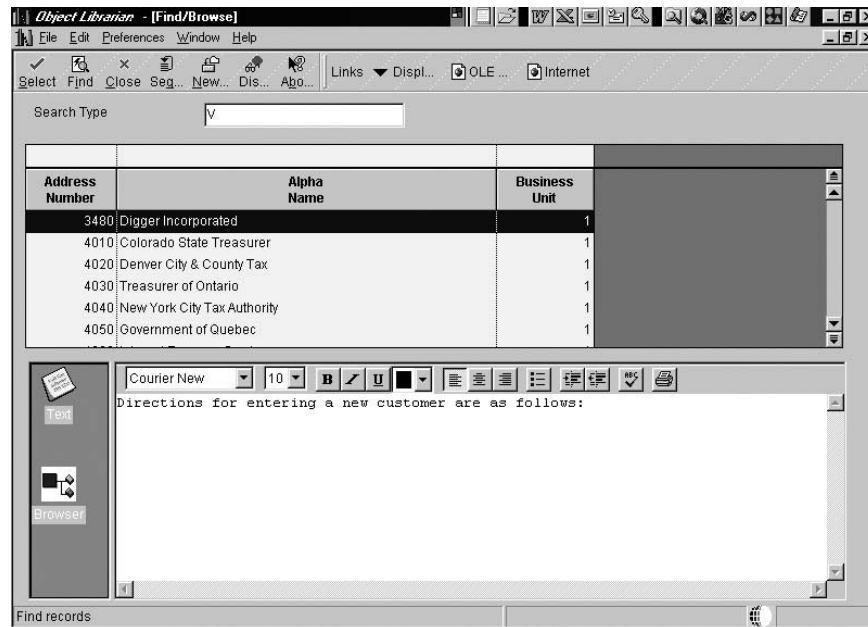
If you want to display a certain file available on the Internet, you can attach the media object control to the form and have a link to the Internet.



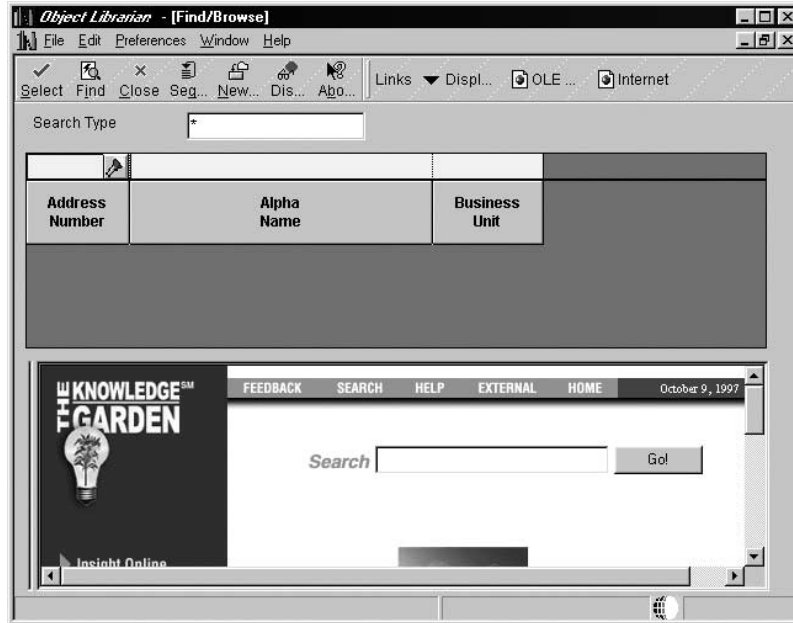
After you define your media object queue for the Internet (for example including a valid HTTP address), you can use the Start Web Browser system function to open the control with the Internet file displayed.

A specific example of how you might use this is if you needed to check a shipping vendor's Web page to track the status of shipments. You can look up the shipment directly within the media object control.

You can also use the Text feature of the media object control. For example, you can use the media object control to display instructions specific to a particular form.



You can use the Hide Splitter Bar system function if you do not want your employees to have access to other media object functions.

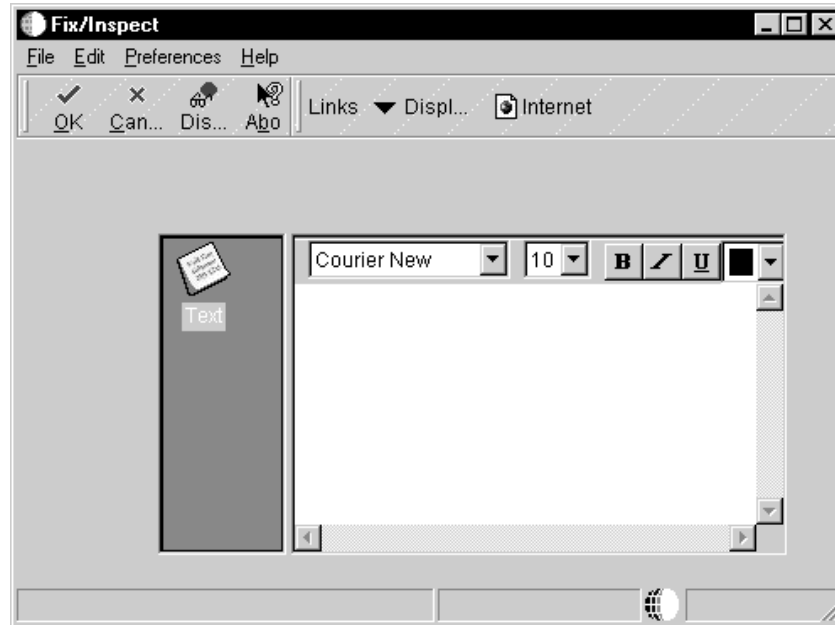


Another way you can use the media object control is to display employee queues where messages are sent. A media object control is laid next to the tree control on a Parent/Child form. Next, event rules are used to establish a relationship between the tree side and the media object side. For example if a message is highlighted in the tree, then the corresponding message text is displayed in the control.

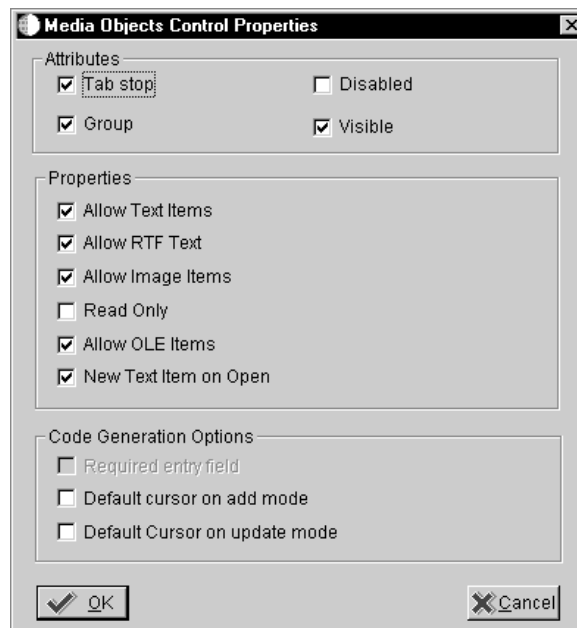
► To create a media object control

1. On the form with which you are working, choose Media Object from the Insert menu.
2. Position the control outline on the form and click.

Be sure to size the control so it is large enough to show the entire control. The control looks like the following example when it is sized. If you do not make the control large enough, it will not show everything that is part of the control.



3. To define the media object control properties do one of the following:
 - Double-click the media object control.
 - Choose the media object control and choose Properties from the Edit menu.



4. On Media Objects Control Properties, click one or more of the following Attributes:
 - Tab Stop
 - Group

- Disabled
 - Visible
5. Click one or more of the following Properties:
- Allow Text Items
 - Allow RTF Text
 - Allow Image Items
 - Read Only
 - Allow OLE Items
 - New Text Item on Open
6. Click one or more of the following Code Generation Options:
- Required entry field
 - Default cursor on add mode
 - Default cursor on update mode

Field	Explanation
Tab Stop	Allows the user to tab into this control.
Group	Defines this control as the first in a series of related controls. A group is defined as this control plus all subsequent controls without the Group option turned on. The next control with Group turned on signals the start of a new group (a single control can form a group). Within a group of more than one control, use the arrow keys to move from one control to another instead of pressing Tab.
Disabled	Disables a control. A disabled control is grayed out at runtime.
Visible	Makes a control visible.
Allow Text Items	Allows text items to be added to the Media Object control.
Allow RTF Text	Allows Rich Text Format text to be added to the Media Object control.
Allow Image Items	Allows images to be placed on the Media Object control.
Read Only	Prevents a user from changing the value in a field.
Allow OLE Items	Allows OLE objects to be placed on the Media Object control.
New Text Item on Open	Adds a text item to the Media Object control when the control is first opened.
Required entry field	Prevents the user from leaving this field blank.

Field	Explanation
Default cursor on add mode	Designates this field as the initial position of the cursor in Add mode.
Default cursor on update mode	Designates this field as the initial position of the cursor in Update Mode.

You can use several applications to help you determine which media objects you may want to use. Use P00166 to add descriptions for your media objects. You can then search for media objects based on their characterization. Use P00167 to set up which characterization categories are used for a particular GT structure. Use P98MOQUE to set up media object queues.

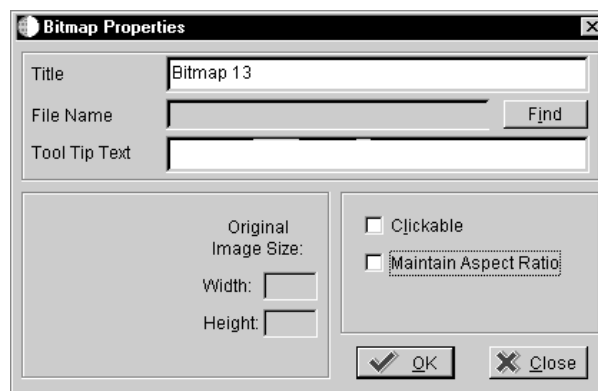
Creating Bitmap Controls

You can create a control that looks like a picture or other art by using a bitmap control. You can then attach event rule logic to the control. For example, you can attach logic to the button clicked event on the control so that when a user clicks the control, the application will automatically link to a different form.

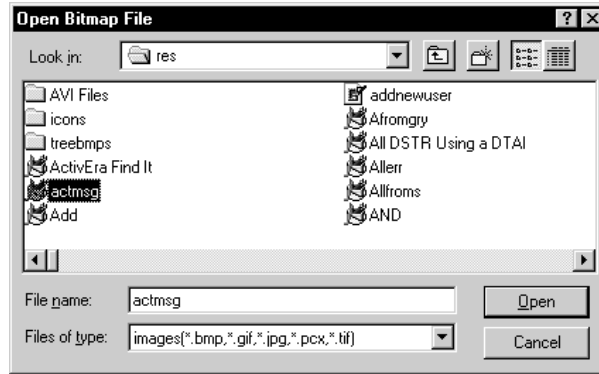
You can also use the bitmap control for an animated gif instead of bitmap. This is particularly useful for Java and HTML applications. The animated gif is animated in Java and HTML applications, however it does not appear animated in Windows applications and only displays the first image of the animated gif file.

▶ To create a bitmap control

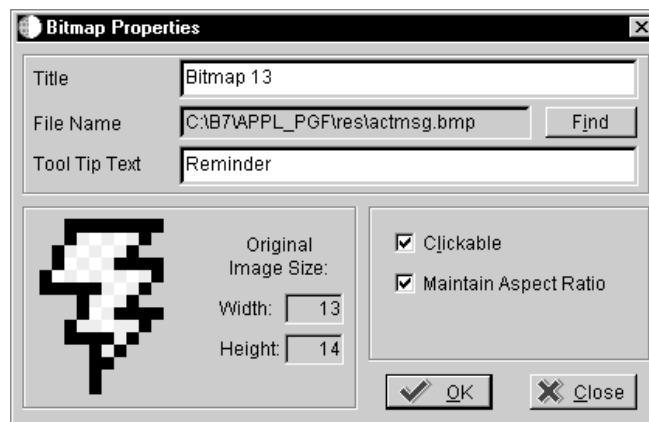
1. On the form with which you are working, choose Bitmap from the Insert menu.
2. Position the control outline on the form and click.



3. On Bitmap Properties, click Find to find the file you wish to use for your bitmap control.



The Bitmap Properties form displays the bitmap you have chosen.



4. Complete the following field:

- Tool Tip Text

This text is like Hover Helps appears if the user makes the mouse pointer hover over the bitmap.

5. Click one or more of the following attributes:

- Clickable

If you turn on the Clickable option, a Button Clicked event is enabled for the bitmap.

- Maintain Aspect Ratio

If you turn on the Maintain Aspect Ratio option, the bitmap will maintain its dimensions if you resize it. This ensures that your image does not become distorted if you resize it.

Field	Explanation
Clickable	Enables the button to perform an action when it is clicked. If you check this option you must have a Button Clicked event enabled for the bitmap.
Wallpaper	Specifies the wallpaper that is displayed.
Title	The title for the form, field, control, or button. <ul style="list-style-type: none"> For a form: This will appear on the form at runtime. For a field, control, or button: If the title is longer than the size of the control or button, it will be truncated. Resize the control to show the complete title.

Creating a Tree Control

You can use the tree control on any form type. Since the tree control is not limited to a particular business view and does not have any default database functionality, you have more flexibility loading tree node data. There are also tree control system functions available to further customize tree control functionality.

You can use Tree Control system functions to add logic to your control. You can use these system functions for actions like contracting and expanding nodes or setting bitmaps for nodes. Refer to the *Online APIs* for more information about these system functions.

► To create a tree control

1. On the form with which you are working, choose Tree Control from the Insert menu.
2. Position the control outline on the form and click.

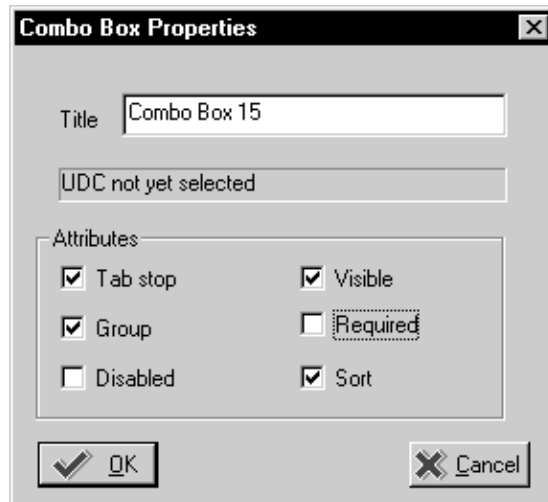
Size the control if needed.

Creating Combo Boxes

You can use a combo (list) box to display a list of items to choose from. The combo box has type-ahead functionality. You can associate the combo box with a data dictionary item so that it preloads with UDC values. When you create a combo box be sure to use a data dictionary item that has the value you want and use meaningful UDCs. You can also use event rule logic in your application to load the values.

► **To create a combo box control**

1. On the form with which you are working, choose Combo Box from the Insert menu.
2. Position the control outline on the form and click.

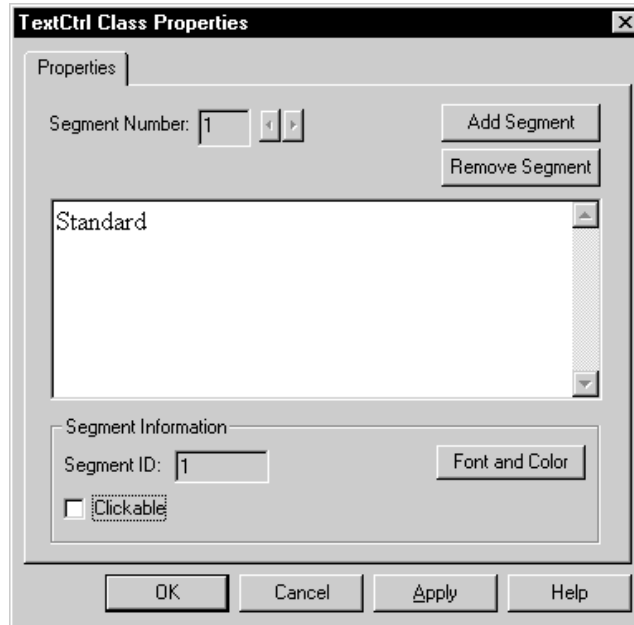


Creating Text Boxes

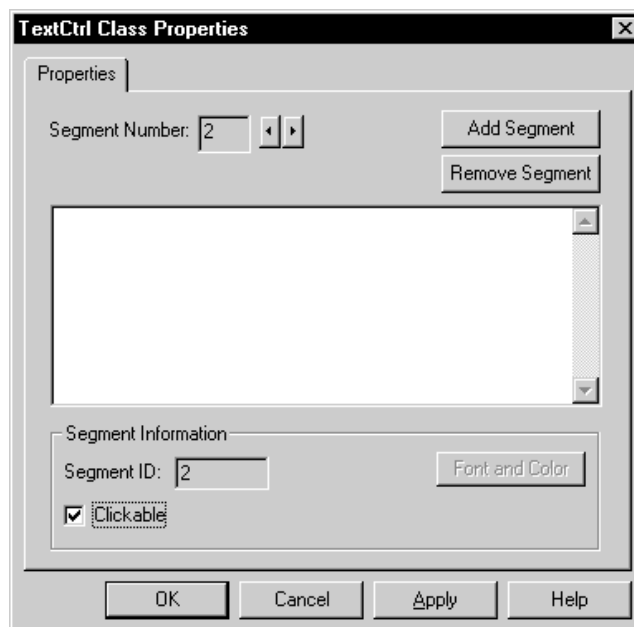
You can use a text box control to create different text segments and then attach attributes to them. You can format the text segments differently so that each segment looks different. For example you can create a clickable text segment and add event rules to the *Click* event so that you can click the text to connect to a different form. There are also several system functions available for use with this control. The text box control is particularly useful for web applications.

► **To create a text box control**

1. On the form with which you are working, choose Text Box from the Insert menu.
2. Position the control outline on the form and click.



3. Type in the text you wish to use.



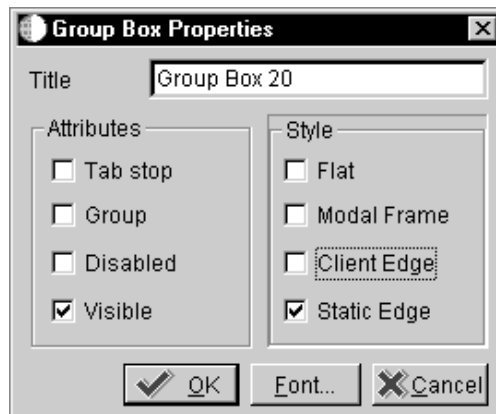
4. Click Add Segment to add additional text segments.

Creating Group Boxes

A group box visually categorizes and defines a group of fields by surrounding the controls and providing an optional title to the group. You can display the group box in different styles.

► **To create a group box**

1. On the form with which you are working, choose the controls to be included in the group box by drawing a box around the fields.
2. From the Insert menu, choose Group Box.
3. To define the group box properties do one of the following:
 - Double-click the group box
 - Choose the group box and choose Properties from the Settings menu.



4. Complete the following field:

- Title

Delete the title from the Title field if you do not want a title to appear on the form.

5. Click one or more of the following Attribute selections:

- Tab Stop
- Group
- Disabled
- Visible

6. Click one or more of the following Style selections:

Flat The box has an older Windows flat style border.

Modal Frame The box has a double border.

Client Edge The box has a border with a sunken edge.

Static Edge

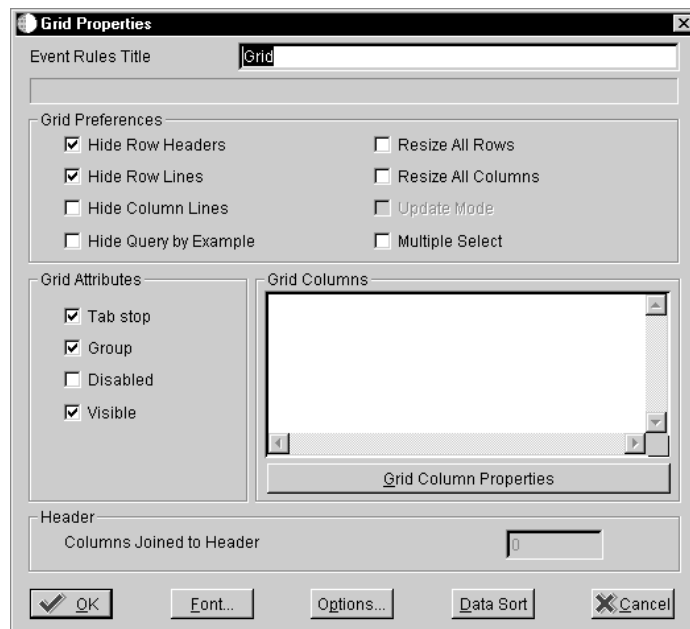
The box has a three-dimensional border.

Defining Grid Properties

You can define grid properties such as the font or sort order for a grid.

► To define grid properties

1. On the form with which you are working, to define the grid control properties do one of the following:
 - Double-click on the grid
 - Choose the grid control and choose Item Properties from the Edit menu.



2. Click any of the following Grid Preferences:
 - Hide Row Headers

The row header must be displayed when media objects are used. This is the gray column to the left of the grid row where the paper clip icon displays.

- Hide Row Lines

This is standard for Find/Browse forms.

- Hide Column Lines
- Hide Query By Example
- Resize All Rows
- Resize All Columns
- Update Mode
- Multiple Select

This option must be turned on in order to make a form multiselect, for example to allow multiple grid rows to be selected for an operation. The property event rule for the button (for example Select or Delete) must have the Repeat Event Rule for Grid option turned on.

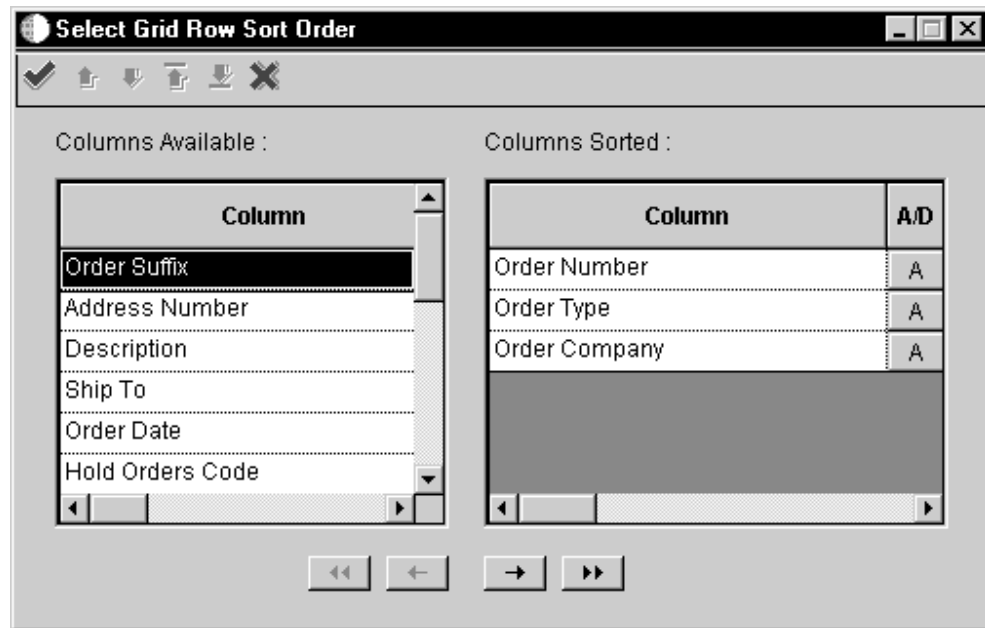
3. Click any of the following Attributes:
 - Tab Stop
 - Group
 - Disabled
 - Visible
4. To define properties for a selected column, choose an item from the Grid Columns list and click Select.
5. Click any of the following to specify additional grid properties:
 - Font
 - Options

For information about grid options refer to *Defining Options For a Grid*.

- Data Sort

If you click Data Sort, the following form appears and you can choose the way you want your data sorted.

Choose an Available Column and click the right arrow button to sort on that column. After the column appears in Columns Sorted, choose it and right mouse click on the A/D (ascending/descending) column to toggle to either an ascending or descending sort order for that column. The columns are sorted starting with the top one under Columns Sorted. You can sort on several columns if needed. Use the arrows in the menu to change the order of your sorted columns.



Field	Explanation
Hide Row Numbers	Hides the horizontal row headings.
Hide Row Lines	Hides the horizontal row lines in the grid.
Hide Column Lines	Hides the vertical column lines in the grid.
Hide Query By Example	Suppresses the QBE functionality by hiding the QBE line. Do this only if the grid does not contain any fields from the table, that is, there is nothing on which to query.
Resize All Rows	Enables the user to enlarge the grid rows.
Resize All Columns	Enables the user to widen the grid columns.
Update Mode	Defines the grid to be input capable. This only works on the header/detail and headerless detail forms. This option has no effect on the find/browse form because that form type is incapable of input.
Multiple Select	Allows you to select more than one row in the grid. When you click on Select, Event Rule logic is performed on the first selected row, then the second selected row, and so on. There is no default behavior on Select.
Grid Column List	Displays columns in the grid.
Select Grid Column	Displays grid column properties for a selected column.
Number of columns joined to	Used for backwards compatibility. All new forms should have a zero in this field.

See Also

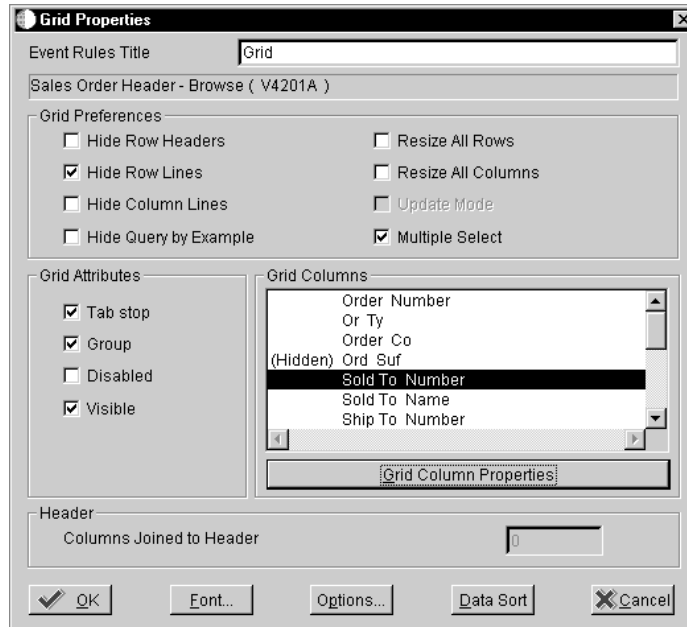
- The *OneWorld Foundation* guide for how to customize the grid at runtime

Defining Grid Column Properties

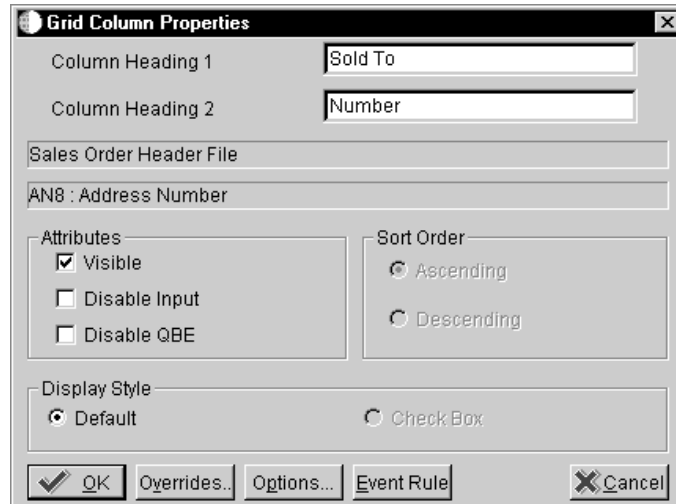
You can set properties for each grid column. On Find/Browse forms, you cannot attach event rules to grid columns.

► To define grid column properties

1. On Grid Properties, choose the grid column for which you wish to define properties.



2. Click the Grid Column Properties button.



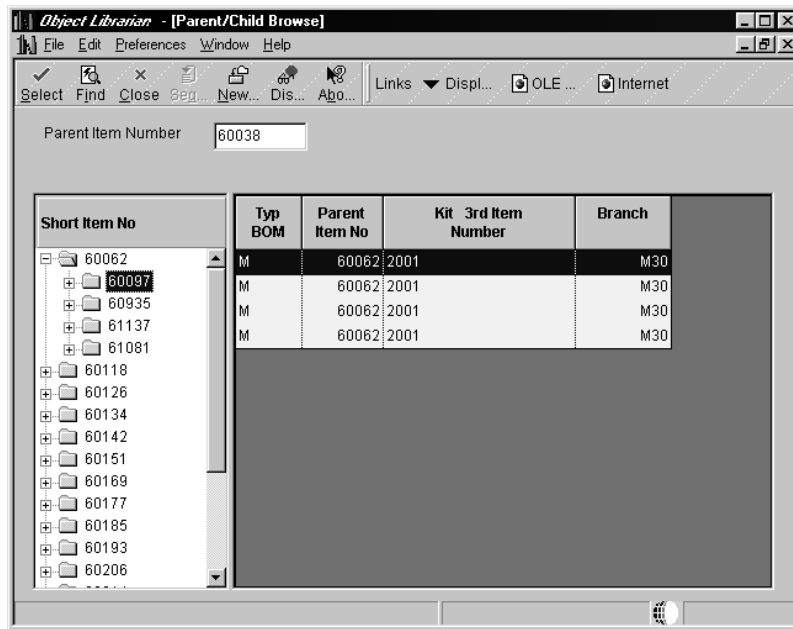
3. Complete the following fields:
 - Column Heading 1
 - Column Heading 2
4. Click one or more of the following attributes:
 - Visible
 - Disable input
 - Disable QBE
5. Click one of the following options to indicate the sort order:
6. Click one of the following display styles:
 - Default
 - Check Box (Check Box is currently unavailable)
7. Click the following to define additional grid properties:
 - Overrides
 - Options
 - Event Rules

Field	Explanation
Column Heading 1/Column Heading 2	Specifies the text for the first and second lines of the column heading.
Visible	Makes a control visible.
Disabled	Disables a control. A disabled control is grayed out at runtime.
Disable QBE	Prevents a query for the grid column.

Field	Explanation
Ascending/Descending	Indicates the ascending or descending sort order for this column.
Display Style-Default or Check Box	Designates whether the data item is displayed only in the grid or available as a check box.

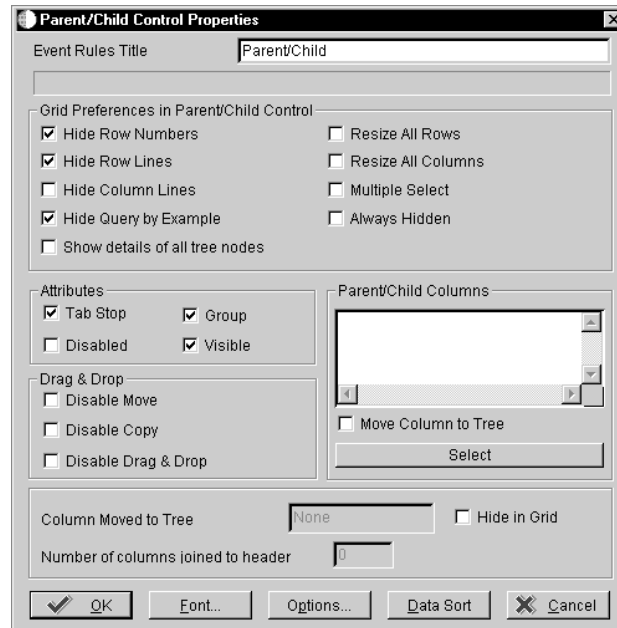
Defining Properties for Parent/Child Controls

The Parent/Child form has a special parent/child control that automatically appears with the form. The control is a composite control with a tree control on the left side and a grid control on the right. The tree and grid control portions are both tied to the same business view. This control is only available on the Parent/Child form. Refer to *Understanding Form Types* for more information about the Parent/Child form. The following graphic shows a Parent/Child form.



► To define properties for the parent/child control

- On the form with which you are working, perform one of the following:
 - Double-click on the grid portion of the parent/child control.
 - Right-click the parent/child control and choose Properties.



2. Click any of the following Grid Preferences:
 - Hide Row Numbers
 - Hide Row Lines
 - Hide Column Lines
 - Hide Query By Example
 - Show details of all tree nodes
 - Resize All Rows
 - Resize All Columns
 - Multiple Select
 - Always Hidden
3. Click any of the following Attributes:
 - Tab Stop
 - Disabled
 - Group
 - Visible
4. Click any of the following Drag & Drop options:
 - Disable Move
 - Disable Copy
 - Disable Drag & Drop
5. To define properties for a selected column, choose an item from the Parent/Child columns list and click Select.

6. Complete the following field:
 - Number of columns joined to
7. Click any of the following buttons to specify additional properties:
 - Font
 - Options
 - Data Sort

You can also double click on the grid to display the properties for the grid. Choose the grid column you wish to display in the tree within the scroll window and select the check box to move it to the tree side of the control. This hides the chosen column from the right side of the grid (default is to hide unless checkbox is unchecked) and displays it only in the left side of the control or the tree. This also moves the column description over.

You can select multiple objects; however, they must be at the same node level. There are two modes available. In the first mode, there is a one-to-one correspondence between the tree and the grid. The other mode shows all details of the tree. You can also load trees pre expanded. This option is on the right mouse menu.

After the above steps you set up either a parent/child relationship or use event rules to load child nodes to the tree. The method you use depends on whether your table has an inherent parent/child relationship. In the following examples the first example describes a situation where there is an inherent parent/child relationship in a table. The second example describes a situation where the data does not have an inherent parent/child relationship by keys, but can be displayed in a hierarchical manner for readability.

You can use event rules and system functions to customize the way your parent child control functions. For example you can change the top node of a tree or change which node is the first child on a tree.

Field	Explanation
Event Rules Title	The default name of the control for use in an event rule. The event rules title defaults to the static text associated with an item. If there is no associated static text and you want to use this control in an event rule, enter a name.
Hide Row Numbers	Hides the horizontal row headings.
Hide Row Lines	Hides the horizontal row lines in the grid.
Hide Column Lines	Hides the vertical column lines in the grid.
Hide Query By Example	Suppresses the QBE functionality. Do this only if the grid does not contain any fields from the table, that is, there is nothing on which to query.

Field	Explanation
Show details of all tree nodes	
Resize All Rows	Enables the user to enlarge the grid rows.
Resize All Columns	Enables the user to widen the grid columns.
Multiple Select	Allows you to select more than one row in the grid. When you click on Select, Event Rule logic is performed on the first selected row, then the second selected row, and so on. There is no default behavior on Select.
Always Hidden	Hides the grid control on the Parent/Child Form.
Disable Move	Disables the Move Here option on the popup menu at the end of a drag and drop using the right mouse button.
Disable Copy	Disables the Copy Here option on the popup menu at the end of a drag and drop using the right mouse button.
Disable Drag and Drop	Prevents a user from performing a drag and drop operation in the tree view of the control.
Parent/Child Columns	
Move Column to Tree	Displays the text for a column as the text of the corresponding tree node.
Hide in Grid	Within the Parent/Child Properties form, this option determines which columns in the related business view you want to hide or display in the detail area. When you designate the Move Column to Tree option, the Hide in Grid option is automatically selected. The default is to display all columns.
Number of Columns Joined to Header	Used for backward compatibility. All new forms should have a zero in this field.

Example: Inherent Parent/Child Relationship

This example describes a case in which there is an inherent parent/child relationship in a table. For instance, in the Bill of Materials table (F3002), IXKIT is the short item number for the kit. Kits are composed of multiple items within IXKIT, represented by IXITM, the short item number for the item.

Kit item 60038 includes items 60062, 60118, 60126, and other items as its children.

<i>IXTBM</i>	<i>IXKIT (Parent)</i>	<i>IXITM (Child)</i>	<i>IXKITA</i>	<i>IXMMCUCU</i>	<i>IXAITM</i>
<i>M</i>	60038	60062	220	<i>M30</i>	2001
<i>M</i>	60038	60118	220	<i>M30</i>	2006
<i>M</i>	60038	60126	220	<i>M30</i>	2007

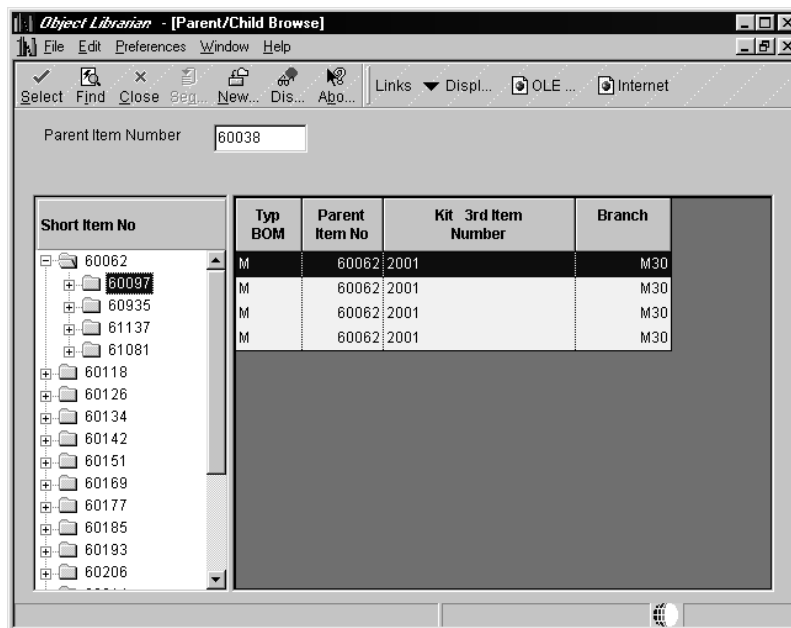
Each one of the items (IXITM) can be a kit within itself.

Item 60062 includes 60097, 60935, 61137, and other items as its children.

<i>IXTBM</i>	<i>IXKIT (Parent)</i>	<i>IXITM (Child)</i>	<i>IXKITA</i>	<i>IXMMCU</i>	<i>IXAITM</i>
M	60062	60097	2001	M30	2004
M	60062	60935	2001	M30	9011
M	60062	61137	2001	M30	9031

Because this inherent relationship exists within the table, you can develop a parent/child application over the Bill of Materials table F3002 (using V3002A business view).

The following form represents this data.



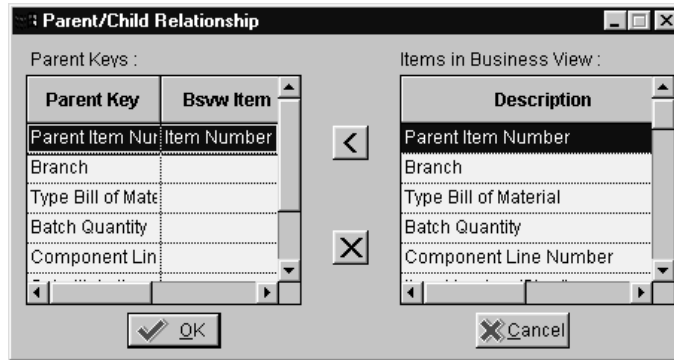
This structure represents data from the same table shown in the parent/child format. When you first inquire on the 60038 item in the filter field you generate the same SQL statement as you would on a normal Find/Browse form.

When you click on the + node next to item 60062, the runtime engine takes the value of the Item Number and moves it into the Parent Item Number, making it the parent for the fetch.

To set up the parent/child relationship to be used when the runtime engine loads nodes to the tree:

- Focus on the parent/child control.
- From the Form menu, choose Parent/Child Relation.

The following form appears:



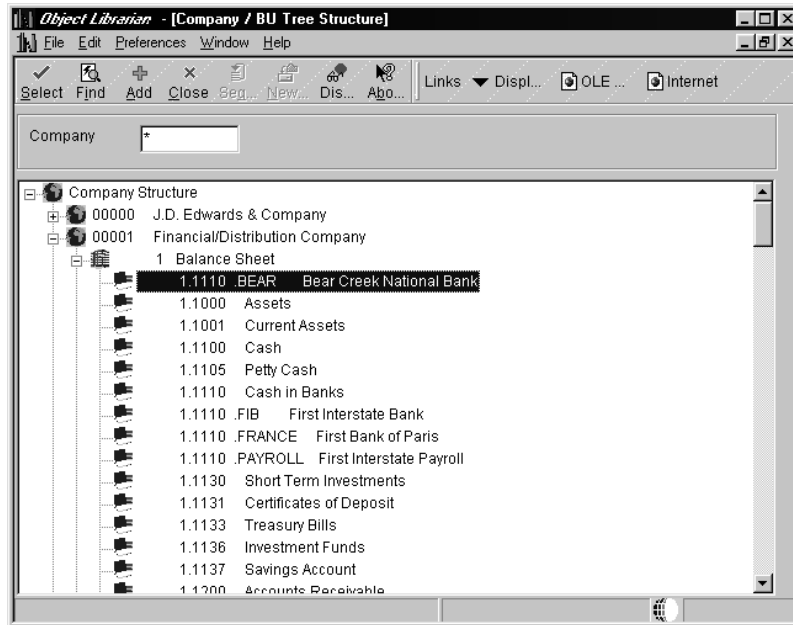
- Indicate which child item should be used as the parent for loading the next level nodes.

A Parent/Child form is an extension of the Find/Browse form, where the fetches are done by the tool using different keys to access its data. Some things to consider when you create a Parent/Child form include:

- You usually use the parent field for the filter.
- You usually use the child for the left side of the parent/child control.
- The data item type between the parent and the child must be compatible.
- You must establish the parent/child relationship correctly. You usually move the child item into the parent field.
- Subsequent (RESET) fetches are done by the tool only when a node is clicked. The tool takes the value of the child at that node and substitutes it into the parent field for the fetch. Once a fetch is performed for that node, that information is saved in a runtime structure. If the same node is clicked again, the data is not fetched from the database, but instead from the runtime structure.

Example: No Inherent Parent/Child Relationship

This example describes a situation where there is no inherent parent/child relationship. In this situation, the runtime engine does not automatically fetch records because the parent/child relationship is not inherent, and the relationship is between data items of different types. In this case, the data item you chose to be in the left side of the parent/child control is usually a string (alphanumeric) variable that holds any data type.



The parent/child relationship is not inherent to a particular table because the application is accessing multiple tables, such as Company Master, Account Master, and Chart of Accounts, to get information.

Page-at-a-Time Processing

Page-at-a-time processing ensures that each fetch only fetches one page of data. Page-at-a-time processing is the default mode for all parent/child forms.

During page-at-a-time processing in standard mode, the page size is defined as the number of nodes that can fit in the current view. When the Find process starts only one page of the first level nodes is fetched from the table and inserted in the tree. When the user scrolls down a new page of data is fetched.

Page-at-a-time processing for expand-all style Parent/Child forms is similar to that for standard mode. When the Find process starts, one page of first level nodes are fetched from the table and inserted into the tree. Then all of the first level nodes are expanded. Each expansion fetches only one page of data. Because the tree expands exponentially in expand-all mode, a tree that is very deep may impact performance.

Tree Nodes

When the parent and child nodes come from different tables or are of different data types, the parent/child relationship is not automatically set up. When this is the case the runtime engine does not automatically fetch the child database records because it does not know which table to retrieve them from.

Parent Nodes

If possible, use the runtime engine to load the initial set of parent nodes to the tree for you. You do this by using the based-on view, which is a view over the upper most node's table. This way a parent filter can still be used in the control, and the runtime engine will load the first level nodes to the tree. If you cannot do this, you must insert the first level nodes yourself. To do this, you usually use Table I/O on the *Button is Clicked* event of the Find button. You use the same methods as you do to insert child nodes. Use a *Suppress Find* system function to stop the runtime engine from attempting to load any nodes.

Child Nodes

Whenever a node is expanded, the system function *Suppress Fetch on Node Expand* is called from the event *Tree Node Is Expanded*. This tells the runtime engine not to do any fetches because event rules will handle the loading of the child nodes. *Tree Node Is Expanded* is the main event of the application. This event occurs when the tree node is expanding (for example, the plus next to a child node is expanded for the first time). You place event rules on this event to read the next records to be loaded to the tree as children of the expanded node. You can use table I/O or business functions to retrieve these records. Often the children may be coming from different tables based on the type of parent node that is expanded. If possible, you should do a SELECT and then use the FETCH NEXT command to retrieve records in a DO WHILE loop. The GB runtime data structure should be populated with data from the records read in the loop, and then an INSERT GRID BUFFER ROW system function should be called. This parent child system function is different than the INSERT GRID BUFFER ROW in the normal GRID section. At this point you also have the ability to set your custom tree bitmaps using the SET TREE NODE BITMAP system function.

Example

In the following example, event rules are attached to an application on the *Tree Node is Expanding* event:

```

Suppress Fetch on Node Expand(FC Parent/Child)
//
// Here are the variables to get out of the account and the business unit
loop
// being initialized.
VA frm_OutOfLoop = "0"
VA frm_ExistAcctLoop = "0"
//
// If Loop looking at the GC Business Unit Field.
If GC BusinessUnit is equal to <Blank>
//
// Select the F0006 differently if there is one company or all companies
//
F0006.Open
If VA frm_AllCompanies is equal to "1"
VA frm_CurCompany = GC Co
F0006.Select
Else
VA frm_CurCompany = BC Company
F0006.Select
End If
//
// While Loop which fetches all the all the business unit for a specific
company.
// If the company changes we get out of the loop.
While VA frm_OutOfLoop is equal to <Zero>
// Fetch the records from the F0006 Table.
F0006.FetchNext
GB BusinessUnit = GB Companies
If SV File_IO_Status          is equal to CO SUCCESS

GB Co = BC Company
VA frm_PreCompany = BC Company
VA frm_ConcateBuDesc = " "
VA frm_ConcateBuDesc = concat([VA frm_ConcateBuDesc],[VA frm_NameOfBU])
GB CompanyStructure = concat([GB Companies],[VA frm_ConcateBuDesc])
GB Companies = concat([GB Companies],[VA frm_ConcateBuDesc])
//
// Tells the Level of the Tree Structure.
GB LevelOfTreeInt = "1"
//
Insert Grid Buffer Row(FC Parent/Child, <After Last Row>, <Yes>, <No>, <No>,
<No>, <No>, <Yes>)
Set Tree Node Bitmap(FC Parent/Child, <Last Grid Row>, BussUnit.bmp, <Yes>)
//
//
If VA frm_PreCompany is not equal to VA frm_CurCompany
VA frm_OutOfLoop = "1"
End If
Else
VA frm_OutOfLoop = "1"
End If
End While
F0006.Close
Else
// Loop thru the F0901 to get the corresponding accounts.
//
VA frm_CURBU = GC BusinessUnit
F0901.Open
F0901.Select
If SV File_IO_Status          is equal to CO ERROR

//
End If

```

```

GC Companies = " "
GB Companies = " "
Business Unit, Object, Subsidiary Merge
GB Companies = concat([VA frm_DBANI],[VA frm_AcctDesc])
GB CompanyStructure = concat([VA frm_DBANI],[VA frm_AcctDesc])
GC Companies = VA frm_AIDF0901
//
// Tells the level of the Tree Structure '2'
GB LevelOfTreeInt = "2"
//
Insert Grid Buffer Row(FC Parent/Child, <After Last Row>, <Yes>, <No>,
<No>, <No>, <No>, <No>)
Set Tree Node Bitmap(FC Parent/Child, <Last Grid Row>, accounts.bmp, <Yes>)
End If
End While
End If
FC BUFrom = " "

```

Dragging and Dropping

When a Parent/Child form is created, both move and copy drag-and-drop operations are enabled. You can turn these options off in the properties for the form. If drag-and-drop is turned off and a user attempts to drag a node, the cursor will indicate that dragging is not allowed. None of the drag events will execute. You can control operations to the database using the drag-and-drop events:

- *Begin Drag* - If drag-and-drop is allowed then on the Begin Drag Operation event, GCs are copied to GBs.
- *Drag Over Node* - You can attach event rule logic to this event to validate that the node the dragged record is about to be dropped on is a valid situation. If it is not, you can use a system function to change the cursor to a No Drop Cursor to indicate that dropping the record there will not be allowed. If the cursor is not the no drop cursor, when the record is dropped, the event *End Drag* will run.
- *End Drag* - You can attach event rules to this event to update or insert information that has been moved or copied via the drag. Be aware of the impact from using *Insert Grid Buffer Row* in the *End Drag* event, as well as deleting the grid row dragged if the operation is a Move.
- *Drag Mode* - The Drag Mode system value can be checked at any time to see what kind of drag a user is doing, for example, whether the drag is a move or a copy.

Tree Node is Selected

This event runs every time a tree node is selected, either by clicking on it once with the mouse or by moving an arrow up and down the nodes. You can place event rules that need to run when a mode is selected on the *Tree Node is Selected* event. The Work Center application uses this feature to load the media object that sits next to the tree with the message information for each node. You

can also use this when controls or exits must be protected based on the kind of node that is selected.

Defining Options for Forms, Grid, and Edit Controls

You can use options to define additional properties for forms and edit controls or UDC edit controls. After you create the form and controls, you can define options for a:

- Form
- Grid
- Control (edit and UDC edit only)

If you change any control options, you do not have to regenerate the application. Control options are interpreted at runtime.

Defining Options for a Form

► To define control options for a form

1. Focus on the form and choose Properties from the Form menu. On Form Properties, click the Options button.



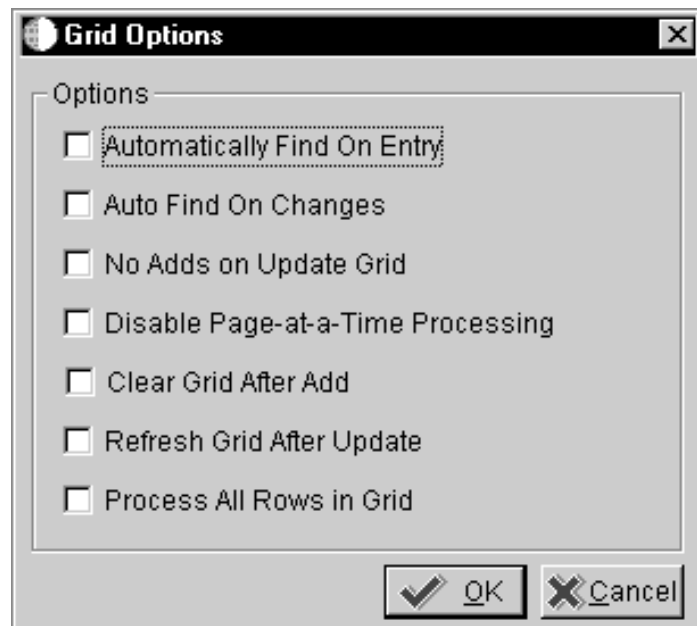
2. Click any of the following options to override the default processes:
 - No update on business view
 - No update on grid business view
 - No fetch on form business view
 - No fetch on grid business view
 - End form on add

Field	Explanation
No update on business view	Ignores updates to tables when you click OK.
No update on grid business view	Ignores updates to the grid when you click OK.
No fetch on form business view	Does not fetch records from the table when you enter a form.
No fetch on grid business view	Does not fetch records from the table when you enter a grid.
End form on add	Returns to the previous form after you add a record and click OK.

Defining Options For a Grid

► To define grid options

1. Focus on the grid and choose Properties from the Form menu.
2. On Grid Control Properties, click the Options button.



3. Click any of the following options to override the default process:
 - Automatically Find On Entry
 - Auto Find On Changes
 - No Adds On Update Grid
 - Disable Page-at-a-Time Process

- Clear Grid After Add
- Refresh Grid After Update
- Process All Rows in Grid

Field	Explanation
Automatically Find On Entry	<p>Automatically loads data into the grid upon entering the Find/Browse or Headerless Detail form. Use this feature only if there are not many records to retrieve, otherwise you may slow system performance unnecessarily.</p> <p>Note: J.D. Edwards standard is to turn this option off for Find/ Browse forms and to turn it on for Header Detail and Headerless Detail forms.</p>
Auto Find On Changes	<p>Automatically refreshes the grid after a change is made on another form and the user is returned to the Find/Browse form. Because of the impact on system performance, use this option only in applications in which instantaneous, up-to-date information for all users is required.</p> <p>Note: J.D. Edwards standard is to turn this option off.</p>
No Adds On Update Grid	<p>Prevents a user from adding records on an input-capable grid, such as on a Headerless Detail or Header Detail form. The user will still be able to change records.</p>
Disable Page-at-a-Time Process	<p>Page-at-a-time processing loads one page of records into the grid at one time. When the user scrolls past the last record in the page, the next page of records is read into the grid.</p> <p>Turning this option on disables page-at-a-time processing, and OneWorld will load all records into the grid at one time. If there is a large number of records, turning this option on could have a negative impact on performance.</p>
Clear Grid After Add	<p>On Header and Headerless Detail forms, this enables the user to add more records after clicking OK.</p>
Refresh Grid After Update	<p>On Header/Detail and Headerless Detail forms, this reloads the grid from the database in update mode.</p>
Process All Rows in Grid	<p>Processes all grid rows, including those that have not been changed. Use this option carefully because it negatively impacts performance.</p>

Defining Options For an Edit or UDC Edit Control

► To define control options for an edit or UDC edit control

1. Focus on the control and choose Properties from the Form menu.

- On Edit Control Properties or UDC Edit Control Properties, click the Options button.



- Click any of the following options:
 - Do not clear after add
 - Required entry field
 - Default cursor on add mode
 - Default cursor on update mode
 - No display if currency is OFF

Field	Explanation
Do not clear after add	Prevents the user from clearing this field after a record is added.
Required entry field	Prevents the user from leaving this field blank.
Default cursor on add mode	Designates this field as the initial position of the cursor in Add mode.
Default cursor on update mode	Designates this field as the initial position of the cursor in Update Mode.
No display if currency is OFF	Hides this option if the Multicurrency flag = No in System Setup, General Accounting Constants (P0000).

Associating Database Items, Dictionary Items, or Descriptions

Associate any of the following with a control:

- Database item
- Dictionary item
- Description

Radio buttons, check boxes, edit controls, and UDC edit controls can have associated values. A push button cannot have an associated value. Database items and dictionary items can only be associated with check box or radio button controls. You can look at the properties for a control to see which data dictionary item the control is associated with.

Database Items

If you want the user-entered value for a checkbox or radio button to update the record, then the control must be associated with a database item.

For example, a check box can be associated with the database field, “Taxable.” In the check box properties, the Checked value should be Y and the Unchecked value should be N. These values can be used in event rules and will update the Taxable database field in the table.

Dictionary Items

Because dictionary items never update the table, they can only be used in event rules. The user can enter values in the dictionary item that will be used in event rules or an event rule can produce a value that will be displayed in a dictionary item.

Descriptions

When you associate a description with an edit control or UDC edit control, a description of the value in the control is automatically displayed next to it. Associating a description is optional but is useful as a visual aid on the form.

For example, suppose the address book number 1001 appears in an edit control or in a UDC edit control. When the user tabs out of the control, the address book name “XYZ Company” is displayed next to the control.

To associate an item or description with a control

1. On the form with which you are working, click on the control.
2. From the Edit menu, click one of the following:
 - Associate Database Item
 - Dictionary Item
 - Description
3. If you are associating a database item or dictionary item, locate and choose the item.
4. If you are working with a description, move the control to the desired position on the form and click once to place it on the form.

If you associate radio buttons and you want them to be mutually exclusive, select them and apply a group box.

Changing Tab Sequence

The order in which the cursor travels through the controls on your form is determined by the tab sequence of the controls.

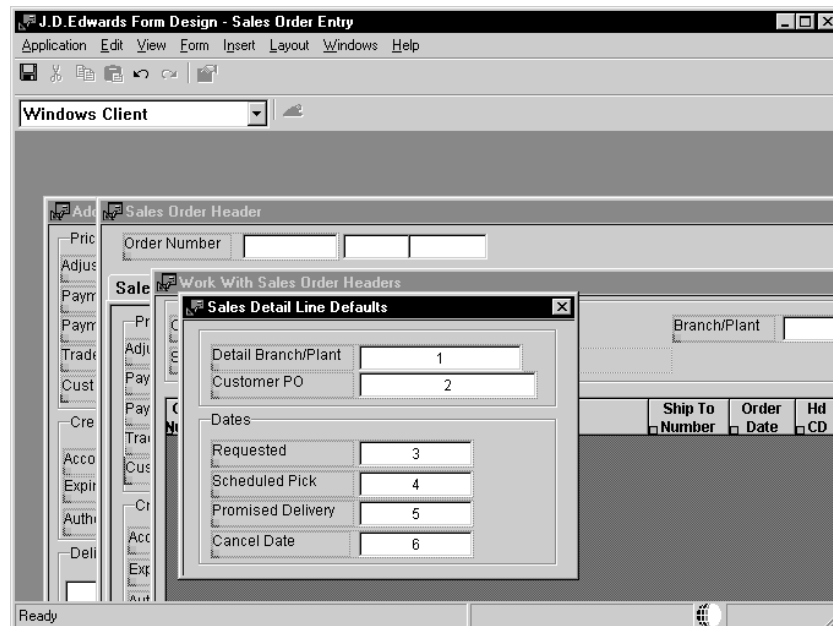
Each control that is designated as a tab stop is numbered according to the way the cursor travels. For example, when the form is first opened, the cursor will be on the control labeled number one. When the user presses the Tab key, the cursor moves sequentially to the next tab on the form.

Only those controls that are designated as tab stops in the control properties are affected by the tab sequence.

The default tab sequence is the order in which the controls are placed on the form.

► To change the tab sequence

1. On the form with which you are working, choose Tab Sequence from the Form menu.



The controls are already numbered.

2. Click the controls in the order you want the cursor to travel.

The first control you click becomes number one, the second becomes number two, and so on. To reset the numbers to the original values, click the right mouse button anywhere in the window.

3. To hide the tab sequence, from the Form menu, choose Tab Sequence again.

To set the tab sequence for a group of controls (such as a set of radio buttons), the Tab Stop and Group properties must be turned on for the first control in the group. All other controls in the group should have these properties turned off. This ensures that the cursor tabs to the first control in the group and skips the others. To move within the group, use the arrow keys.

The following two options override the first tab stop:

- Default cursor on add mode
- Default cursor on update mode

Creating Tab Controls

You can create a control that allows you to split a form into different tabbed pages. This allows you to use more controls for a single form. You can group your control functions by placing related controls on different tab pages for a single form. You can cut and paste controls from one page onto other pages.

There is a single business view for the form. There is one commit for the form on the OK button. You can use system functions, for example, Set Focus, to add additional functionality for the tab controls. Each tab page has a *Tab Page is Selected* event and a *Tab Page is Initialized* event associated with it. You can attach additional event rule logic to these events. When you use tab pages in an application, you can focus on the upper right corner of the tab page and move it around. This allows you to see several pages at the same time.

The following illustration shows a form that has tab pages.

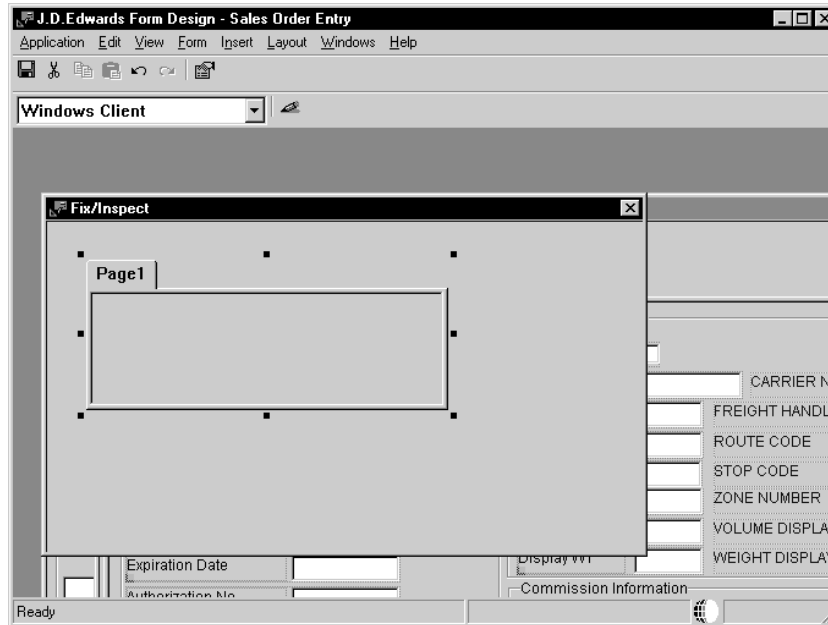
► To create a tab control

1. On the form with which you are working, choose Tab Control from the Insert menu.

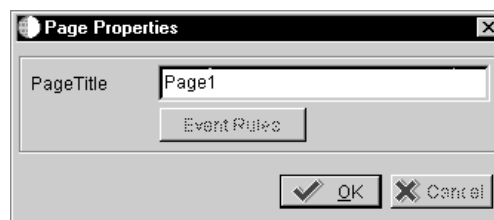
Page Properties appears to indicate which page of information you are on.

2. On Page Properties, complete the Page Title.

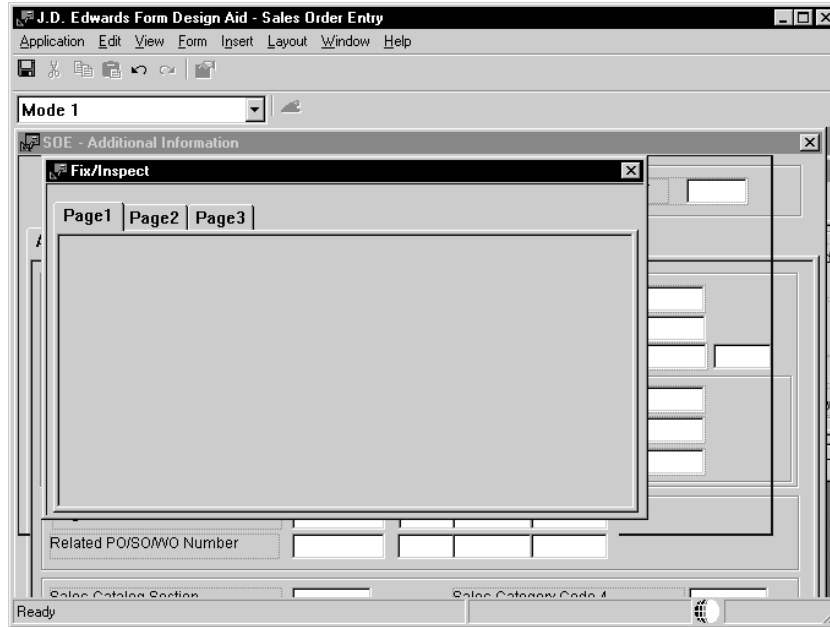
Your form appears with a tab at the top.



3. Position and resize the tab.



4. On Page Properties, change the generic tab name to one that suits your needs.



- Once you have created the initial tab control, choose Tab Page from the Insert menu to add additional tab pages as necessary.

The size of each page in the tab control is equal to the size of the entire tab control. You cannot resize an individual page to be bigger or smaller than the others.

Field	Explanation
Event Rules Title	The default name of the control for use in an event rule. The event rules title defaults to the static text associated with an item. If there is no associated static text and you want to use this control in an event rule, enter a name.
Page Title	Text that appears on the page tab. Avoid abbreviations and acronyms in the page title.

Designing Forms Using Multiple Modes

You can use control modes to develop an application with multiple interfaces. This reduces the need to maintain several different versions of the same application. You can create one base application and use modes to modify the application for different interfaces. You can enable or hide controls on forms for each mode. Only visibility properties for controls and columns are different for different modes. If you show hidden fields, they appear only for the mode you are in. All other properties are the same and are common for all modes. All fields are enabled and shown on all forms.

Control modes are not recognized by the Windows runtime engine, only the Java runtime. Mode 1 is the default mode. You attach an application to a menu to run. This menu allows you to run an application in different modes. When you run an application over the Web, the application runs in mode 1 by default and another mode if one is specified. If you attach an application to a Windows menu, the Windows runtime engine ignores any modes that are specified and runs the application in mode 1. Try to use modes consistently throughout your applications.

To create web-enabled versions of your forms, you generate them in Java and HTML using the Java & HTML Generator. The Generator allows you to generate forms for one or more modes simultaneously.

Designing forms using multiple modes contains the following topics:

- Viewing forms in a particular mode
- Setting control mode properties

See Also

 *Developing Web Applications*

Viewing Forms in a Particular Mode

You can view the forms you are developing in one of three modes:

- Mode 1
- Mode 2
- Mode 3

To view forms in a particular mode

1. On Forms Design, from the View menu, choose Control Modes
2. From the mode drop down menu that appears, choose one of the following mode options:
 - Mode 1
 - Mode 2
 - Mode 3

The forms and controls particular to the mode you have chosen appear.

You usually develop your applications in Mode 1 and customize them as needed for Mode 2 and Mode 3.

Setting Control Mode Properties

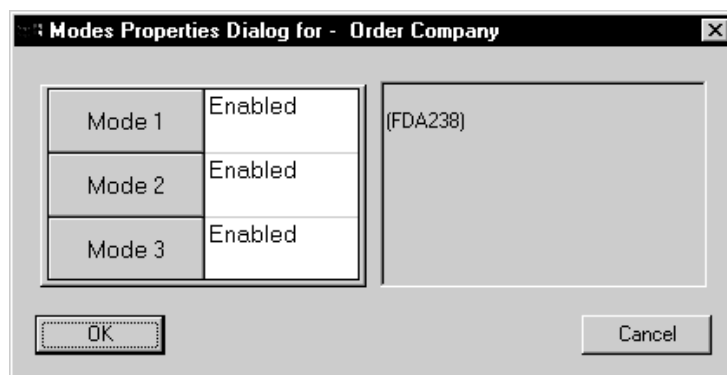
You can set mode properties individually for each control on a form. The default display for all modes is enabled. This allows you to enable or hide controls for each of the following modes:

If you hide a control, it is disabled. This means that the event rules and logic for that control will not run, so be very careful when you hide controls.

- Mode 1
- Mode 2
- Mode 3

► To set control mode properties

1. On the form with which you are working, right-click on the control you wish to enable or hide.
2. Choose Modes from the menu that appears.



3. On Modes Properties double-click on the Enabled box beside one of the following modes:
 - Mode 1
 - Mode 2
 - Mode 3
4. From the menu that appears, choose Enabled or Hidden.

Controls are automatically enabled unless you change them to hidden.

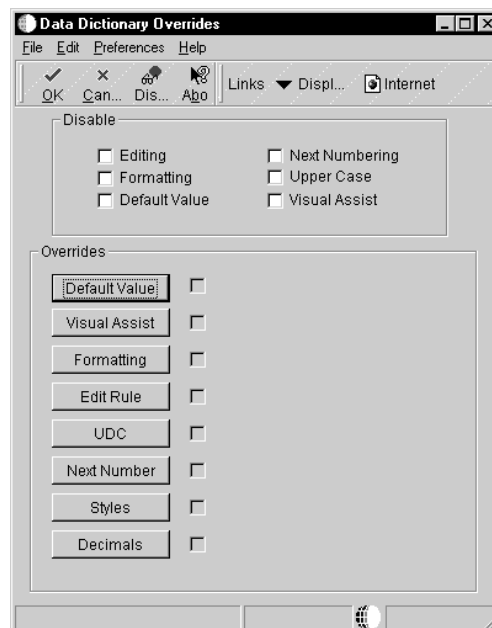
Overriding Data Dictionary Triggers at Design Time

When default triggers are defined in the data dictionary, the application they are attached to automatically executes them at runtime. However, you can override previously defined triggers at the application level, allowing you to further customize how data items behave at runtime.

Use the disable option to turn off data dictionary triggers for a specific item, or to override triggers to change to a new value or option.

► To override previously defined triggers

1. On the Form, Grid, Edit Control, or UDC Edit Control Properties form, click Overrides.



2. Click any of the following Disable options:
 - Editing
 - Formatting
 - Default Value
 - Next Numbering
 - Upper Case

- Visual Assist
3. Click any of the following to override triggers that were previously defined in the data dictionary:
- CB - Default Value
 - CB - Visual Assist
 - Formatting
 - CB - Edit Rule
 - CB - UDC
 - CB - Next Number
 - CB - Styles
 - CB - Display Decimals

See Also

- *Data Dictionary* for information about setting triggers in the data dictionary

Field	Explanation
Disable Editing	Disables the edit trigger, edit rule, and UDC edits for a field.
Disable Formatting	Disables the format trigger and format rule for a field.
Disable Default Value	Prevents the default value from automatically loading in a field when the user leaves the field blank.
Disable Next Numbering	Disables a next number trigger attached to this data item.
Disable Upper Case	Allows the user to enter lower case characters into a field that has been defined as Upper Case Only in the Data Dictionary.
Disable Visual Assist	Prevents the visual assist button from appearing when the control or grid column is selected.
Override Default Value	Inserts the value into the associated field whenever the user leaves the field blank.
Override Visual Assist	Allows the user to control the type of visual button that is associated with a field. If a UDC override is entered this field, it is automatically loaded with the UDC Search Form visual assist. (This override does not apply to the Universal Batch Engine.)
Override Formatting	Allows the user to override the format trigger or format rule that is associated with a field/control.

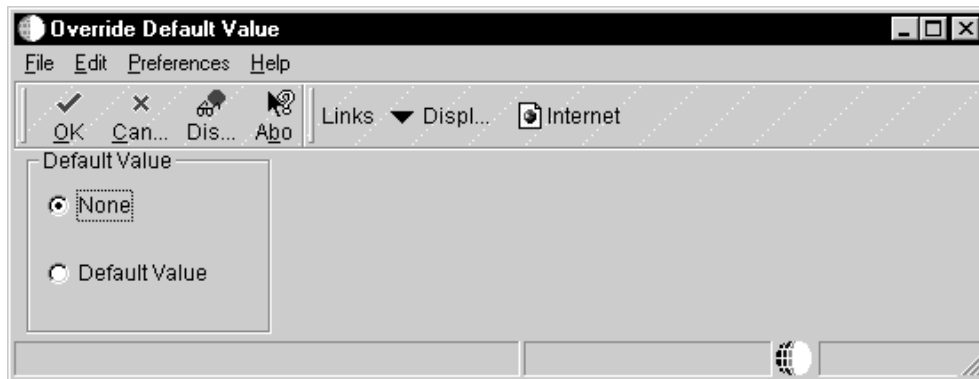
Field	Explanation
Override Edit Rule	Allows the user to override the edit trigger and edit rules associated with this field. If edit rules or procedures are entered for control, then the UDC override is disabled. A field can have either an edit trigger, edit rule, or UDC edit; it cannot have more than one of these types of edits.
Override UDC	Allows the user to specify the UDC table to which a field is validated against. When a UDC override is entered, the override search form is automatically assigned to the UDC Search Form. Entering an override disables the edit rule button and clears any edit rules or procedures.
Override Next Number	Allows the user to specify the location and starting value for next numbering on a field.
Override Styles	Allows the user to override the styles, Allow Blank Entry and Upper Case Only. Allow Blank Entry is used in conjunction with a UDC edit. If the UDC table for which a field is validated does not contain a blank value, then specifying the allow blank entry flag allows the UDC field to contain a blank value without generating errors. The upper case only flag causes a field to convert all entry values to upper case.
Override Decimals	Allows the user to control the number of decimals instead of the traditional two decimal places. This override pertains to display decimals only.

Overriding a Default Value Trigger

You can override a default value trigger to assign a different default value for a field or to allow no default value.

To override a default value trigger

1. On Data Dictionary Overrides, click the Default Value button.
2. On Override Default Value, click one of the following:
 - None
 - Default Value



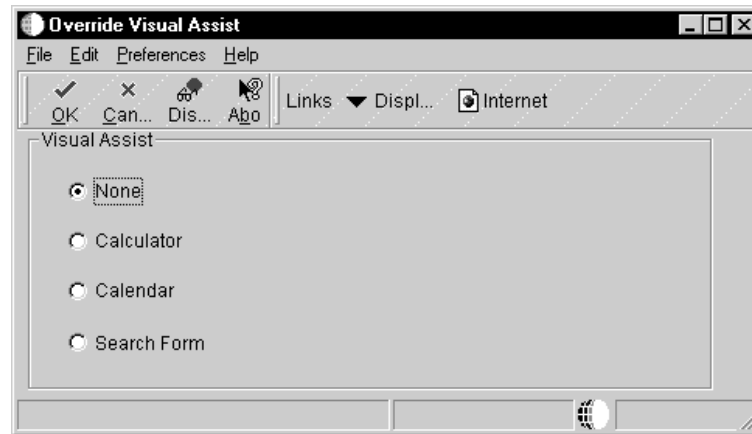
Field	Explanation
Default Value	<p>Indicates whether there is a default value on the field. Choose either of the following options:</p> <ul style="list-style-type: none"> • None - no default value is assigned • Default Value - assigns a default value to the field. Specify the default value that this field will have.

Overriding a Visual Assist Trigger

You can override a visual assist trigger.

► To override a visual assist trigger

1. On Data Dictionary Overrides, click the Visual Assist button.
2. On Override Visual Assist, click one of the following:
 - None
 - Calculator
 - Calendar
 - Search Form

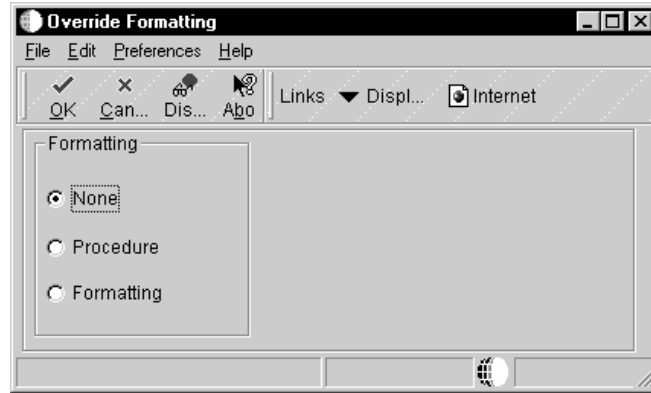


Field	Explanation
Visual Assist	<p>Indicates whether there is a visual assist on a field and the type. Choose from the following options:</p> <ul style="list-style-type: none"> • None - no Visual Assist is assigned • Calculator - assigns a calculator to the field. Use this on all numeric data items where a user might need to type in an amount. • Calendar - assigns a calendar to the field. Use this on data items where a user might need to type in a date. • Search form - assigns a search form to the field. Use this on all fields where the valid values for the field are found in a file.

Overriding a Formatting Trigger

► To override a formatting trigger

1. On Data Dictionary Overrides, click the Formatting button.
2. On Override Formatting, click one of the following:
 - None
 - Procedure
 - Formatting



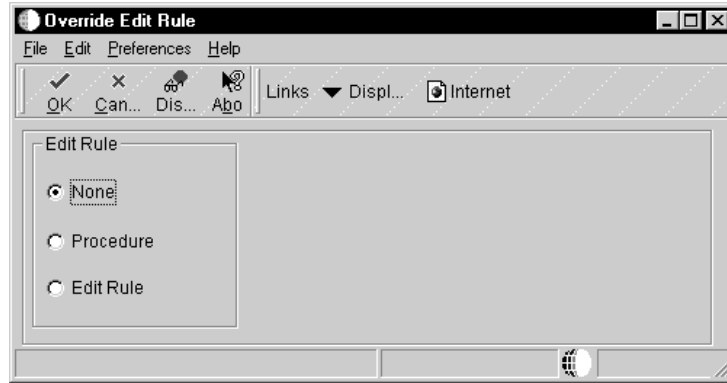
Field	Explanation
Display Rule	<p>Indicate whether or not there is a Formatting rule on the field. Choose from the following options:</p> <ul style="list-style-type: none"> • None - no Formatting rule • Procedure - assigns a business function ID to the field. Specify the business function procedure in the field that becomes available when you choose this option. Use the visual assist to view and select a business function. • Formatting - assigns a display rule to the field. Specify the display rule in the fields that become available when you choose this action. Use the visual assist to view and select rules.

Overriding an Edit Rule Trigger

You can override an edit rule trigger.

▶ To override an edit rule trigger

1. On Data Dictionary Overrides, click the Edit Rule button.
2. On Override Edit Rule, click one of the following:
 - None
 - Procedure
 - Edit Rule



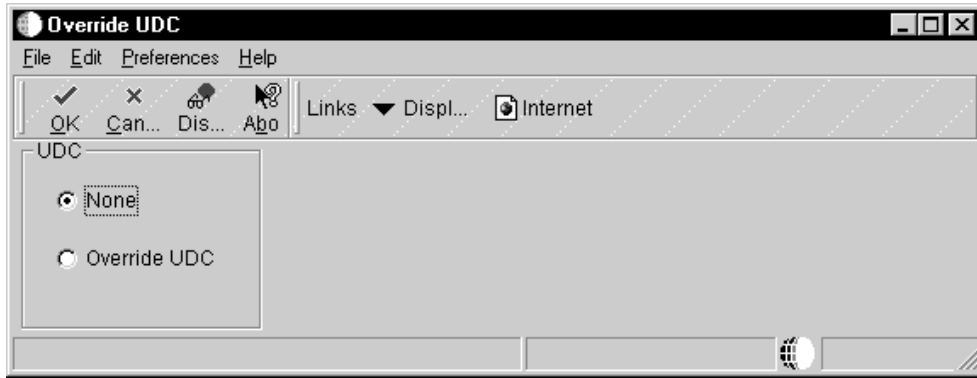
Field	Explanation
Edit Rule	<p>Indicate whether there is an Edit Rule on the field and the type. Choose from the following options:</p> <ul style="list-style-type: none"> • None - no Edit Rule is assigned • Procedure - assigns an Edit Rule based on a business function. Use Business functions when you need special processing that cannot be done through one of the five display rules. For example, if you would like to format the appearance of the account number, attach a business function called Display Account Number. Specify the business function Procedure in the field that becomes available when you choose this option. Use the visual assist to view and select a business function. • Edit Rule - assigns an Edit Rule. Specify the edit rules in the fields that become available when you choose this option. Use the visual assist to view and select rules.

Overriding a User Defined Codes Trigger

You can override a user defined codes trigger.

► To override a User Defined Codes trigger

1. On Data Dictionary Overrides, click the UDC button.
2. On Override UDC, click one of the following:
 - None
 - Override UDC



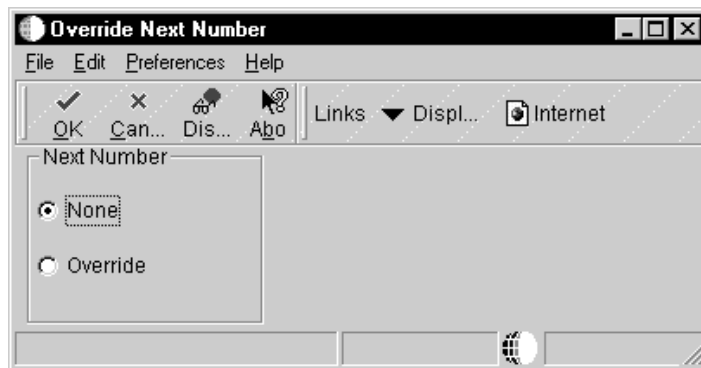
Field	Explanation
UDC	<p>Indicate whether there is a User Defined Code override on the field. Choose either of the following options:</p> <ul style="list-style-type: none"> • None - no UDC table is available • Override UDC - assigns a user defined code override to the field. Specify the System Code and Code Type for the UDC table.

Overriding a Next Number Trigger

You can override a next number trigger.

► To override a next number trigger

1. On Data Dictionary Overrides, click the Next Number button.
2. On Override Next Number, click one of the following:
 - None
 - Override



Field	Explanation
Next Number	<p>Indicate whether there is a Next Number override on the field. Choose either of the following options:</p> <ul style="list-style-type: none"> • None - no Next Number is assigned • Override - assigns next number logic to the data item. Specify the Next Number System Code and Next Number Index the data will use in the Next Numbers file.

Overriding a Styles Trigger

You can override a styles trigger to change the way information on a form displays.

► To override a styles trigger

1. On Data Dictionary Overrides, click the Styles button.
2. On Override Styles, click one of the following:
 - Allow Blank Entry (Y/N)
 - Upper Case Only (Y/N)



Field	Explanation
Allow Blank Entry	<p>Turning this option on allows a blank value to be written to the database under the following conditions:</p> <ol style="list-style-type: none"> (1) If the field is edited against a UDC table, a blank value will be allowed regardless of whether a blank value is valid for the table. (2) If the field is specified to be a mandatory entry, a blank value will be allowed as a valid entry.

Field	Explanation
Upper Case Only	If the value of this field is a Y, the user will be allowed to enter only upper case characters in the entry control.

Overriding a Decimal Trigger

You can override a decimal trigger to change how decimals display.

To override a decimal trigger

1. On Data Dictionary Overrides, click the Decimals button.
2. On Override Display complete the following field:
 - Display Decimals

Field	Explanation
Display Decimals	Designates the number of decimals in the currency, amount, or quantity fields the system displays. For example, U.S. Dollars would be 2 decimals, Japanese Yen would be no decimals, and Cameroon Francs would be 3 decimals.

Using Text Variables

Text variables are stored as strings and can be used as an alternative to hard coding text strings in assignments. Because text variables are not hard-coded, they are easier to maintain. Following are some of the ways you can use text variables:

- To reuse forms, for example, you can reuse message forms and display the appropriate message depending on specific conditions.
- To reuse grid columns instead of hiding and showing them by changing the column heading text.

To use a text variable you:

- Add the text variable itself
- Use system functions to attach the text variable or use the assignment to assign it

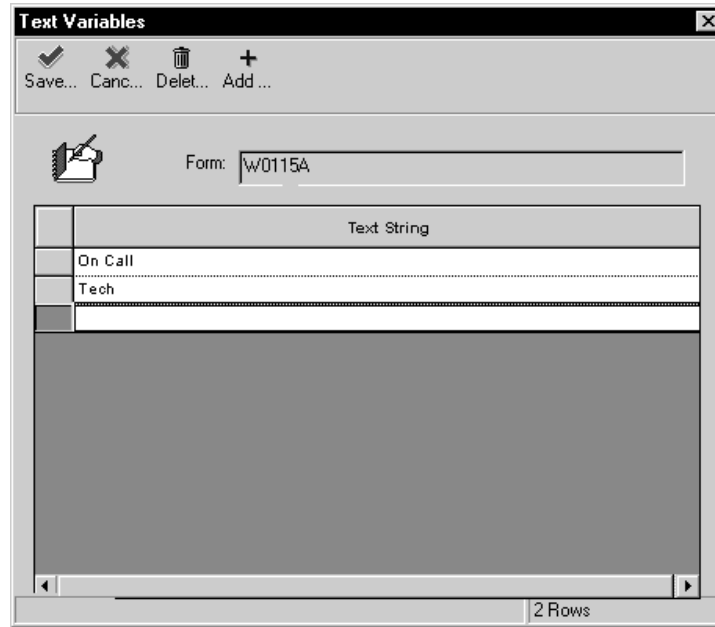
Adding a Text Variable

You can use the Text Variable form to add all of the text variables you want to use for a specific form.

To add a text variable

1. In Form Design, on the form for which you wish to add a text variable, choose Text Variables from the Form menu.

Current text variables associated with the selected form appear.



2. Under Text String, choose the last empty row and enter the text string you wish to use.
3. Tab to the next row to create another text variable.

If you try to create a text string that is named the same as another variable on the same form (or on the same section in Report Design) an error occurs. Rename the variable to clear the error.

4. Click Save to save your variables.

You can modify and delete text variables. To modify a text variable, simply type over it and save the new text.

If you delete a text variable that is referenced in event rules be sure to delete its reference in event rules.

Attaching or Assigning a Text Variable

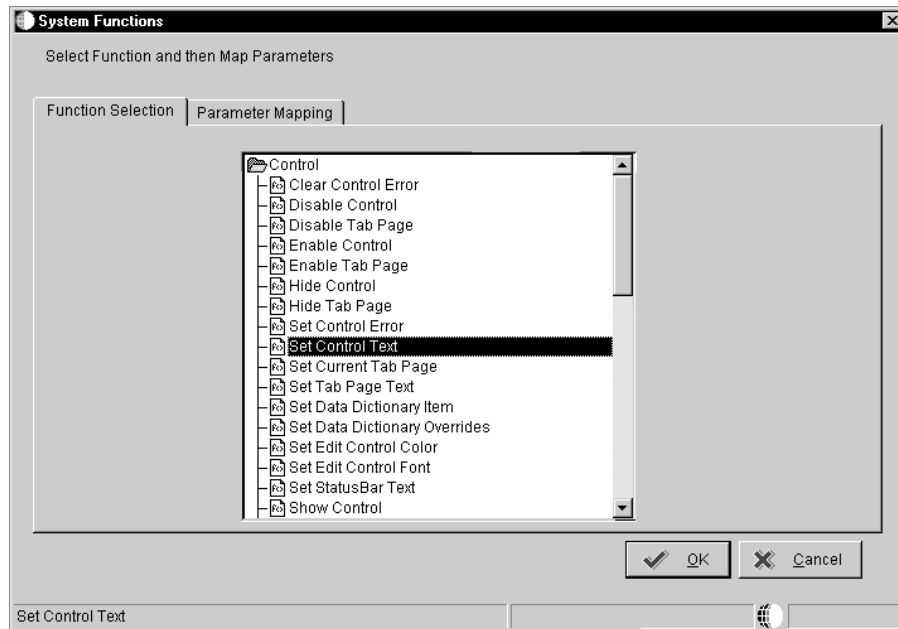
You can use system functions to attach text variables. You can use a system function to change the text for a control, to change the name on the form, or to change the text in a grid column heading. The following system functions are typically used for text variables:

- SetControlText
- SetGridColumnHeading
- SetFormTitle

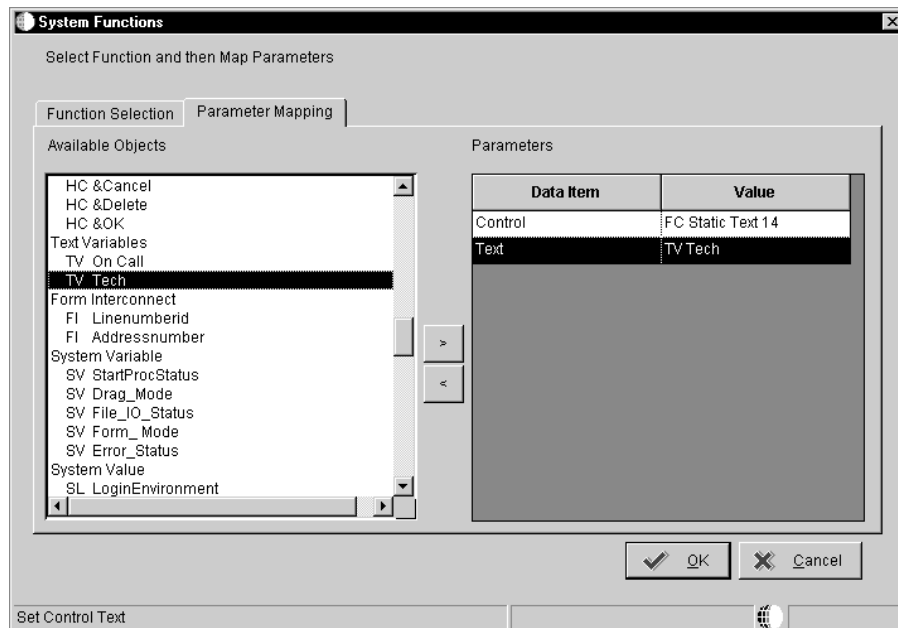
The available text variables appear in the Available Object list for System Functions. Available text variables also appear when you use assignments.

► To attach a text variable

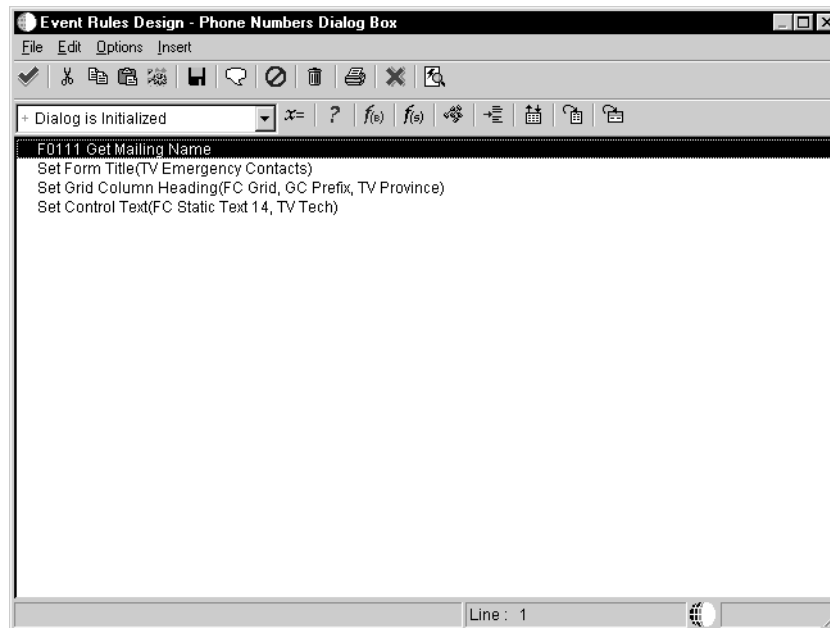
1. If you want to attach text to a control, on the form you are working with, choose event rules from the Edit Menu.
2. From the events list, choose the event to which you want to attach the text variable.
3. Click on the System Function button.



4. Choose the Set Control Text system function.



5. Complete the parameters.



Refer to *Attaching Functions* for more information about attaching system functions.

You can also assign a text variable to the edit fields of data dictionary and business view items. Refer to *Creating Assignments*.

Using Quick Form

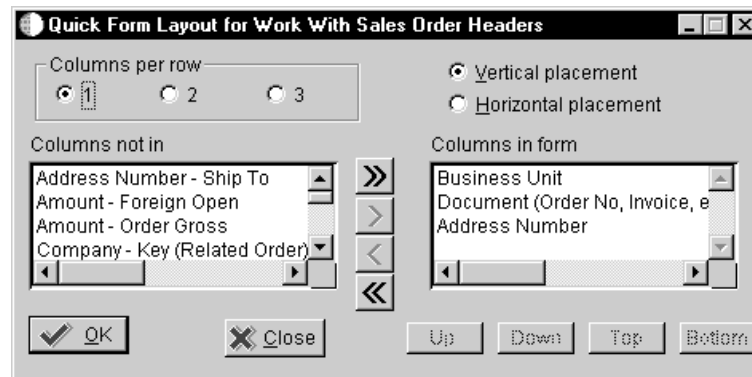
Use Quick Form to easily place multiple database fields on a form, rather than choosing each data item individually. Simply choose one or more data items and Quick Form automatically places each field on the new form simultaneously.

Depending on the number of selected fields, you might need to resize the form or move and align fields to achieve the desired layout.

► To use Quick Form

1. On Forms Design, choose Quick Form from the Form menu.

The title bar reflects the business view associated with the form.



2. Click the following options:
 - Columns Per Row
 - Vertical/Horizontal Placement
3. Choose and sort columns from the business view.
4. Click OK to place the fields on the form.

Quick Form remains open so you can adjust the arrangement by changing the columns per row and the vertical and horizontal placement.

5. Click Close when you are finished.

Field	Explanation
Columns Per Row	Specifies how many columns will be used to arrange the fields.
Vertical/Horizontal Placement	Specifies the space between the horizontal and vertical grid coordinates. The default spacing is 8 pixels between each horizontal and vertical gridline.
Columns not in form	Indicates the data items in the business view that are associated, but not yet placed on the form. Use none, some, or all the items in the business view. For example, on Find/Browse forms you can display fields only in the grid control and none on the form.
Columns in form	Lists the selected data items from the business view that are placed on the form.

Processing Media Objects

Media Objects are used to link information or “attachments” to application transactions. For example, you can attach drawings of products to a form. You use a media object data structure to pass information between your application and the media object table.

You can use standard processing for media objects to bypass all event rules that are required to implement media objects. All of the required information is gathered for a form in Forms Design Aid and does not require the entry of any event rules. Standard processing does the following:

- Allows implementation of media objects at the form level with no event rule coding required
- Standardizes the usage of media objects across forms
- For any grid, places a paper clip icon on the row header if a media object is defined for that row
- For a form, places an icon in the status bar if a media object is defined for the form
- Allows you to attach documents to the form or to a row in the grid
- Allows you to double click on the paper clip in a row to start media objects for that row
- Allows you to click on the paper clip in the status bar to start media objects for the form

If you choose not to use standard processing for a form, you can still use event rules and system functions to make media objects work.

The F00165 table is used to store link records for media objects and imaging. You must define your media object data structure using a unique key structure so that the F00165 table can store data correctly. The layout of the F00165 table is as follows:

GTxxx || F4211Keys || The media object text

GTxxx is the naming convention when defining a media object data structure. The F4211Keys portion is what the system uses to access the unique media object attachment for that particular record. The keys here typically match what the unique key would be in the F4211 for each detail line. The media object text portion is the actual text attachment that would store information typed in by the user. The F00165 table chains records for text greater than 30K. A sequence field indicates the order of the text and the text is stored in two files.

The F00166 table contains a record with characterization values for each media object that has been characterized. Each record contains the characterization values for a single media object, for example a text or word document. There can be multiple F00166 records for each F00165 record. Their key is the same as that of the F00165 plus the Object ID.

The F00167 contains a record that indicates which categories will be used for characterize setup for each GT structure. There is only one record for each GT structure.

The F98MOQUE table is used for multiple OLE queues. It includes

- Online/offline queue paths
- Secondary text queues
- Queue status information, for example read/only and read/write information
- Longer queue names that allow FTP access by the Java Application Server

This table allows you to put media objects in several different queues.

Imaging

Imaging allows you to use images from third-party software. Imaging is done on a form-by-form basis for performance reasons. When you attach an image to a document, the media object form includes the vendor's software as a choice.

When you use a third-party vendor, the F00165 file stores the reference to image attachments, but the third-party software controls the search and retrieval of images.

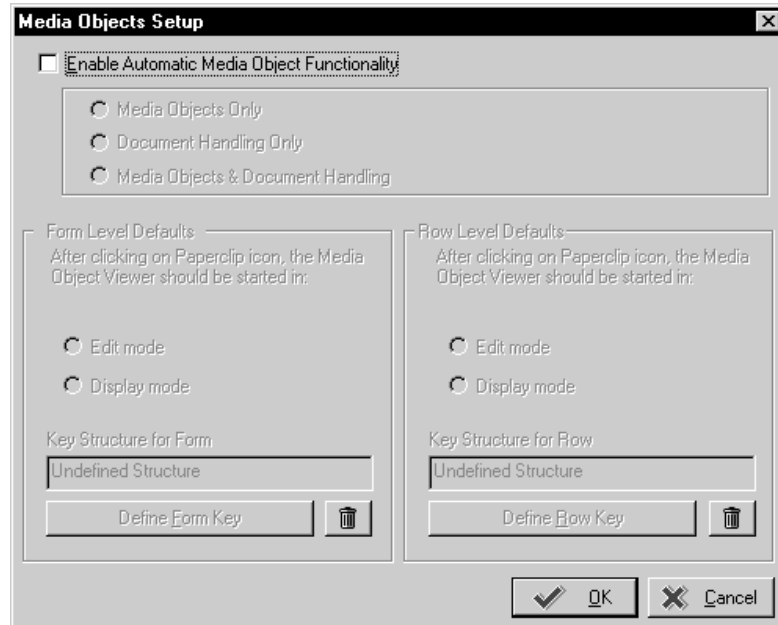
Before You Begin

In order to see the media object paper clip column on your form, you must turn off the Hide Row Numbers option in the Grid properties for the form in Form Design.

Using Standard Processing for Media Objects

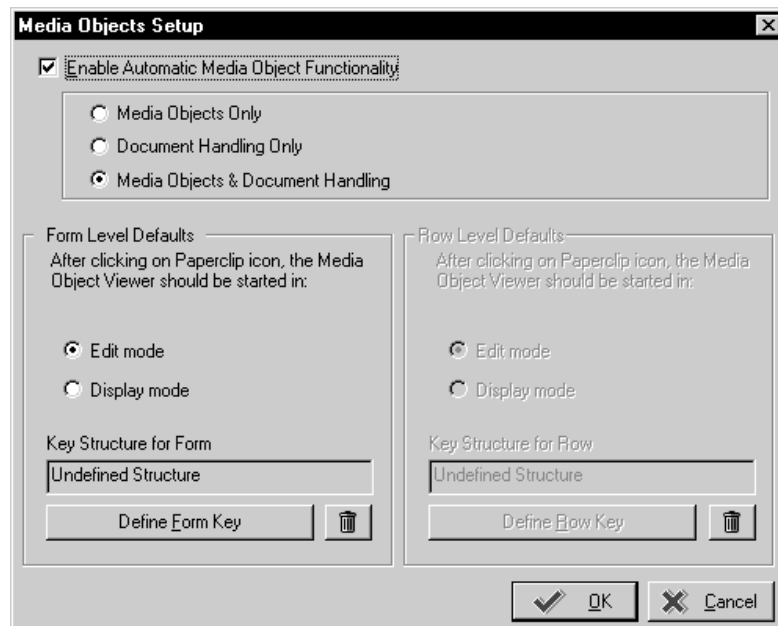
► **To use standard processing for media objects**

1. On Form Design, choose Media Objects Setup from the Form menu.



2. On Media Objects Setup, choose Enable Automatic Media Object Functionality.

This enables imaging and opens the other fields on the form



3. Click one of the following:
 - Media Objects Only
 - Document Handling Only

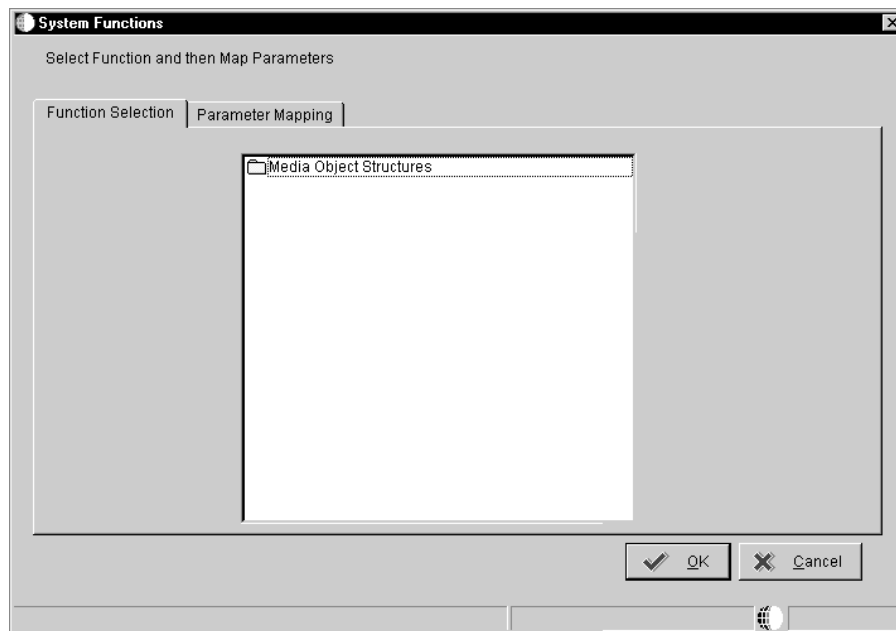
Use the Document Handling Only option if you are developing a form that is enabled for media objects via functionality in event rules and you wish to bypass standard processing. If you wish to enable standard processing later, you must delete all of the event rules for media objects and click the Media Objects & Document Handling option.

- Media Objects & Document Handling
4. Click Edit mode or Display mode.

Edit mode allows the user to make changes, while display mode is read-only.

5. Click the Define Form Key button.

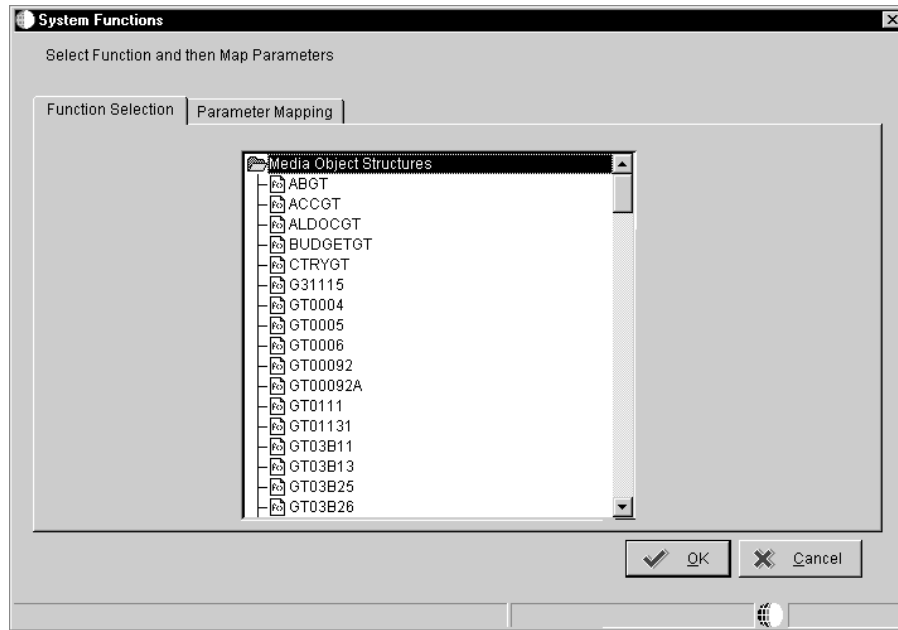
The System Functions form appears. This form is identical to the parameter definition form used to define system functions in Event Rules, except only the Media Objects header is displayed.



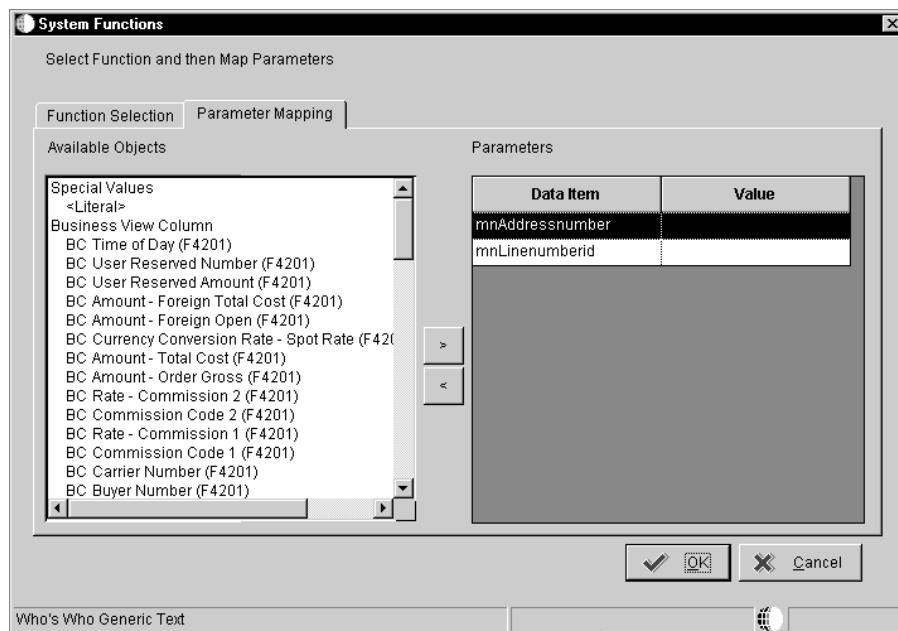
The key structure is used to store link records in the F00165 file. These links are critical to proper functioning of Imaging.

6. Double-click on the Media Object Structures folder.

A list of all of the currently defined data structures for Media Objects displays.



7. Choose the appropriate structure and define it.



Field	Explanation
Enable automatic media object function	Check this to define the media object. When this is checked, the form level defaults and row level defaults are enabled so that you can define the media object. When this is not checked, all fields on this form are dimmed.

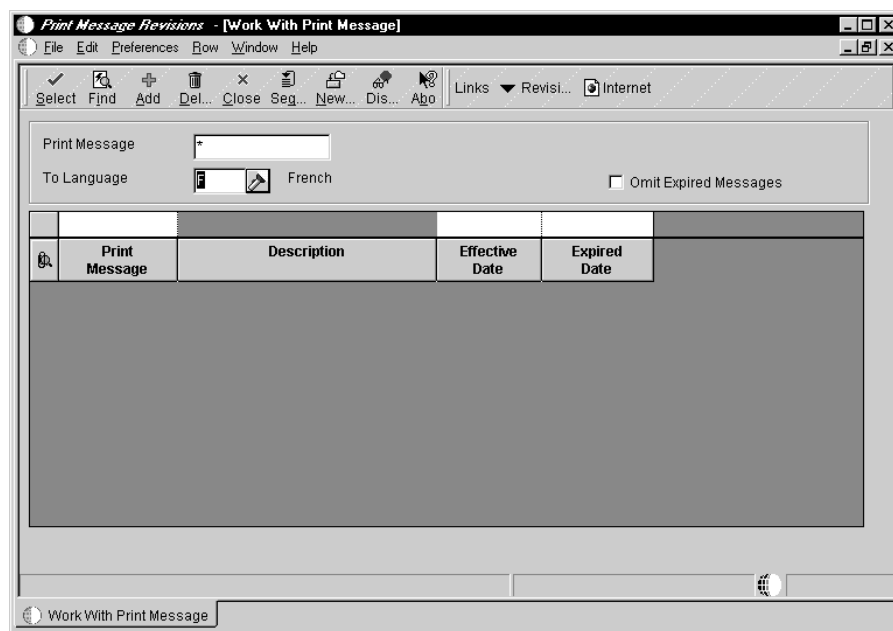
Field	Explanation
Media Objects Only	After enabling automatic media object functionality, choose one of the following options: <ul style="list-style-type: none"> • Media objects only • Document handling only • Media object and document handling Use option, Document Handling Only, if the form uses event rule logic rather than standard processing for media objects. If you want to enable standard processing later, you must delete all event rules for media objects and then select Media Objects and Document Handling.
Edit Mode	Allows the user to change the media object.
Display Mode	Changes for the media object are not allowed. The user can only view the media object.

Language Considerations for Media Objects

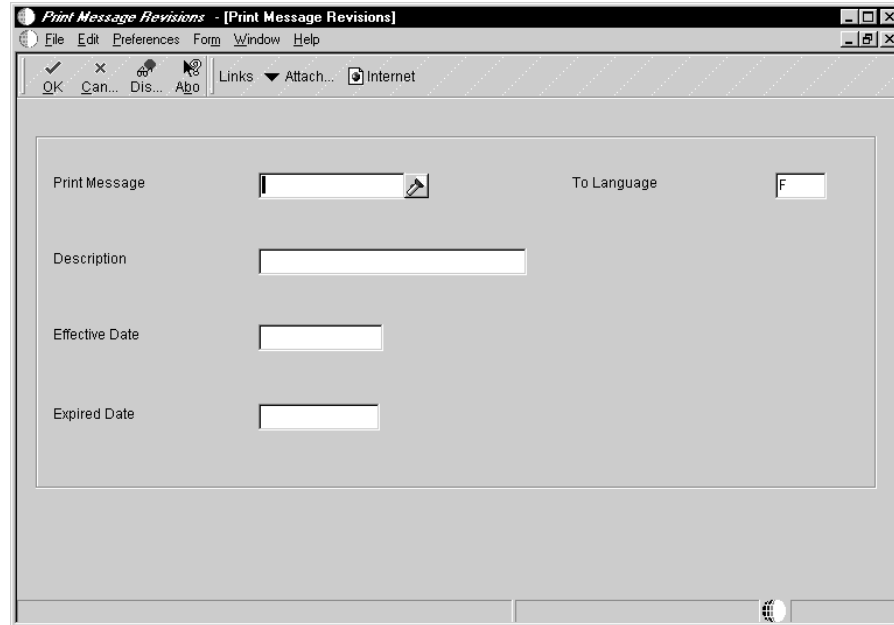
You can add media object (attachments) for a particular language. Most J.D. Edwards applications are media object enabled for languages.

► To add a language-specific media object attachment

1. On the application which you wish to use, type a language in the filter field.



2. Click Add.



3. Add multiple records if you want the attachment for multiple languages or base.

If you create a custom application that you want enabled for media object language handling, you must include a data item language preference (alias LNGP) in the generic text data structure you create.

Example: Language Considerations for Media Objects

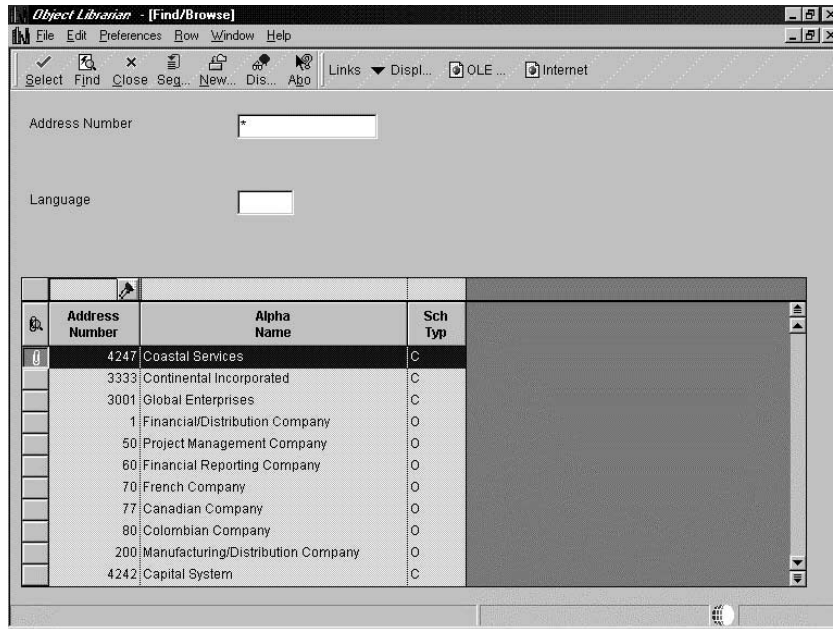
When you design an application, you can allow the end user to add separate and unique media objects, to the same record or different records, based on the language chosen.

If language (LNGP) is not a database column, then you define your media object (GT) data structure to include language as part of the data structure. You place a data dictionary control (LNGP) on the application as a filter field which should then be loaded with the system value for language. When you design your application this way, you attach two separate media objects, based on the language, to the same record.

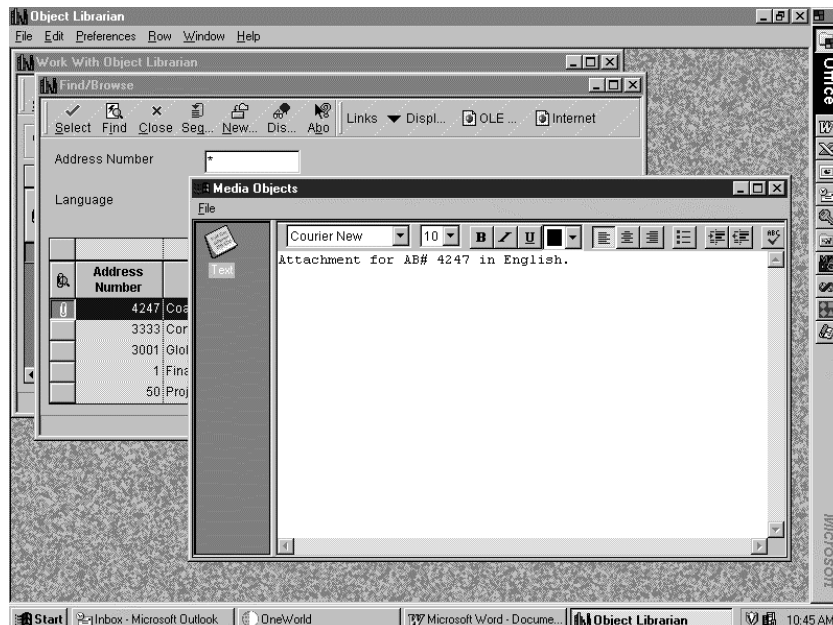
If language (LNGP) is a database column, then you include LNGP (database) as a filter field, but you must add separate record to the database table along with its media object attachment. The media object data structure still contains language as part of the key to retrieve the media object attachment. In both cases, the language filter fields (LNGP) needs to be loaded with the system value for language.

In both of the above situations, the language filter fields (LNGP) must be loaded with the system value for language. LNGP must be built into the key and not associated with the F00165 LNGP column.

For example, the following Address Book application includes Address Number and Language as filter fields. Note that the language control is not from the database, but is a Dictionary control.



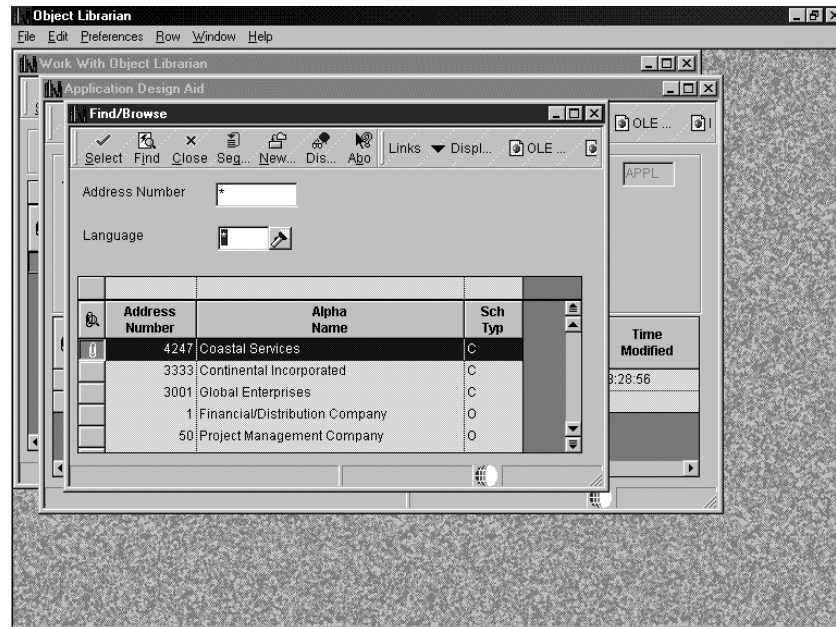
For a blank language (English), Address Book record number 4247 has a media object attachment in English.



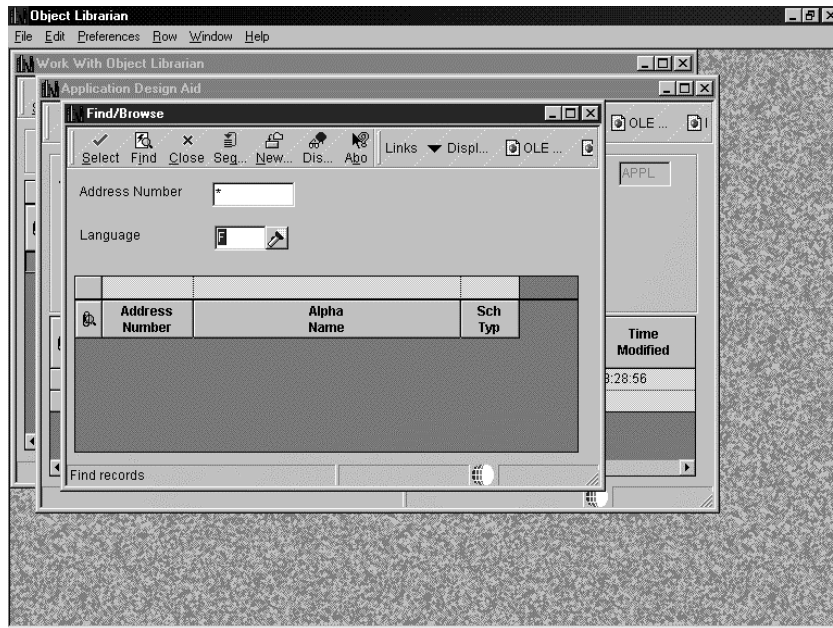
If the same record was accessed by a user whose language preference was set to French (F), the filter field would get loaded with “F” and the user would see a totally different Media Object attachment for the same record. The Address Book would not have two separate records based on the language, but only

one record in English. For the French Media Object attachment a separate media object attachment record is kept for the same Address Book record in the F00165 table. It can be accessed using the language filter control field.

In this example, the ABLNGP field/column from the Address Book table is used as the filter field on the form. The Language field on the form is from the database and not a Data Dictionary item.



When you search for records using blank or * (English or ALL) as the base language, you see all records in the Address Book with their corresponding media object attachments. You use the same media object data structure as you did with the previous case, which contains the Address Number and Language data items. You still load the user's language preference in the language filter field, but now when the application fetches records from the database it is using the language as part of the key to select records from the Address Book. In this example, if there are no records in the Address Book with the language preference for French "F" you do not see any records and you then must add records using French as the language.



Because the Address Book is only keyed using Address Number and not the language we cannot add a record for Address Book# 4247 in French, but we can add new records with new Address Book Records with new AB #'s with the language Preference of French and then be able to see only those records.

For any database table containing language as part of its key, you can attach media objects functionality for records with different languages. For example, one record for English and a copy of the record for French with unique media object attachments. For tables that do not include language as part of the key to that table, you can have media object languages, for example item master print messages GH4112, P40162.

Testing a Form

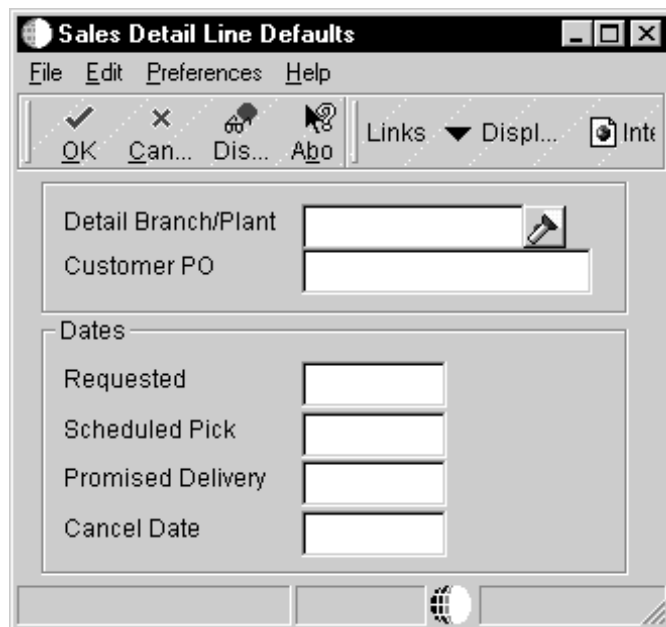
You should test a form to verify placement of controls on the form. When you test a form, it displays exactly as it will appear in the application.

Test mode is only for viewing. You cannot modify a form in test mode. Test works for the default Windows mode.

▶ To test a form

1. On Forms Design, choose Test from the Form menu.

The form is displayed as it will appear in the working application.



2. To return to design mode, click Close or Cancel in the test form.

Language Preview

You can preview forms in foreign languages. You must have the base language installed on your machine to do this.

▶ **To preview a form in another language**

1. On the form you want to preview, choose Language preview from the Form menu.
2. Choose the language in which to display the form and click OK.
3. To continue designing the form, close the language preview form.



ActivEra Portal Design

The flexible nature of the ActivEra Portal is based on the ability of its users to design their own components. A component is an object that appears within the Portal to provide links, information, or other web- or network-related functions. There is no limit to the number of components that you can design or to the number of components that end users can include in their workspaces. You can make components selectable by end users so that they can freely place the components in their workspaces. You can also limit access to components and create system-wide workspaces for which end users cannot configure.

Note: To design components and system-wide workspaces, you must have appropriate security access to the Portal. See the *ActivEra Portal Configuration* in the *System Administration* guide for more information.

System-wide workspaces are created in exactly the same way as personal workspaces. If your permissions are set properly, you can designate any workspace that you create as a system-wide workspace, making it available to end users. See *ActivEra Portal* in the *OneWorld Foundation* guide for information about creating workspaces.

When you design components, the ActivEra Portal Builder Servlet (Portal Builder) provides a variety of implementation strategies and interface choices. The base component, that is, a component in normal mode, can be designed with HTML code or based on a URI reference. You can also write a Java applet to use as the base component. The Portal Builder imbues all components with the ability to be minimized. A minimized component displays its title bar only. Additionally, you can provide any component with functionality for one or more of the following modes: maximize, help, and personalize.

If you provide functionality for a mode, then the Portal Builder includes an appropriate icon on the title bar that the user can click to launch the mode. All three of these alternate modes take up the entire workspace area. Additionally, you can provide functionality for a mode called link. The link mode functions exactly like the maximize mode, but it has no associated icon. If you provide a component with link mode functionality, you will also need to provide users with a way to access the mode from within the component.

In a typical design scenario, the designer renders the component's normal view with HTML, a URI, a Java applet, or a OneWorld component. Then the designer creates either additional web pages for the other modes and then references them with URIs, JavaScripts, or OneWorld components that can be invoked to provide functionality. You can use different solutions for different modes. For example, a component's maximize icon can call a URL while its personalize and help icons can invoke JavaScript functions.



ActivEra Portal design consists of the following topics:

- Planning Portal components
- Building HTML components
- Building URI components
- Building OneWorld components
- Building Java applet components
- Building Java servlets components
- Understanding the Link Center component
- Understanding Portal design standards

Planning Portal Components

Regardless of the kind of components you design for use in the ActivEra Portal, you should be aware of a number of factors that might affect your design decisions. Your component will share screen space with several other components when in normal mode. Components in this mode should adapt to changes in width to the greatest extent possible. If your component cannot adapt to changes in width, the Portal framework displays a horizontal scroll bar to allow users to access off-screen content.

A reasonable size for your component is approximately 250–300 pixels wide by 250–300 pixels high. The height of the component is not as critical as the width, because the component will push other components located beneath it down. Keeping components within these limits, however, will help to provide a much cleaner interface and easier interoperability with other components. The normal mode of the component should be relatively small. Remember that the portal framework supports a maximized view in which your component can use the majority of the screen. Normal mode should be a reasonably-sized introduction to your main content.

Your component is responsible for storing any information that it needs for personalization or to support its functionality. You may use any mechanism or technology to store your information. The OneWorld Component Java API for the ActivEra Portal also provides a mechanism for storing XML Data in a database, but the ActivEra Portal Builder Servlet itself is not involved in storing your component's data.

It is your responsibility to determine the browsers on which your components will function properly. Currently, the ActivEra Portal supports Netscape 4.08 and higher and Microsoft Internet Explorer 4.01 and higher. If your target audience uses only one browser, your component can take advantage of the features provided by that browser. If you target a generic audience, you will want to use generic functions and ensure that your component works on multiple browsers.

Finally, pages within a component need to fully qualify its URL's for images and anchors or the ActivEra Portal Builder Servlet's server information will be assumed. Consider the following example:

```
<html>
<body>
<image src="mypic.jpg" >
</body>
</html>
```

In this example, `mypic.jpg` is a relative link. If this page was located at `http://www.mysite.com/mypic.html`, then the browser would look for `mypic.jpg` at `http://www.mysite.com/mypic.jpg`. Now consider an ActivEra Portal, located at `http://www.aap.com/servlet/...` tries to call your component. When looking for your image, the browser would try and find `mypic.jpg` at `http://www.aap.com/servlet/mypic.jpg`. This problem can be solved by changing the image tag to read:

```

```

Consider the following additional issues when planning new Portal components:

- Will you store your component on a server or in an area accessible only to the ActivEra Portal?

If the component should be accessible to the ActivEra Portal only, then it must be an HTML component.

If the Portal can access the component via a server, then you can use straight HTML (including ActiveX components), Java applets, URI components, or OneWorld components. If you create an interactive URI, J. D. Edwards recommends using Java or JSP because the Portal itself is built on this technology, so integrating components built with Java or JSP is easier. If you need to create a OneWorld component (that is, a component that requires access to the database information, sign on information, or both), then your component just be a Java component. Note that applet tags reside within the Portal itself and are not accessible outside the Portal, even though the applet itself can be accessed outside of the Portal.

- Will your component need server-side programs to support it?

These programs can be in almost any programming language that is supported by the server that is hosting the component. If you must use server-side programs, you will probably need to create a URI or OneWorld component. URI and OneWorld components can also include applets and ActiveX controls.

- What type of component will you be building?
 - HTML components require no interaction with an external server. They can contain references to applets, ActiveX controls, and images, but the HTML is taken directly from the ActivEra Portal Builder's database.
 - URI components always require interaction between the ActivEra Portal and a web server. These components can be written as CGI, Java servlets, or HTML pages.
 - OneWorld components must be written in Java, and they must derive themselves from the `OneWorldComponentServlet` class. This

class must be accessible from the PortalBuilderServlet through the classpath. The components might need to exist on the Web Server if you want to access them directly. Typically, these components are essentially Java servlets.

- Applet components require the browser to access an applet from a web server. This applet runs on the client side and requires no resources other than a place to download your applet's classes. This type of component requires the most testing out of all of the components because it is dependent on the user's Java Virtual Machine (JVM), browser, and operating system.
- Will your component provide access to OneWorld applications, require database access using the ComponentDatabaseInterface, or need to reference the session information?

To access a OneWorld application, or to interact with the active session (such as when you need to access or pass a user ID), you must create a OneWorld component. OneWorld components are essentially URI components, but do not require a fully-qualified URL; OneWorld components use a class and package name instead. Additionally, OneWorld components must be located on the same server as the Portal because they must be able to access session information.

- Will your component display help or copyright information to users, or need a maximized view or personalization pages?

To display information in help, maximized, or personalization mode, you must enable the mode for the component and then provide material to be displayed. If this information contains multiple pages, HTML probably will not work because it supports only one page. You can use the Java APIs in the ActivEra Portal to wrap multiple pages into the component, if necessary.

Additionally, ensure you fully qualify graphics from external locations. Keep in mind that all three of these modes take up the entire workspace of the Portal.

The design approach that you take and the type of component that you create depend on the specifications that you outline here.

Building HTML Components

HTML components require no interaction with an external server. They can contain references to applets, ActiveX controls, and images, but the HTML is taken directly from the ActivEra Portal Builder's database. Additionally, HTML components cannot contain multiple pages.

Building HTML components consists of the following topics:

- Creating an HTML component
- Example: Designing an HTML component

See Also

- Planning Portal Components* for more information about determining what kind of component is appropriate for your needs

Creating an HTML Component

You can create components either with the Portal itself or with the OneWorld application, ActivEra Portal Applications Maintenance. Creating an HTML component describes the following tasks:

- Creating an HTML component with the Portal
- Creating an HTML component with OneWorld

▶ **To create an HTML component with the Portal**

Note: Component security is not part of the component creation process in the Portal. For information on defining access to a component, see *Setting Component Permissions* in the *OneWorld System Administration* guide.

1. From any ActivEra Portal workspace, click Personalize on the Secondary Navigation toolbar.
2. Click Components.

The Portal Component Manager appears.

3. Click Add HTML.

The HTML Component Entry Form appears.

4. Complete the following fields, and then click Submit Query:

- Component name
- Description
- Banner Flag

Because the mode icons reside on the component's title bar, hiding the title bar will prevent users from accessing any associated modes or other Portal-enabled component features such as the minimize feature.

- Enter/Edit HTML for this component below

If you use `<A>` tags, the tags should include a `TARGET` reference to a new window so that the Portal is not replaced by the content to which the user navigated.

You can use DHTML code, but be sure to name any controls or JavaScript functions with the following prefix to avoid naming conflicts: `com_companyname_applicationname`.

- Personalize

To display the content of a URL target, choose the URL option, and then enter the URL in the space provided.

To execute JavaScript, enter the complete JavaScript function call (including the parentheses and ending semicolon).

Leave this field blank if you do not want to provide this capability for the component. The associated icon will not appear on the component's title bar.

- Help

To display the content of a URL target, choose the URL option, and then enter the URL in the space provided.

To execute JavaScript, enter the complete JavaScript function call (including the parentheses and ending semicolon).

Leave this field blank if you do not want to provide this capability for the component. The associated icon will not appear on the component's title bar.

- Maximize

To display the content of a URL target, choose the URL option, and then enter the URL in the space provided.

To execute JavaScript, enter the complete JavaScript function call (including the parentheses and ending semicolon).

Leave this field blank if you do not want to provide this capability for the component. The associated icon will not appear on the component's title bar.

Note: Avoid tags such as META, HTML, and BODY. Do not use the FRAMESET tag.

Field	Explanation
Component name	The system name for the component. This name appears in the Component Manager's component list. The maximum field length is 10 characters.
Description	The name for the component as it appears in the component title bar in the ActivEra Portal. The maximum field length is 30 characters.
Banner Flag	Indicates whether to display the component title bar in the ActivEra Portal. The title bar does not appear when this option is checked.
Enter/Edit HTML for this component below	The HTML code (up to 30,000 characters) that defines your component in normal mode.
Personalize	The URL target to display, OneWorld component to instantiate, or JavaScript to execute when the user clicks the Personalize icon. The maximum field length is 256 characters. If you specify a OneWorld component, include the class name only. You can pass parameters in the same way that you do with a URL component—for example: <code>classname?variablename = value</code> .
Help	The URL target to display, OneWorld component to instantiate, or JavaScript to execute when the user clicks the Personalize icon. The maximum field length is 256 characters. If you specify a OneWorld component, include the class name only. You can pass parameters in the same way that you do with a URL component—for example: <code>classname?variablename = value</code> .

Field	Explanation
Maximize	<p>The URL target to display, OneWorld component to instantiate, or JavaScript to execute when the user clicks the Personalize icon. The maximum field length is 256 characters.</p> <p>If you specify a OneWorld component, include the class name only. You can pass parameters in the same way that you do with a URL component—for example: <code>classname?variablename = value</code>.</p>

► **To create an HTML component with OneWorld**

1. From the Solution Explorer, enter P9060 in the Fast Path, and then push Enter on your keyboard.
2. On ActivEra Portal Applications Maintenance, click Components Setup Director.
3. On Components Setup Director, click Next.
4. On Components Definition Revisions, complete the following fields, and then click Next:
 - Component Name
 - Description
 - Component Type

Enter HTML in this field.

 - Banner

Because the mode icons reside on the component's title bar, hiding the title bar will prevent users from accessing any associated modes or other Portal-enabled component features such as the minimize feature.

 - Personalization
 - Help
 - Maximize
5. On HTML Component Entry, complete the following fields, and then click Next:
 - Enter/Edit HTML for this component below

If you use <A> tags, the tags should include a TARGET reference to a new window so that the Portal is not replaced by the content to which the user navigated.

You can use DHTML code, but be sure to name any controls or JavaScript functions with the following prefix to avoid naming conflicts: `com_companyname_applicationname`.

- Personalize

To display the content of a URL target, choose the URL option, and then enter the URL in the space provided.

To execute JavaScript, enter the complete JavaScript function call (including the parentheses and ending semicolon).

Leave this field blank if you do not want to provide this capability for the component. The associated icon will not appear on the component's title bar.

- Help

To display the content of a URL target, choose the URL option, and then enter the URL in the space provided.

To execute JavaScript, enter the complete JavaScript function call (including the parentheses and ending semicolon).

Leave this field blank if you do not want to provide this capability for the component. The associated icon will not appear on the component's title bar.

- Maximize

To display the content of a URL target, choose the URL option, and then enter the URL in the space provided.

To execute JavaScript, enter the complete JavaScript function call (including the parentheses and ending semicolon).

Leave this field blank if you do not want to provide this capability for the component. The associated icon will not appear on the component's title bar.

Note: Avoid tags such as META, HTML, and BODY. Do not use the FRAMESET tag.

6. On Workspace Relationship Revisions, complete the following fields for each user or role for which you want to define access to the component, and then click End:
 - Relationship User/Role
 - Modify Flag

Example: Designing an HTML Component

This example consists of the following processes:

- Creating the component
- Adding JavaScript for the personalization mode

Note: These steps in this example describe creating a component using the Portal. Although the process varies slightly, the HTML code itself is identical whether you are using the Portal or the OneWorld application.

Creating the Component

In this part of the example, you will create an HTML component called Hello OneWorld.

► **To create the component**

1. From any ActivEra Portal workspace, click Personalize on the Secondary Navigation toolbar
2. Click Components.

The Portal Component Manager appears.

3. Click Add HTML.

The HTML Component Entry form appears.

4. Fill out the following fields as indicated:

- Component name

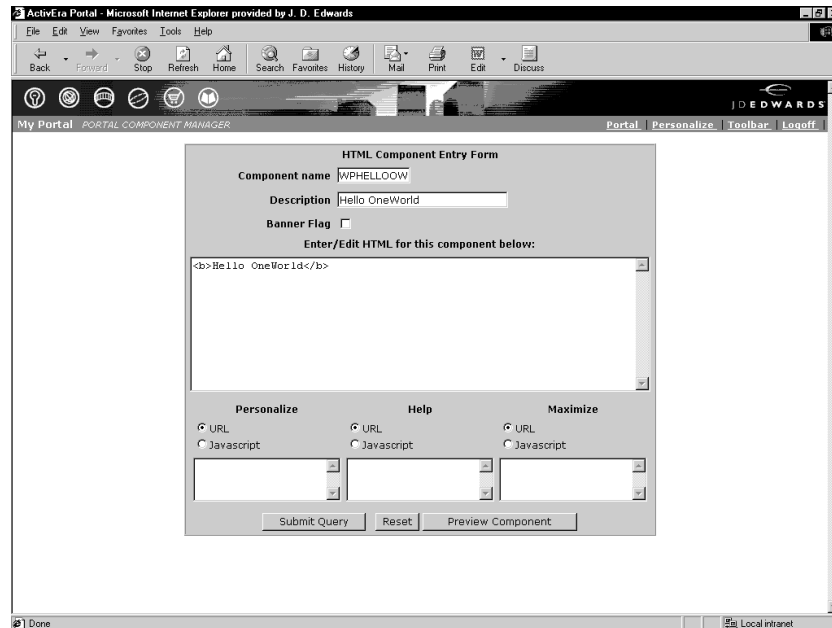
Type : OWPHELLOOW

- Description

Type: Hello OneWorld

- Enter/Edit HTML for this component below

Type: Hello OneWorld



5. Ensure that the remaining fields are blank, and then click Submit Query.

The Component Manager appears. Your new component, OWPHELLOOW, appears in the component list.

6. Test the component by adding it to a workspace.

In the Portal, you should see a component, in boldface, that says, “Hello OneWorld.” The Personalize, Help, and Maximize buttons are not available for your component because you left those sections of the component entry form blank. The next part of the process describes how to enable these modes for the component.

Adding JavaScript for the Personalization Mode

In this part of the component creation process, you will add JavaScript to provide a function for the component’s personalization icon. Although this example is applied to the personalization mode specifically, the process is identical for adding functionality to the help and maximize functions as well.

▶ To add personalization mode JavaScript

1. From any ActivEra Portal workspace, click Personalize on the Secondary Navigation toolbar.
2. Click Components.

The Portal Component Manager appears.

3. Select the OWPHELLOW component that you created at the beginning of this example, and then click Edit Selected.

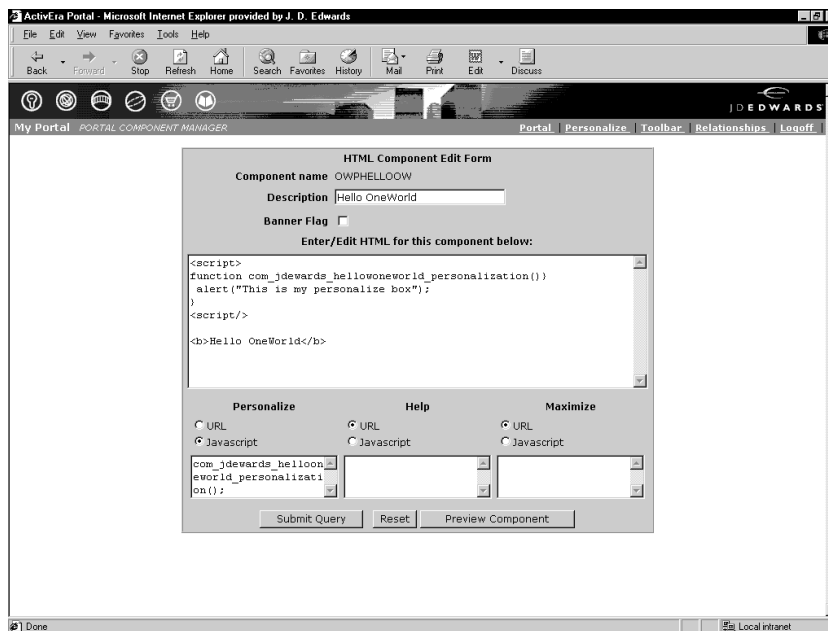
The HTML Component Entry form appears, displaying the code that you have entered for the component to this point.

4. Add the following script to the *Enter/Edit HTML for this component below* field (add the code above the existing code):

```
<script>
function com_jdedwards_hellooneworld_personalization()
alert("This is my personalize box");
}
</script>
```

5. In the Personalize section of the form, select the Javascript option, and then enter the following in the text field:

```
com_jdedwards_hellooneworld_personalization();
```



6. Click Submit Query, and then return to the workspace view of the Portal to test your changes.

A personalization icon has been added to the component's title bar. When you click it, an Alert box appears, displaying a message that reads "This is my personalize box."

Building URI Components

URI components always require interaction between the ActivEra Portal and a web server. These components can be written as CGI, Java servlets, or HTML pages.

Java URI components should be packaged as:

```
com.yourcompany.oneworld.activera.owportal.components:compname
```

where *yourcompany* is the name of your company and *compname* is the name of the component.

Building URI components consists of the following topics:

- Creating a URI component
- Calling URLs
- Wrapping URIs
- Obtaining user information
- Example: Designing a URI component

See Also

- Planning Portal Components* for more information about determining what kind of component is appropriate for your needs

Creating a URI Component

You can create components either with the Portal itself or with the OneWorld application ActivEra Portal Applications Maintenance. Creating a URI component describes the following tasks:

- Creating a URI component with the Portal
- Creating a URI component with OneWorld

► To create a URI component with the Portal

Note: Component security is not part of the component creation process in the Portal. For information on defining access to a component, see *Setting Component Permissions* in the *OneWorld System Administration* guide.

1. From any ActivEra Portal workspace, click Personalize on the Secondary Navigation toolbar.
2. Click Components.

The Portal Component Manager appears.

3. Click Add URI.

The URI Component Entry form appears.

4. Complete the following fields, and then click Submit Query:
 - Component name
 - Description
 - Banner Flag

Because the mode icons reside on the component's title bar, hiding the title bar prevents users from accessing any associated modes or other Portal-enabled component features such as the minimize function.

- Component URL

If you use <A> tags, the tags should include a TARGET reference to a new window so that the Portal is not replaced by the content to which the user navigates.

You can use DHTML code, but be sure to name any controls or JavaScript functions with the prefix `com_companyname_applicationname` to avoid naming conflicts.

- Personalize

To display the content of a URL target, choose the URL option, and then enter the URL in the space provided.

To execute JavaScript, enter the complete JavaScript function call (including the parentheses and ending semicolon).

Leave this field blank if you do not want to provide this capability for the component. The associated icon will not appear on the component's title bar.

- Help

To display the content of a URL target, choose the URL option, and then enter the URL in the space provided.

To execute JavaScript, enter the complete JavaScript function call (including the parentheses and ending semicolon).

Leave this field blank if you do not want to provide this capability for the component. The associated icon will not appear on the component's title bar.

- Maximize

To display the content of a URL target, choose the URL option, and then enter the URL in the space provided.

To execute JavaScript, enter the complete JavaScript function call (including the parentheses and ending semicolon).

Leave this field blank if you do not want to provide this capability for the component. The associated icon will not appear on the component's title bar.

Field	Explanation
Component name	The system name for the component. This name appears in the Component Manager's component list. The maximum field length is 10 characters.
Description	The name for the component as it appears in the component title bar in the ActivEra Portal. The maximum field length is 30 characters.
Banner Flag	Indicates whether to display the component title bar in the ActivEra Portal. The title bar does not appear when this option is checked.
Component URL	URL that points to the normal view of your component. The maximum field length is 256 characters.
Personalize	<p>The URL target to display, OneWorld component to instantiate, or JavaScript to execute when the user clicks the Personalize icon. The maximum field length is 256 characters.</p> <p>If you specify a OneWorld component, include the class name only. You can pass parameters in the same way that you do with a URL component—for example: <code>classname?variablename = value</code>.</p>

Field	Explanation
Help	<p>The URL target to display, OneWorld component to instantiate, or JavaScript to execute when the user clicks the Personalize icon. The maximum field length is 256 characters.</p> <p>If you specify a OneWorld component, include the class name only. You can pass parameters in the same way that you do with a URL component—for example: <code>classname?variablename = value</code>.</p>
Maximize	<p>The URL target to display, OneWorld component to instantiate, or JavaScript to execute when the user clicks the Personalize icon. The maximum field length is 256 characters.</p> <p>If you specify a OneWorld component, include the class name only. You can pass parameters in the same way that you do with a URL component—for example: <code>classname?variablename = value</code>.</p>

► To create a URI component with OneWorld

1. From the Solution Explorer, enter P9060 in the Fast Path, and then push Enter on your keyboard.
2. On ActivEra Portal Applications Maintenance, click Components Setup Director.
3. On Components Setup Director, click Next.
4. On Components Definition Revisions, complete the following fields, and then click Next:

- Component Name
- Description
- Component Type

Enter URI in this field.

- Banner

Because the mode icons reside on the component's title bar, hiding the title bar will prevent users from accessing any associated modes or other Portal-enabled component features such as the minimize feature.

- Personalization
- Help
- Maximize

5. On URI Component Entry, complete the following fields, and then click Next:

- Component URL

If you use <A> tags, the tags should include a TARGET reference to a new window so that the Portal is not replaced by the content to which the user navigates.

You can use DHTML code, but be sure to name any controls or JavaScript functions with the prefix `com_companyname_applicationname` to avoid naming conflicts.

- Personalize

To display the content of a URL target, choose the URL option, and then enter the URL in the space provided.

To execute JavaScript, enter the complete JavaScript function call (including the parentheses and ending semicolon).

Leave this field blank if you do not want to provide this capability for the component. The associated icon will not appear on the component's title bar.

- Help

To display the content of a URL target, choose the URL option, and then enter the URL in the space provided.

To execute JavaScript, enter the complete JavaScript function call (including the parentheses and ending semicolon).

Leave this field blank if you do not want to provide this capability for the component. The associated icon will not appear on the component's title bar.

- Maximize

To display the content of a URL target, choose the URL option, and then enter the URL in the space provided.

To execute JavaScript, enter the complete JavaScript function call (including the parentheses and ending semicolon).

Leave this field blank if you do not want to provide this capability for the component. The associated icon will not appear on the component's title bar.

6. On Workspace Relationship Revisions, complete the following fields for each user or role for which you want to define access to the component, and then click End:
 - Relationship User/Role
 - Modify Flag

Calling URLs

If your component must interact with server-side programs, your client-side code should open a new browser frame from which the client-side code interacts with the server without replacing the Portal's presentation. If the interaction is one-time in nature (such as an HTML form submission) then the server-side program should use the PBSURL request parameter to redirect the resulting response. See *Wrapping URIs* for more information about this parameter.

To support language localization, all URLs entered via the Component Manager can contain the string:

*L;

This string is replaced with the language code string currently in use by the Portal framework. This is known as run-time substitution. The purpose of this is to permit component builders to direct URLs to language-specific directories or files when necessary.

For example, if the user supplied the URL *http://www.myplace.com/*L;/help.html* and the current language code string was ENG, then the URL would be changed to *http://www.myplace.com/ENG/help.html* before it was requested.

Similarly, if the user supplied the URL *http://www.myplace.com/help-*L;.html* and the current language code string was FRE, then the URL would be changed to *http://www.myplace.com/help-FRE.html* before it was requested.

Wrapping URIs

The Portal provides four JavaScript functions that allow you to wrap URIs into the current form. Thus, you can provide multiple pages of material. The four functions are:

- `showPersonalizationPage(compname, uri, addextra)`

Use this function to wrap a URI into the personalize mode of a component.

- `showHelpPage(compname, uri, addextra)`

Use this function to wrap a URI into the help mode of a component.

- `showMaxPage(compname, uri, addextra)`

Use this function to wrap a URI into the maximize mode of a component.

- `showLinkPage(compname, uri, addextra)`

Use this function to wrap an external URI into a component without associating a particular mode with the function. The Portal displays the URI in a full-workspace view, identical to the maximize mode, except that the component title bar does not display a mode name.

The arguments for the functions have the following characteristics:

compname	The name of the component calling the URI. You can obtain this name from a Java servlet; however, if you are using DHTML, you must hard code the component name into the HTML.
uri	The URI to call. Include any parameters (that is, the <code>?parm=value</code> pairs) that are required to call the page.
addextra	An option to send and receive extra parameters. True enables the extra parameters; false disables the feature. These parameters are not user configurable; that is, the component either passes the entire set as described below or not. The parameters are: <ul style="list-style-type: none"> • <code>jsessionid</code> – A reference to the session ID. For the purpose of component design, you always use <code>jsessionid</code> to retrieve the session id of the user, as opposed to <code>sessionId</code> or <code>sessionid</code>. • <code>sessionId</code> – A redundant reference to the session ID. To be removed in future releases. • <code>sessionid</code> – A redundant reference to the session ID. To be removed in future releases. • <code>LANGCODE</code> – The language code identifying the current user language preference. This value currently is not used. • <code>PBSURL</code> – The URI to the ActivEra Portal Builder Servlet. • <code>COMPNAME</code> – The name of the component calling the URI as it exists in the ActivEra Portal Database.

You can also wrap URIs with the Java `ComponentServlet` class. Its `owpWrapForm` JavaScript mirrors the functions above with the advantage of its not requiring the component's ID. The advantage of using URI wrapping, however, is that the URI functions can run in environments that do not support

Java servlets. See *Understanding the ComponentServlet Class* for more information about the ComponentServlet class.

Obtaining User Information

To acquire user name and ID from an HTML or Applet component, use a script tag that references the `/servlet/com.jdedwards.oneworld.owportal.js.PortalScript` servlet. This tag returns a script called `JDEUserData` which provides the following information:

JDEUserData.name	The user's name. If OneWorld Name is unavailable, this variable returns the user's ID instead.
JDEUserData.address	The user's address book number.
JDEUserData.userID	The user's OneWorld ID.

For example, consider the following code fragment:

```
<html>
<body>
<script language="JavaScript" type="text/javascript"
src="/servlet/com.jdedwards.oneworld.owportal.js.PortalScript">
</script>
<script>
document.write("Hello"+document.JDEUserData.userID);
</script>
</body>
</html>
```

If a user logged in as SO1234567, the above code would display the following string:

```
Hello SO1234567
```

Example: Designing a URI Component

This example consists of the following processes:

- Creating the component
- Wrapping URLs in personalization mode

Note: The steps in this example describe creating a component using the Portal. Although the process varies slightly, the code itself is identical whether you are using the Portal or the OneWorld application.

Before You Begin

- Create a text file on a web server that contains the following text:

```
<html>
<body>
<script>
/*
 * This function parses &-separated name=value pairs.
 /

function com_jdedwards>HelloOneWorld_getArgs(){
  var args = new Object();
  var query = location.search.substring(1);
  var pairs = query.split("&");
  for (var i = 0; i < pairs.length; i++){
    var pos = pairs[i].indexOf('=');
    if (pos == -1) continue;
    var argname = pairs[i].substring(0,pos);
    var value = pairs[i].substring(pos+1);
    args[argname]=unescape(value);
  }

  var args = com_jdedwards>HelloOneWorld_getArgs();
  if (args.mode == null)
    document.writeln("<b>Hello OneWorld</b>");
  else if (args.mode=="personalize")
    document.writeln("This is my personalization screen for Hello
OneWorld");
</script>
</body>
</html>
```

When the page is loaded into the browser, the JavaScript is executed. First it retrieves the arguments from the URL. If the mode argument does not exist, then “Hello OneWorld” is displayed in bold letters. If mode = personalize, then “This is my personalization screen for OneWorld” is displayed instead.

To test the functionality of the component, save your work and point your browser to the URL of the component. The screen should display “Hello OneWorld” in bold letters. To test the personalize function, add “?mode=personalize” to the URL. Now the page should display “This is my personalization screen for OneWorld.”

Creating the Component

The first step in the process of designing a URI component is to create the actual URI component.

► **To create the component**

1. From any ActivEra Portal workspace, click Personalize.
2. Click Components.

The Portal Component Manager appears.

3. Click Add URL.

The URI Component Entry form appears.

4. Complete the following fields as indicated:

- Component Name

Type: OWPHELLOO2

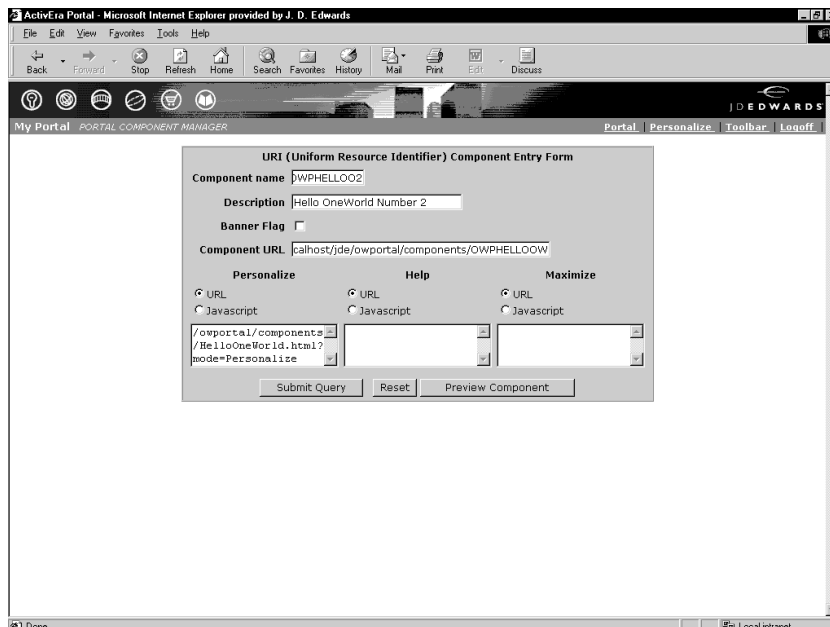
- Description

Type: Hello OneWorld Number 2

- Component URL

Enter the URL of the text file that you created before you started this process. Even though you did not create the file using Portal tools, it is technically a component.

5. Choose the URL option in the Personalize section, and then enter the same URL that you entered in the Component URL field with ?mode=personalize appended to the end of it.



6. Leave the remaining fields blank, and then click Submit Query.

The Component Manager appears. Your new component, OWPHELLOW2, appears in the component list.

7. Test the component by adding it to a workspace.

In the Portal, you should see a component, in boldface, that says “Hello OneWorld Number 2.” Click the Personalize icon to see the message “This is my personalization screen for OneWorld.”

Wrapping URLs in Personalization Mode

Modify the OWPHHELLOO2 file so that it reads as follows:

```

<html>
<body>
<script>
/*
 * This function parses &-separated name=value pairs.
 */
function com_jdedwards_HelloOneWorld_getArgs(){
var args = new Object();
var query = location.search.substring(1);
var pairs = query.split("&");
for (var i = 0; i < pairs.length; i++){
var pos = pairs[i].indexOf('=');
if (pos == -1) continue;
var argname = pairs[i].substring(0,pos);
var value = pairs[i].substring(pos+1);
args[argname]=unescape(value);
}
}
function com_jdedwards_HelloOneWorld_show(){
if (document.ActiveEraPortal != null)
showPersonalizationPage(args.COMPNAME, "http://localhost/jde/owportal/component
s/HelloOneWorld.html?mode=personalize2", true);
else
window.location="http://localhost/jde/owportal/components/HelloOneWorld.html?m
ode=personalize2";
}

var args = com_jdedwards_HelloOneWorld_getArgs();
if (args.mode == null)
document.writeln("<b>Hello OneWorld</b>");
else if (args.mode=="personalize")
{
document.writeln("<p>This is my personalization screen for Hello
OneWorld.</p>");
document.writeln('<p><a href="JavaScript:
com_jdedwards_HelloOneWorld_show()">>');
document.writeln("Goto Personalize 2</a></p>");
}
else if (args.mode=="personalize2")
{
document.writeln("<p>This is the second personalization page</p>");
document.writeln("<p> jsessionid="+args.jsessionid+"<br>");
document.writeln("LANGCODE="+args.LANGCODE+"<br>");
document.writeln("PBSURL="+args.PBSURL+"<br>");
document.writeln("COMPNAME="+args.COMPNAME+"<br></p>");
}
}
</script>
</body>
</html>

```

Note that the `com_jdedwards_HelloOneWorld_show()` function checks to see if the calling component is currently displayed in the ActivEra Portal. If it is, then `document.ActiveEraPortal` will never be null. If it is not, then `document.ActiveEraPortal` will be null. This check is necessary to support the J.D. Edwards standard that Portal components be able to function as well alone as within the Portal. Because the JavaScripts used in this example are specific to the Portal, you should always check to ensure that the ActivEra Portal is

displaying an object before using one of these scripts. In this example, if the ActivEra Portal is unavailable, the code calls the URL directly.

After the `com_jdedwards_HelloOneWorld_show()` function checks for the ActivEra Portal and finds that it is available, it then executes the `showPersonalizationPage` function. Notice that the component name can be obtained from the arguments because the `COMPNAME` is provided to the personalization page by default. The second argument references a text file called `HelloOneWorld.html`, and the last argument tells the ActivEra Portal to send the extra arguments to the component.

This example also displays the second personalization page when `mode=personalize2`. This function lists a page that displays all of the parameters provided by the ActivEra Portal Builder Servlet.

Typically, you will want to gather the extra component information as in this example; however, depending on the source of the URI, you might not always find it advantageous to do so. For example, you might have a component that provides data and exists on a foreign host such as Yahoo that returns server-generated errors if it receives invalid parameters (as a protection against hacking). In this scenario, passing the extra parameters to a the foreign host might prevent this page from loading. Of course, if the information is not provided, then the component will know nothing about itself and might not be able to display additional information in the ActivEra Portal.

The `showMaxPage`, `showHelpPage`, and `showLinkPage` functions work the same way as the `showPersonalizationPage` function in this example. Additionally, you set up `showFramedLinkPage` as you do `showPersonalizationPage`. However, `showFramedLinkPage` displays the component and the Portal's two navigation bars each in their own frame.

Building OneWorld Components

OneWorld components are specialized URI components that you must use if you want:

- To use a OneWorld application.
- To reference or update the session object.
- To send and receive data from the database using the `ComponentDatabaseInterface` or any other Database access through the OneWorld API.

Like URI components, OneWorld components always require interaction between the ActivEra Portal and a web server. These components must be written in Java as OneWorld component servlets.

Java OneWorld components should be packaged as:

```
com.jedwards.oneworld.owportal.components.compname
```

Building OneWorld components is similar to building URI components. Instead of providing a fully-qualified URI, however, you provide a class name that is accessible to the `PortalBuilderServlet` through the server's classpath. Use the same method to pass parameters to the OneWorld component that you use to pass parameters from a browser. For example, consider the following URL:

```
http://myserver.com/servlets/com.jdedwards.oneworld.owportal.  
components.componentName.MyServlet?mode=personalize
```

To declare a OneWorld component from with the Portal, you might define this URL in this way:

```
com.jdedwards.oneworld.owportal.components.componentName.MyServlet?  
mode=personalize
```

Building OneWorld components contains the following task:

- Creating a OneWorld component

See Also

- Planning Portal Components* for more information about determining what kind of component is appropriate for your needs

- Calling URLs
- Wrapping URIs
- Writing Java Servlets and OneWorld Components

Creating a OneWorld Component

You can create components either with the Portal itself or with the OneWorld application ActivEra Portal Applications Maintenance. Creating a OneWorld component describes the following tasks:

- Creating a OneWorld component with the Portal
- Creating a OneWorld component with OneWorld

► **To create a OneWorld component with the Portal**

Note: Component security is not part of the component creation process in the Portal. For information on defining access to a component, see *Setting Component Permissions* in the *OneWorld System Administration* guide.

1. From any ActivEra Portal workspace, click Personalize on the Secondary Navigation toolbar.
2. Click Components.

The Portal Component Manager appears.

3. Click Add OneWorld.
4. On OneWorld Component Entry, complete the following fields, and then click Submit Query:
 - Component name
 - Description
 - Banner Flag

Because the mode icons reside on the component's title bar, hiding the title bar prevents users from accessing any associated modes or other Portal-enabled component features such as the minimize function.

- Class Name
- Personalize

To display a OneWorld component, choose the OneWorld option, and then enter its class name.

To display the content of a URL target, choose the URL option, and then enter the URL in the space provided.

To execute JavaScript, enter the complete JavaScript function call (including the parentheses and ending semicolon).

Leave this field blank if you do not want to provide this capability for the component. The associated icon will not appear on the component's title bar.

- Help

To display a OneWorld component, choose the OneWorld option, and then enter its class name.

To display the content of a URL target, choose the URL option, and then enter the URL in the space provided.

To execute JavaScript, enter the complete JavaScript function call (including the parentheses and ending semicolon).

Leave this field blank if you do not want to provide this capability for the component. The associated icon will not appear on the component's title bar.

- Maximize

To display a OneWorld component, choose the OneWorld option, and then enter its class name.

To display the content of a URL target, choose the URL option, and then enter the URL in the space provided.

To execute JavaScript, enter the complete JavaScript function call (including the parentheses and ending semicolon).

Leave this field blank if you do not want to provide this capability for the component. The associated icon will not appear on the component's title bar.

Field	Explanation
Component name	The system name for the component. This name appears in the Component Manager's component list. The maximum field length is 10 characters.
Description	The name for the component as it appears in the component title bar in the ActivEra Portal. The maximum field length is 30 characters.

Field	Explanation
Banner Flag	Indicates whether to display the component title bar in the ActivEra Portal. The title bar does not appear when this option is checked.
Class Name	The location of the OneWorld component. Class name is similar to a URL, but does not require full qualification and consequently uses the following format: com.jedwards.oneworld.owportal.components. compname
Personalize	The URL target to display, OneWorld component to instantiate, or JavaScript to execute when the user clicks the Personalize icon. The maximum field length is 256 characters. If you specify a OneWorld component, include the class name only. You can pass parameters in the same way that you do with a URL component—for example: classname?variablename = value.
Help	The URL target to display, OneWorld component to instantiate, or JavaScript to execute when the user clicks the Personalize icon. The maximum field length is 256 characters. If you specify a OneWorld component, include the class name only. You can pass parameters in the same way that you do with a URL component—for example: classname?variablename = value.
Maximize	The URL target to display, OneWorld component to instantiate, or JavaScript to execute when the user clicks the Personalize icon. The maximum field length is 256 characters. If you specify a OneWorld component, include the class name only. You can pass parameters in the same way that you do with a URL component—for example: classname?variablename = value.

► To create a OneWorld component with OneWorld

1. From the Solution Explorer, enter P9060 in the Fast Path, and then push Enter on your keyboard.
2. On ActivEra Portal Applications Maintenance, click Components Setup Director.
3. On Components Setup Director, click Next.
4. On Components Definition Revisions, complete the following fields, and then click Next:
 - Component Name

- Description
- Component Type

Enter OneWorld in this field.

- Banner

Because the mode icons reside on the component's title bar, hiding the title bar will prevent users from accessing any associated modes or other Portal-enabled component features such as the minimize feature.

- Personalization
- Help
- Maximize

5. On OneWorld Component Entry, complete the following fields, and then click Next:

- Class Name

The location of the OneWorld component. Class name is similar to a URL, but does not require full qualification and consequently uses the following format:

`com.jedwards.oneworld.owportal.components.compname`

- Personalize

To display a OneWorld component, choose the OneWorld option, and then enter its class name.

To display the content of a URL target, choose the URL option, and then enter the URL in the space provided.

To execute JavaScript, enter the complete JavaScript function call (including the parentheses and ending semicolon).

Leave this field blank if you do not want to provide this capability for the component. The associated icon will not appear on the component's title bar.

- Help

To display a OneWorld component, choose the OneWorld option, and then enter its class name.

To display the content of a URL target, choose the URL option, and then enter the URL in the space provided.

To execute JavaScript, enter the complete JavaScript function call (including the parentheses and ending semicolon).

Leave this field blank if you do not want to provide this capability for the component. The associated icon will not appear on the component's title bar.

- Maximize

To display the content of a URL target, choose the URL option, and then enter the URL in the space provided.

To execute JavaScript, enter the complete JavaScript function call (including the parentheses and ending semicolon).

Leave this field blank if you do not want to provide this capability for the component. The associated icon will not appear on the component's title bar.

6. On Workspace Relationship Revisions, complete the following fields for each user or role for which you want to define access to the component, and then click End:
 - Relationship User/Role
 - Modify Flag

Building Java Applet Components

When you create an applet component, you enter all of the parameters necessary for the APPLET tag and any PARAM tag information into the Applet Component Entry form. If you want to use the toolbar icons to invoke maximize, personalize, or help functionality, you must make these functions part of the applet's public interface. The ActivEra Portal framework provides the JavaScript, which invokes your applet's functions whenever you click one of the component's toolbar icons.

Your applet may change its interface to reflect the new state (help or personalize) but you will probably want to generate a new frame window in response to the maximize function invocation. Applets can communicate with the server, from which they were served via HTTP or RMI. Applets can communicate with servlets or other server-side programs so that the applet can display databased information or other dynamic information.

You can create components either with the Portal itself or with the OneWorld application, ActivEra Portal Applications Maintenance. Building Java applet components describes the following tasks:

- Building Java applet components with the Portal
- Building Java applet components with OneWorld

To build Java applet components with the Portal

Note: Component security is not part of the component creation process in the Portal. For information on defining access to a component, see *Setting Component Permissions* in the *OneWorld System Administration* guide.

1. From any ActivEra Portal workspace, click Personalize.
2. Click Components.
3. On Portal Component Manager, click Add Applet.
4. On Applet Component Entry, complete the following fields, and then click Submit Query:
 - Component name
 - Description
 - Banner Flag

Because the mode icons reside on the component's title bar, hiding the title bar prevents users from accessing any associated modes or other Portal-enabled component features such as the minimize feature.

- Codebase
- Code
- Alt (Alternate HTML if applets are not enabled (255 chars max))

Leave this field blank unless you are using HTML version 4.0 or later.

- Width
- Height
- Align
- Vspace
- Hspace
- Archive
- Personalize

To display the content of a URL target, choose the URL option, and then enter the URL in the space provided.

To invoke an applet, enter its public function name. The response to the function call depends on the applet and might include opening additional Java frame windows as the user interacts with the applet.

To execute JavaScript, enter the complete JavaScript function call (including the parentheses and ending semicolon).

Leave this field blank if you do not want to provide this capability for the component. The associated icon will not appear on the component's title bar.

- Help

To display the content of a URL target, choose the URL option, and then enter the URL in the space provided.

To invoke an applet, enter its public function name. The response to the function call depends on the applet and might include opening additional Java frame windows as the user interacts with the applet.

To execute JavaScript, enter the complete JavaScript function call (including the parentheses and ending semicolon).

Leave this field blank if you do not want to provide this capability for the component. The associated icon will not appear on the component's title bar.

- Maximize

To display the content of a URL target, choose the URL option, and then enter the URL in the space provided.

To invoke an applet, enter its public function name. The response to the function call depends on the applet and might include opening additional Java frame windows as the user interacts with the applet.

To execute JavaScript, enter the complete JavaScript function call (including the parentheses and ending semicolon).

Leave this field blank if you do not want to provide this capability for the component. The associated icon will not appear on the component's title bar.

- Parameters; name=val | name=val | val | name=val,...(No spaces, NO CRs)

Field	Explanation
Component name	The system name for the component. This name appears in the Component Manager's component list. The maximum field length is 10 characters.
Description	The name for the component as it appears in the component title bar in the ActivEra Portal. The maximum field length is 30 characters.
Banner Flag	Indicates whether to display the component title bar in the ActivEra Portal. The title bar does not appear when this option is checked.
Make Public	Indicates whether to allow all users to select the component to add to their workspaces.
Codebase	The codebase argument of the applet. The maximum field length is 256 characters.
Code	The code argument of the applet. The maximum field length is 256 characters.
Alt	The alt argument of the applet. The maximum field length is 256 characters.
Width	The width argument of the applet. The maximum field length is 4 digits.

Field	Explanation
Height	The height argument of the applet. The maximum field length is 4 digits.
Align	The align argument of the applet. The maximum field length is 10 digits.
Vspace	The vspace argument of the applet. The maximum field length is 4 digits.
Hspace	The hspace argument of the applet. The maximum field length is 4 digits.
Archive	The archive argument of the applet. This argument is valid only for HTML version 4.0 and later. The maximum field length is 256 characters.
Personalize	<p>The URL target to display or JavaScript to execute when the user clicks the Personalize icon. The maximum field length is 256 characters.</p> <p>If you specify a OneWorld component, include the class name only. You can pass parameters in the same way that you do with a URL component—for example: <code>classname?variablename = value.</code></p>
Help	<p>The URL target to display or JavaScript to execute when the user clicks the Help icon. The maximum field length is 256 characters.</p> <p>If you specify a OneWorld component, include the class name only. You can pass parameters in the same way that you do with a URL component—for example: <code>classname?variablename = value.</code></p>
Maximize	<p>The URL target to display or JavaScript to execute when the user clicks the Maximize icon. The maximum field length is 256 characters.</p> <p>If you specify a OneWorld component, include the class name only. You can pass parameters in the same way that you do with a URL component—for example: <code>classname?variablename = value.</code></p>
Parameters	The param value or values that follow the applet. Each value must be expressed in the form of name argument = value argument. To enter more than one param value, separate each value with a pipe (). Do not use spaces or carriage returns in this field. This field accepts up to 30,000 characters.

 **To build Java applet components with OneWorld**

1. From the Solution Explorer, enter P9060 in the Fast Path, and then push Enter on your keyboard.

2. On ActivEra Portal Applications Maintenance, click Component Setup Director.
3. On Components Setup Director, click Next.
4. On Components Definition Revisions, complete the following fields, and then click Next:

- Component Name
- Description
- Component Type

Enter APPLET in this field.

- Banner

Because the mode icons reside on the component's title bar, hiding the title bar will prevent users from accessing any associated modes or other Portal-enabled component features such as the minimize feature.

- Personalization
- Help
- Maximize

5. On Applet Component Entry, complete the following fields, and then click Next:

- Codebase
- Code
- Alt (Alternate HTML if applets are not enabled (255 chars max))

Leave this field blank unless you are using HTML version 4.0 or later.

- Width
- Height
- Align
- Vspace
- Hspace
- Archive
- Personalize

To display the content of a URL target, choose the URL option, and then enter the URL in the space provided.

To invoke an applet, enter its public function name. The response to the function call depends on the applet and might include opening additional Java frame windows as the user interacts with the applet.

To execute JavaScript, enter the complete JavaScript function call (including the parentheses and ending semicolon).

Leave this field blank if you do not want to provide this capability for the component. The associated icon will not appear on the component's title bar.

- Help

To display the content of a URL target, choose the URL option, and then enter the URL in the space provided.

To invoke an applet, enter its public function name. The response to the function call depends on the applet and might include opening additional Java frame windows as the user interacts with the applet.

To execute JavaScript, enter the complete JavaScript function call (including the parentheses and ending semicolon).

Leave this field blank if you do not want to provide this capability for the component. The associated icon will not appear on the component's title bar.

- Maximize

To display the content of a URL target, choose the URL option, and then enter the URL in the space provided.

To invoke an applet, enter its public function name. The response to the function call depends on the applet and might include opening additional Java frame windows as the user interacts with the applet.

To execute JavaScript, enter the complete JavaScript function call (including the parentheses and ending semicolon).

Leave this field blank if you do not want to provide this capability for the component. The associated icon will not appear on the component's title bar.

- Parameters; name=val | name=val | val | name=val,...(No spaces, NO CRs)

6. On Workspace Relationship Revisions, complete the following fields for each user or role for which you want to define access to the component, and then click End:
 - Relationship User/Role
 - Modify Flag

Building Java Servlet Components

In the ActivEra Portal, JavaScript code can use the open function to create new browser windows, invoke the functionality of applets or objects contained in the user entered HTML, and so forth. Script-generated frame windows can contain dynamically generated HTML for the user to interact with. Alternately, the SRC parameter in the open call can reference a URL whose output makes up the content of the new window. Whichever path you choose, make sure that your functionality does not impose itself on the Portal presentation. If your client-side code must interact with server side programs, it should open a new browser frame from which it interacts with the server without replacing the Portal's presentation.

The ActivEra Portal provides the ActivEra Portal JAVA Component API. To gain access to the API, you must have the jar file for the ActivEra Portal in your class path, and then you must import *com.jdedwards.oneworld.owportal.components.** into your class. Currently, this package includes the following four classes:

ComponentServlet

An abstract class that handles template loading, loading of the OneWorld Java Server, and some template parsing for some standard keys and fields, including some JavaScript that facilitates implementation of the functions showPersonalizationPage, showMaxPage, showHelpPage, and showLinkPage.

Note: The Portal Builder expects that all servlet components will be derived from the ComponentServletClass.

ComponentTemplateParser

Provides access to template parsing facilities for your servlet, which allow you to use standard key replacement, tag replacement, and other functions for handling HTML templates in your servlet.

ComponentDatabaseInterface

Provides a method for storing data to and retrieving data from a JAS database. The package also provides a convenient way to import and export data to and from Sun's XML parser so that you can easily use data with the DOM.

ComponentDatabaseException An exception model for database errors generated by the ComponentDatabaseInterface.

All JavaScripts within your component should be named:

```
com_yourcompany_yourcomponent_scriptname
```

where *yourcompany* is the name of your company and *yourcomponent* is the name of your component.

All other files (non-JavaScript files) should reside in directories named:

```
yourcompany/owportal/components/yourcomponent
```

where *yourcompany* is the name of your company and *yourcomponent* is the name of your component.

Building Java components consists of the following topics:

- Understanding the ComponentServlet class
- Understanding the ComponentTemplateParser class
- Understanding the ComponentDatabaseInterface class

Understanding the ComponentServlet Class

The ComponentServlet allows your component to run under the JAS as well as to function with the ActivEra Portal. If you extend this class when you make your own components, you can take advantage of J. D. Edwards enhancements that will help you integrate into other components or allow your component to function in a stand-alone environment.

Understanding the ComponentServlet class consists of the following topics:

- Extending the component servlet
- Understanding the loadTemplate() method
- Understanding the OwpWrapForm script
- Understanding the OwpWrapPage script
- Understanding the OwpTargetedPage script

Extending the Component Servlet

The following servlet is the minimum servlet required to extend the component servlet correctly:

```
package com.jdedwards.oneworld.owportal.components.helloWorld;
import com.jdedwards.oneworld.owportal.components.*;
class HelloWorldServlet extends ComponentServlet
{
protected void service(HttpServletRequest req,
HttpServletResponse res)throws ServletException, IOException
{
super.service(req, res);
PrintWriter out = res.getWriter();
HTMLWEnv owEnv =
PortalBase.jdeServletComponentLogin(this, req, res, mySplash, false);
if ( owEnv == null )
{
out.close();
return;
}
out.println(getTitle());
out.close();
}
public String getTitle()
{
return "Hello OneWorld";
}
}
```

First, the service class must be implemented by every servlet. It is part of the standard Java Servlet API. The service method calls the same method in the super class. By including this call in your code, you ensure that your component has access to a OneWorld Java Server. You will also automatically set up any required interfaces to other components.

Second, the servlet gets a writer from the `HttpServletResponse.getWriter`. This allows you to write data to the output stream so that it will appear in the user's browser.

Third, the servlet gets the OneWorld environment that was referenced by the `jsessionId`. It is important to understand that the ActivEra Portal retrieves information from a URI via a web server, and therefore requires two session IDs. One session ID is the one that the servlet was called with; the other is the session ID that the ActivEra Portal Builder Servlet was called with. If you need to obtain information about the user and his or her login, you must get the OneWorld environment for the ActivEra Portal Builder with the `PortalBase.jdeServletComponentLogin` method. If the user has not logged in, the `jdeServletComponentLogin` will return null to the servlet and will then proceed to log the user into the system in its own thread.

Fourth, the servlet checks to see if it received information from the login. If not, that indicates that the user was not logged in. The servlet closes the output stream and terminates.

Finally, the servlet uses the `getTitle` method to return the name of the current component to an output string. Every child of the `ComponentServlet` class must override this method. It will help you access other portals, the `ComponentServlet`, or any object that the component servlet assumes exists.

See Also

- ❑ *Wrapping URIs* for more information about the extra parameter, `jsessionId`

Understanding the `loadTemplate()` Method

The `loadTemplate` method in the `ComponentServlet` class loads a template from an input stream or a URI. A template is a document encoded with a special version of HTML that allows data to be embedded in the document. For example, a template file might look like the following (assuming the component name is `mycomp`):

```
<html>
<head>
<title><PageDescription/></title>
<Common.Scripts.All>
function com_jdedwards_mycomp_getOneWorldPortal()
{
return (document.ActiveEraPortal != null)?true:false;
}
</Common.Scripts.All>
</head>
<body>
<a href="http://{#jas}/jde/owportal">Click Here</a>
</body>
</html>
```

This example shows that, with a few exceptions, templates look much like HTML documents. For instance, some tags such as `<Common.Scripts.All>` are not standard to HTML. Additionally, the file includes XML entities such as `<PageDescription/>` and a key, `{#jas}`, within one of the tags. Because the Portal cannot interpret templates, you must use the `loadTemplate` method to convert the template into an HTML format that it can understand. When `loadTemplate` reads this file, you can instruct it to modify it, as follows:

```

<html>
<head>
<title>Hello OneWorld</title>
<script language="JavaScript">
function mycompOwpWrapForm(url, myForm, addextra)
{
    ...
}
function mycompOwpWrapPage(url, addextra)
{
    ...
}
function mycompOwpTargetedPage (mylink)
{
    ...
}
function com_jdedwards_mycomp_getOneWorldPortal()
{
return (document.ActiveEraPortal != null)?true:false;
}
</script>
</head>
<body>
<a href="http://www.mysite.com/jde/owportal">Click Here</a>
</body>
</html>

```

To accomplish this conversion, the `loadTemplate` method uses the `ComponentTemplateParser` class to return a standard HTML template with a few additions that Portal components can use. See *Understanding the ComponentTemplateParser Class* for more information about how the `ComponentTemplateParser` class works.

You can call `loadTemplate` in any of the following ways:

- Stream `loadTemplate` functions
 - `loadTemplate (InputStream, HttpServletRequest);`
 - `loadTemplate (InputStream, HttpServletRequest, boolean);`
- URI `loadTemplate` functions
 - `loadTemplate (String, HttpServletRequest);`
 - `loadTemplate (String, HttpServletRequest, boolean);`

The stream functions can accept an `InputStream` of any source, including a file, an HTML stream, or even an input stream from the `ComponentDatabaseInterface` class. See *Understanding the ComponentDatabaseInterface Class* for more information about using this class. The URI functions, on the other hand, accept a URI of a template, create their own http stream, and then call their corresponding stream functions.

Two of the calls to `loadTemplate` contain Boolean operands. If set to true, `loadTemplate` converts the template to an HTML file as demonstrated at the beginning of this topic. If set to false, then the `loadTemplate` function returns the template without any replacements. The two calls without Boolean

operands assume that you want to convert the template; that is, they function in the same way as their Boolean-enabled counterparts with the boolean argument set to true.

The conversion includes replacing two keys and four tags. Keys are used for inter-tag replacement. For example, `{#jas}` in the template example above is a key. Keys are case-sensitive, cannot include any white spaces, and are always in the form of `{#key}`. Tags, on the other hand, cannot be embedded inside of other tags. Tags are not case-sensitive, can include white spaces, and can either be in the form of `<tag></tag>` or `<tag/>`. Tags from the template example above are `<PageDescription/>` and `<Common.Scripts.All> </Common.Scripts.All>`. Also notice that each tag must either have a `<tag></tag>` equivalent or have a `<tag/>` equivalent. You cannot perform `<tag>` replacements.

The following tags and keys constitute the standard replacements:

- Keys
 - `{#jas}`

This key is replaced with the location of the jde server.
 - `{#portalurl}`

This key is replaced with the URL of the Portal.
 - `{#componentname}`

This key is replaced with the name of the component as specified by the name tag in the Portal Builder.
- Tags
 - `<Common.Scripts.All/>`

This tag is replaced with all of the scripts from `<Common.Scripts.owpWrapForm/>`, `<Common.Scripts.owpWrapPage/>`, and `<Common.Scripts.owpTargetedPage/>`. These scripts support Netscape 4 and Microsoft Internet Explorer 4 and allow greater integration between the Portal and the component without making the component dependent on the Portal.
 - `<Common.Scripts.owpWrapForm/>`

This tag is replaced with a script that accepts a URL and an HTML form as data. If the component is running within the ActivEra Portal, the script builds the URL from the fields in the HTML form using the get method. It then submits the URL to the Portal Builder's standard JavaScript routine, which wraps the page referenced in the URL in a Portal Builder interface. The data

contained in the HTML form is submitted via a get method to the URL that you specify. You can also hard code some get method parameters in the URL because this script determines whether a ? symbol already exists in the URL.

If the servlet is being run outside of the ActivEra Portal, the script submits the HTML form using the method that is specified in the HTML form header. The data for this tag replacement is taken from the protected string `getOwpWrapForm()` method in the `ComponentServlet` class. See *Understanding the OwpWrapForm Script* for more information.

- `<Common.Scripts.owpWrapPage/>`

This tag behaves the same as the `OwpWrapForm` JavaScript function except that it does not append HTML form data. If the component is being displayed in the Portal, the provided URL is sent to the Portal's standard JavaScript function. Otherwise, the URL is called directly. See *Understanding the OwpWrapPage Script* for more information.

- `<Common.Scripts.OwpTargetedPage/>`

This tag behaves the same as the `OwpWrapPage` JavaScript function except that it handles a target. The specified target can be any valid target; however, a target of `_portal` will be wrapped in the Portal Builder if the servlet is running in the Portal. If the servlet is not running in the Portal, then the tag will act the same as `_top`. See *Understanding the OwpTargetedPage Script* for more information.

Unless specifically noted above, all tag replacements also include the unmodified tag contents. To illustrate, the tag

```
<Common.Scripts.OwpTargetedPage>
document.write("me");</Common.Scripts.OwpTargetedPage>
```

would be replaced with this code (assuming that the component name is `mycomp`):

```
<SCRIPT language="JavaScript">
function mycompOwpTargetedPage (url, target){
...
}
document.write("me");
</SCRIPT>
```

The tags themselves (`<Common.Scripts.owpTargetedPage>` and `</Common.Scripts.OwpTargetedPage>`) would be replaced as shown, but the body (`document.write("me")`) would be carried over intact.

The following is another example of a base template before and after processing by load Templates.

HelloWorldServlet before processing

```
package com.jdedwards.oneworld.owportal.components.helloWorld;
import com.jdedwards.oneworld.owportal.components.*;
class HelloWorldServlet extends ComponentServlet
{
protected void service(HttpServletRequest req,
HttpServletResponse res)throws ServletException, IOException
{
super.service(req, res);
PrintWriter out = res.getWriter();
HTMLWEnv owEnv =
PortalBase.jdeServletComponentLogin(this, req, res, mySplash, false);
if ( owEnv == null )
{
out.close();
return;
}
out.println(getTitle());
out.close();
}
public String getTitle()
{
return "Hello OneWorld";
}
}
```

HelloWorldServlet after processing

```
package com.jdedwards.oneworld.owportal.components.helloWorld;
import com.jdedwards.oneworld.owportal.components.*;
class HelloWorldServlet extends ComponentServlet
{
protected void service(HttpServletRequest req,
HttpServletResponse res)throws ServletException, IOException
{
super.service(req, res);
PrintWriter out = res.getWriter();
HTMLWEnv owEnv =
PortalBase.jdeServletComponentLogin(this, req, res, mySplash, false);
if ( owEnv == null )
{
out.close();
return;
}
if (mypage==null)
myPage=loadTemplate("http://www.mysite.com/mytemplate.htm", req);
out.println(myPage);
out.close();
}
public String getTitle()
{
return "Hello OneWorld";
}
static String myPage;
}
```

Notice that the `myPage` variable that stores the template is declared as static. This way the standard replacements and the template only have to be loaded once, after which they are cached by the application server.

Understanding the `OwpWrapForm` Script

The `OwpWrapForm` script is used to submit the contents of an HTML form so that the HTML form can be wrapped into the Portal. The function has the following format:

```
componentnameOwpWrapForm(url, Form, addExtra, type)
```

- *componentname* is the system name of the component (not its description).
- `url` is the URL of the object to which you would like to submit the form. You can include `?property=value` properties in the URL.
- `Form` is the name of the HTML form.
- `addExtra` is a Boolean value specifying whether the extra fields should be added to the URL. The following is a list of the extra fields:

- `jsessionid` (contains the Portal Builder Servlet's session ID string)

This parameter allows components to dynamically generate relevant state-based content.

- `STATE` (contains one of the following strings: MIN, MAX, PER, HLP, RES, FRM)

This parameter aids in component language localization. MIN stands for minimized mode, MAX stands for maximized or link mode, PER stands for personalize mode, HLP stands for help mode, and RES stands for normal mode. These codes are subject to change in future releases; consequently, you might want to create your own vehicle for returning component mode.

- `LANGCODE` (contains the language code string currently in use in the portal environment)

This parameter allows a component to redirect to the `PortalBuilderServlet` when necessary.

- `PBSURL` (contains the URL of the `PortalBuilderServlet`)
- `type` specifies the mode in which the URL target should be displayed. Valid types are Personalize, Maximize, Help, or Link. It also checks to make sure that the component is being run from the ActivEra Portal.

The script is already aware of the component name, so component name is not a required argument.

You can also wrap URIs with the four JavaScript functions, `showPersonalizationPage`, `showHelpPage`, `showMaxPage`, and `showLinkPage`. These functions mirror the `OwpWrapForm` JavaScript with the advantage of the ability to run in environments that do not support Java servlets. The advantage of using the Java servlet, however, is that the servlet does not require the component's ID. See *Wrapping URIs* for more information about these four JavaScript functions.

The following is an example of the `OwpWrapForm` script. If the component is inside the ActivEra Portal, the script executes the `showPersonalizationPage` method with the following URL:

```
http://www.mysite.com/test.html?id=7&mode=me
```

If the component is not in the ActivEra Portal, the script executes the URL directly.

```
<html>
<head>
<Common.Scripts.OwpWrapForm/>
</head>
<body>
<form name=test>
  <field type=hidden name=mode value=me>
</form>
<a
href="javascript:{#componentname}OwpWrapForm('http://www.mysite.com/test.html?
id=7', document.mode, false, 'personalization')">Test</a>
</body>
</html>
```

Understanding the `OwpWrapPage` Script

The `OwpWrapPage` script calls the specified URL. The advantage of using this script instead of the `showPersonalizationPage`, `showMaximizePage`, `showLinkPage`, and `showHelpPage` functions is that the `OwpWrapPage` script already contains the name of the component. Therefore, you do not need to supply code within your servlet or template to extract this information. `OwpWrapPage` has the following format:

```
componentnameOwpWrapPage(url, addExtra, type)
```

- *componentname* is the system name of the component (not its description).
- *url* is the URL of the object to which you would like to submit the form. You can include `?property-value` properties in the URL.
- *addExtra* is a Boolean value specifying whether the extra fields should be added to the URL. The following is a list of the extra fields:
 - `jsessionId` (contains the Portal Builder Servlet's session ID string)

This parameter allows components to dynamically generate relevant state-based content.

- STATE (contains one of the following strings: MIN, MAX, PER, HLP, RES)

This parameter aids in component language localization. MIN stands for minimized mode, MAX stands for maximized or link mode, PER stands for personalize mode, HLP stands for help mode, and RES stands for normal mode. These codes are subject to change in future releases; consequently, you might want to create your own vehicle for returning component mode.

- LANGCODE (contains the language code string currently in use in the portal environment)

This parameter allows a component to redirect to the PortalBuilderServlet when necessary.

- PBSURL (contains the URL of the PortalBuilderServlet)
- type specifies the mode in which the URL target should be displayed. Valid types are Personalize, Maximize, Help, or Link. It also checks to make sure that the component is being run from the ActivEra Portal.

The script is already aware of the component name, so component name is not a required argument.

See Also

- *Wrapping URIs* for information about the URI functions

Understanding the OwpTargetedPage Script

The OwpTargetedPage script should be defined in the *onclick* event in the link. The script accepts any link assigned with one of the following targets and puts them in the Portal:

- `_portal`
- `_portalnostuff`
- `_portalinframe`
- `_portalnostuffinframe`

The inframe variations display the Enterprise Navigation Toolbar and the component each into their own frames, which allows a component that was not designed for the Portal to run smoothly within the Portal.

If none of these targets are indicated, the script allows the link to be processed with the original target and defaults to `_top`.

Consider the following example:

```
<a href= "http://www.myserver.com/index.html?id=1" target= "_portalnostuff"
onclick= "OwpTargetedPage (this)"> test</a>
```

This script sends the URL to the ActivEra Portal and tells the Portal not to send any extra parameters.

OwpTargetedPage has the following format:

*componentname*OwpTargetedPage(myLink)

- *componentname* is the system name of the component (not its description).
- myLink is the name of the link that the user clicks to activate the script.

The script is already aware of the component name, so component name is not a required argument.

Understanding the OneWorldComponentServlet Class

The OneWorldComponentServlet class is an abstract class designed to extend the functionality of the ComponentServlet class to provide the ability to acquire OneWorld environment and session information from the ActivEra Portal or by executing it directly. The OneWorldComponentServlet Class implements the standard service method for you. You, on the other hand, must provide some additional information not normally required by servlets. By deriving itself from the OneWorld component class, your component loads into the Portal faster than standard servlet components and use a minimum of bandwidth.

Understanding the OneWorldComponentServlet class consists of the following topic:

- Extending the OneWorldComponentServlet

Extending the OneWorldComponentServlet Class

The following servlet is the minimum servlet required to extend the OneWorldComponentServlet class correctly:

```
package com.jdedwards.oneworld.owportal.components.helloWorld;
import com.jdedwards.oneworld.owportal.components.*;
class HelloWorldServlet extends OneWorldComponentServlet
{
protected void service(HTMLWEnv owEnv, PrintWriter out,
HttpServletRequest req,
HttpServletResponse res)throws ServletException, IOException
{
out.println(getTitle());
}
public String getTitle()
{
return "Hello OneWorld";
}
public Boolean requiresOneWorld()
{
return false;
}
}
```

The service class must be implemented by every `OneWorldComponentServlet`. Because the standard service method is implemented in the parent class, this service method does not follow the Java servlet specification. For servlets that require an `OneWorld` environment, the `PortalBuilderServlet` must bypass the web server and instantiate the class directly. It will provide a `PrintStream` and an `OWEnvironment` to your class. You must not close the `PrintWriter` or instantiate a new one in your derived class; use the one that is provided to you.

The servlet uses the `getTitle` method to send the name of the current component to an output string. Every child of the `OneWorldComponentServlet` class must implement this method because `OneWorldComponentServlet` class is derived from `ComponentServlet` class. `getTitle` helps you access other portals, the `ComponentServlet`, or any object that the component servlet assumes exists.

The `requiresOneWorld` method tells the `OneWorldComponentServlet` Class or the `PortalBuilderServlet` whether a login and a `OneWorld` environment is needed. If this method returns `false`, you can expect the `OneWorld` Environment to be null. Use this method when you want to write a `OneWorld` component that does not use the database or session information but you still want the more robust interface and quicker instantiation of the `OneWorld` component mechanism.

All other aspects of this class are the same as the `ComponentServlet` class.

See Also

- Wrapping URIs* for more information about the extra parameter, `jsessionId`
- Understanding the ComponentServletClass*

Understanding the ComponentTemplateParser Class

The ComponentTemplateParser is a class that is used to parse HTML templates, which provides component servlets a simple and flexible method to display their data without having to embed the HTML inside of the Java code. You can use this class with both OneWorld components and standard Java servlets because it does not have any OneWorld dependencies.

ComponentTemplateParser accomplishes this by offering a number of simple methods used to process such documents.

Understanding the ComponentTemplateParser class consists of the following topics:

- Constructing a template parser
- Understanding the toString method
- Understanding the setString method
- Understanding the retrieve method
- Using the encompass method
- Understanding the replace method
- Understanding the replaceTag method
- Understanding the parseKey method

Constructing a Template Parser

To construct a template parser, provide ComponentTemplateParser with a string that contains the template to be processed. A copy of the provided string is stored within the ComponentTemplateParser class so that the original string will remain intact. The following is a sample instantiation of the ComponentTemplateParserClass:

```

package com.jdedwards.oneworld.owportal.components.helloWorld;
import com.jdedwards.oneworld.owportal.components.*;
class HelloWorldServlet extends ComponentServlet
{
protected void service(HttpServletRequest req,
HttpServletRequest res)throws ServletException, IOException
{

PrintWriter out = res.getWriter();

if (mypage==null)
  myPage=loadTemplate("http://www.mysite.com/mytemplate.htm", req);
ComponentTemplateParser tp = new ComponentTemplateParser (mypage);

out.println(tp.toString());
out.close();
}

public String getTitle()
{
return "Hello OneWorld";
}
static String myPage;
}

```

You can pass a static variable to the ComponentTemplateParser because the contents of this string are copied into the ComponentTemplateParser during its instantiation; therefore, the class does not actually reference the static variable.

Understanding the toString Method

Like many classes in Java, the ComponentTemplateParser overrides the default toString() method. This method returns a string containing the contents of the template with all of its replacements. In the example shown in *Constructing a Template Parser*, the toString method is used to pass the template to the output stream.

Understanding the setString Method

This method sets the template parser's string. As with the constructor, the string stored in the class is actually a copy instead of a reference to the string provided by the parameter. The format for using this method is:

```
tp.setString("String");
```

Understanding the retrieve Method

This method retrieves the contents of data between two tags. Consider the following template code:

```
<Common> document.write("me");</Common>
```

When used on this template, `myParser.retrieve("common")` returns a string containing: `document.write("me");`

The `retrieve` method also works on tags. If the tag is in the form `<tag/>` then the `retrieve` method will return an empty (not a null) string.

Using the `encompass` Method

Use this method to conditionally display a block of code. Consider the following template code:

```
<ieEncompass>
<script>
document.writeln("This is IE");
</script>
</ieEncompass>
```

If you were to use following call to the `encompass` method:

```
tp.encompass("ieencompass",true);
```

then Template Parser would return:

```
<script>
document.writeln("This is IE");
</script>
```

However, if you made this call to the `encompass` method:

```
tp.encompass("ieencompass",false);
```

then an empty string would be returned.

Understanding the `replace` Method

The `replace` method has two forms. The first form takes a tag name and a value. The second adds a Boolean value whereby you can tell the `replace` method to replace the first tag only. Simply put, this method replaces a tag with the method's value. Therefore:

```
tp.replace("myTitle", "My Component");
```

on the string:

```
<title><myTitle/></title>
```

would become:

```
<title>My Component</title>
```

Likewise, if the same statement were run on:

```
<title><myTitle>This is a test of</myTitle></title>
```

the string would still read:

```
<title>My Component</title>
```

To append a specific title to what was being encompassed in the tags, you would use code similar to the following:

```
String s = tp.retrieve("myTitle");
tp.replace("myTitle", s + " My Component");
```

Using the example above, this code would return a string that reads:

```
<title>This is a test of My Component</title>
```

By default, the replace method replaces every occurrence of a particular tag within the document. The retrieve method retrieves only the first occurrence. Running this code on a template such as this:

```
<title><myTitle>This is a test of</myTitle></title>
<body><myTitle>I like</myTitle></body>
```

would actually yield:

```
<title>This is a test of My Component</title>
```

```
<body>This is a test of My Component</body>
```

Obviously, this is not the intended result. Therefore, the second form of replace offers a Boolean value so that you can specify whether to replace only the first occurrence of an item. To process the template above correctly, the Java code should read as follows:

```
String s = tp.retrieve("myTitle");
while (s != null)
{
    tp.replace("myTitle", s + " My Component", true);
    s = tp.retrieve("myTitle");
}
```

In this code, the first occurrence of myTitle is retrieved. If the tag existed, then the first occurrence of the tag is replaced with the retrieved string and "My Component" is added to it. Then the next occurrence of the tag is retrieved and

processed again. When there are no more myTitle tags to retrieve, then the string becomes null.

Understanding the replaceTag Method

Use the replaceTag method instead of the replace method when you only want to replace tags. Consider the following example template:

```
<html>
<head>
<title>Hello <mrman/></title>
</head>
<body>
Hello <boldIfFred><mrman/></boldIfFred>!
</body>
</html>
```

The purpose of this template is to print Hello Username in regular text unless the user name is Fred, in which case, the user name is printed in bold text. To perform this task, you would use the replaceTag method in the following manner:

```
tp.replace("mrman",username);
if (username.equals("Fred"))
    tp.replaceTag("boldIfFred","strong");
else
    tp.replaceTag("boldIfFred",null);
```

If the username was Fred then the <boldIfFred> and the </boldIfFred> tags would be replaced with and respectively. Otherwise, they would be removed entirely.

Understanding the parseKey Method

Use the parseKey method to replace keys. Key replacement has several advantages over tag replacement. First, the mechanism for replacing keys is simpler; thus, key replacement methods usually execute faster than tag replacement methods. Second, keys can exist and can be replaced while residing within tags, between <Script></Script> tags, and in comments, all of which are ignored by tag methods.

One disadvantage of key replacement is the fact that keys cannot encapsulate any data. Because there is no open key/close key syntax, the functions provided by the retrieve, replace, and encapsulate tag methods cannot be duplicated.

To illustrate how to use the parseKey method, consider a scenario in which you want to define a link to a graphic for a component. The graphic is located in the /images directory of your server. Defining the anchor as:

```
<a href="http://{#jas}/images/myimage.jpg">
```

and then replacing the key “jas” with the name of your server would render a fully qualified path to the image. The `parseKey` call that would perform this replacement would read:

```
tp.parseKey(“jas”, myJasServer);
```

If you derive your servlet from the `ComponentServlet` class and use the `loadTemplate` method that allows standard field replacements, the “jas” key will be automatically processed for you when you use this call.

Understanding the `ComponentDatabaseInterface` Class

The `ComponentDatabaseInterface` class provides access to the set of database tables where your components store data. The class allows this data to be read from and written to an XML Parser (using streams). It also allows strings to be imported and exported. You can convert such strings using the `loadTemplate` method of the `ComponentServlet` class. See *Understanding the `loadTemplate()` Method* for more information.

This class requires a `OneWorld` environment; therefore, you must derive your class from the `OneWorldComponentServlet` class. Additionally, the `requiresOneWorld` method must return `TRUE`.

Understanding the `ComponentDatabaseInterface` class consists of the following topics:

- Constructing a database interface
- Using the public fields
- Understanding the `getComponentName` method
- Understanding the `setComponentName` method
- Understanding the `getData` method
- Understanding the `setData` method
- Understanding the `getDataInputStream` method
- Understanding the `getDataOutputStream` method
- Understanding the `getDefaultUserId` method
- Understanding the `setDefaultUserID` method

Constructing a Database Interface

You can instantiate the `ComponentDatabaseInterface` in the following manners:

```
new ComponentDatabaseInterface(owEnv, componentName);
```

```
new ComponentDatabaseInterface(owEnv, componentName, UserId);
```

In the first constructor, the OneWorld environment is passed in as well as the component description (the COMPNAME field sent by the ActivEra Portal Builder Servlet). This constructor assumes that the default user ID is *public. Consequently, any method that uses the default user ID instead of specifying one explicitly will retrieve only those entries that are available to the public.

Conversely, with the second constructor, you can specify a specific OneWorld user ID.

Using the Public Fields

Several static fields are available in the ComponentDatabaseInterface class. One of these fields is public and the rest are protected so that they may be overridden. All are static so they may be referenced without instantiating the class.

The PUBLIC_USER field contains the string representation of the public component user ID, that is, *public. If this field is overridden with another value, then any instantiation of the ComponentDatabaseInterface that does not provide a user ID will use this instead.

The second static field is protected and stores the name of the database table in the OneWorld environment where data for the class is stored. This field is called DB_NAME. Overloading this field will change the database table that is accessed whenever the class is instantiated. This makes it easy to extend the ComponentDatabaseInterface if you need to use different database tables.

The last static field is also protected and contains the user override type. It is named UOTY. The user override type is an entry type identifier. Most components will have a user override type of PC”(Portal Component). You can, however, override this value by extending this class. All of the functions within the ComponentDatabaseInterface would then use this user override type.

Understanding the GetComponentName Method

The GetComponentName method returns the name of the component that instantiated the method. Because this name is provided in the constructor, it can be retrieved via this method at any time. Consider the following example:

```

package com.jdedwards.oneworld.owportal.components.helloWorld;
import com.jdedwards.oneworld.owportal.components.*;
class HelloWorldServlet extends OneWorldComponentServlet
{
protected void service(HTMLWEnv owEnv, PrintWriter out, HttpServletRequest
req, HttpServletResponse res) throws ServletException, IOException

    ComponentDatabaseInterface cda = new ComponentDatabaseInterface(owEnv, "This
is my Component");
    System.out.println(cda.getComponentName());

}
public String getTitle()
{
return "Hello OneWorld",
}
public Boolean requiresOneWorld()
{
return true;
}
}

```

This code returns the following text string:

This is my Component

Understanding the setComponentName Method

The setComponentName method allows you to change the component name after the class is instantiated. Consider the following example:

```

protected void service(HTMLWEnv owEnv, PrintWriter out, HttpServletRequest
req, HttpServletResponse res) throws ServletException, IOException
{

    ComponentDatabaseInterface cda = new ComponentDatabaseInterface(owEnv, "This
is my Component");
    cda.setComponentName("myComponent");
    System.out.println(cda.getComponentName());

}

```

This code returns the following text string:

myComponent

Understanding the getData Method

This method has two versions:

getData();

getData(String);

Both versions of this method return a string of data for a component in the database. Usually the data returned would be an XML document, but the returned value is not restricted in its format. Note, however, that the method allows only one data field per user, per component.

The first method uses the default user ID that is passed to the class during its instantiation, while the second method requires that you pass it the user ID. If no user ID is specified, then the first method uses the value of the PUBLIC_USER field.

Consider the following example:

```
protected void service(HTMLWEnv owEnv, PrintWriter out, HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException
{
    ComponentDatabaseInterface cda = new ComponentDatabaseInterface(owEnv,
"myComponent", "ME000001");
    System.out.println(cda.getData());
    System.out.println(cda.getData("ME000000"));
}
```

The above code retrieves the data field for the user ME000001, and then retrieves data for the user ME000000.

Understanding the setData Method

The setData method has two versions:

```
setData(data);
```

```
setData(data, userId);
```

The first version sets the data to a string using the default user ID. The second one saves data using a specified user ID. After this data is set, it is also committed to the database.

Understanding the getDataInputStream Method

This method is the same as the getData method. It has two versions, just like the getData method does, and the differences between the two versions are the same. What makes the getDataInputStream method different from the getData method, however, is that getDataInputStream returns an input stream instead of a string. This input stream implementation aids in the integration of some XML DOM parsers. Many DOM parsers expect a file or some other input stream to get the Data into the DOM.

See *Understanding the getData Method* for more information about the getData method.

Understanding the `getDataOutputStream` Method

This method has two versions:

```
getDataOutputStream();
```

```
getDataOutPutStream(UserId);
```

This method provides an output stream which is required by many DOMs to save their data. This stream caches all of the data sent to it, and when the stream is closed, it saves the data to the database.

The first method uses the default user ID that was passed to the class during its instantiation, while the second method requires that you pass it the user ID. If no user ID is specified, then the first method uses the value of the `PUBLIC_USER` field.

Understanding the `getDefaultUserId` Method

The `getDefaultUserId` method returns a string containing the default user ID that was used when the class was instantiated, set with the last `setDefaultUserId` method, or the value of the `PUBLIC_USER` field if no user ID was specified. This ID is used in every method that uses the default user ID.

Understanding the `setDefaultUserId` Method

The `setDefaultUserId` method changes the value of the default user ID.

Understanding the Link Center Component

The Link Center is a OneWorld component based on Java servlet technology. It uses many of the features provided by the ActivEra Portal; consequently, you can use it as a working example for learning to build Portal components. The source code is well documented and should be clear to a user with intermediate or better knowledge of Java.

The Link Center employs the following classes:

- | | |
|-----------------------------|---|
| LinkCenterServlet | This class is the main servlet for the Link Center. It is a child of OneWorldComponentServlet and takes advantage of the ComponentTemplateParser and the ComponentDatabaseInterface. This servlet shows how an implementation of an XML DOM can be used to integrate with the database using streams, and also takes advantage of using variables in the JAS to cache commonly used data. |
| LinkCenterData | This class implements Sun's XML DOM Extension, providing methods that makes the DOM easier to access and able to interface with the LinkCenterElements. This implementation is written specifically for the Link Center, but may be used as an example for wrapping other DOMs. |
| LinkCenterElement | This is an abstract class that implements some standard functionality common among the LinkCenterLinks and LinkCenterCategories below. It was designed to be extended by these two more specific classes, but also provides a common interface among the two. |
| LinkCenterLinks | This class extends LinkCenterElement to provide access to data that is specific to links. |
| LinkCenterCategories | This class extends LinkCenterElement to provide access to data that is specific to categories. |

Finally, a number of template files that are parsed by the Link Center also exist. These templates are located in the owportal/components/linkcenter directory.

Understanding Portal Design Standards

Components in normal mode should adapt to changes in width to the greatest extent possible. A reasonable size for your component is approximately 250–300 pixels wide by 250–300 pixels high.

All components should be able to function independently of the ActivEra Portal. Your component is responsible for storing any information that it needs for personalization or to support its functionality.

Always fully qualify all graphics.

When designing HTML components, avoid tags such as META, HTML, and BODY. Do not use the FRAMESET tag.

Java URI and OneWorld components should be packaged as:

```
com.yourcompany.oneworld.activera.owportal.components:compname
```

where *yourcompany* is the name of your company and *compname* is the name of the component.

All JavaScripts within your component should be named:

```
com_yourcompany_yourcomponent_scriptname
```

where *yourcompany* is the name of your company and *yourcomponent* is the name of your component.

All other files (non-JavaScript files) should reside in directories named:

```
yourcompany/owportal/components/yourcomponent
```

where *yourcompany* is the name of your company and *yourcomponent* is the name of your component.

Understanding Portal Style Sheets

An external style sheet is used to specify fonts in the ActivEra Portal. The following is a list of the style sheets, their properties, and their use:

body	Properties: margin-left:0; margin-top:0; margin-width:0; margin-height:0; font family: verdana, arial, helvetica, sans-serif; font size: 9pt Usage: Portal Style: Body; Tag elements: Margins, body font
.bodycopy	Properties: Font family: verdana, sans-serif; font size: 9pt Usage: Portal STyle: body copy tag
.portaltitle	Properties: Font family: verdana, arial, helvetica, sans-serif; font size:10pt; fontweight: bold; color: #FFFFFF Usage: Portal Style: Portal Title
.welcometitle	Properties: Font family: verdana, arial, helvetica, sans-serif;; font size:8.5pt; font style: italic; font weight: normal; color: #FFCC66 Usage: Portal Style: Portal Welcome Title
.mainappnav	Properties: Font family: verdana, arial, helvetica, sans-serif; font size:8pt; color: #FFFFFF; font weight: bold Usage: Portal Style: Main Application Navigation Title
.comptitles	Properties: Font family: verdana, arial, helvetica, sans-serif; font size:9pt; color: #330066; font weight: bold Usage: Portal Style: Component Titles
.headline	Properties: Font family: verdana, arial, helvetica, sans-serif; font size:8.5pt Usage: News Headlines within components
.smalllinks	Properties: Font family: verdana; font size:7pt Usage: Portal Style: Small Links (for example, Link Center)
.linkheads	Properties: Font family: verdana, arial, helvetica, sans-serif; font size:9pt; font weight: bold; font color: #CC3333 Usage: Portal Style: Heads (for example, Link Center Headings)

A:link	Properties: color: #003399 Usage: Link Color
A:visited	Properties: color: #0066CC Usage: Visited Link Color
A:hover	Properties: color: #003399 Usage: Hovered Link Color
.mainhead	Properties: color: #990033; font family: verdana, arial, helvetica, sans-serif; font size:10pt; font style bold; font weight: bold Usage: Content style: Main Content Head
.secondhead	Properties: color: black; font family: verdana, arial, helvetica, sans-serif; font size:10pt; font style: bold; font weight: bold Usage: Content style: Secondary Content Head
.contacts	Properties: Font family: verdana, arial, helvetica, sans-serif; font size:9pt; font weight: bold Usage: Content style: Headings for contacts

