



EnterpriseOne Tools 8.94

PeopleBook: Development Tools: Form Design Aid

November 2004

EnterpriseOne Tools 8.94 PeopleBook: Development Tools: Form Design Aid
SKU E1_TOOLS8.94TFD-B 1104

Copyright © 2004 PeopleSoft, Inc. All rights reserved.

All material contained in this documentation is proprietary and confidential to PeopleSoft, Inc. ("PeopleSoft"), protected by copyright laws and subject to the nondisclosure provisions of the applicable PeopleSoft agreement. No part of this documentation may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, including, but not limited to, electronic, graphic, mechanical, photocopying, recording, or otherwise without the prior written permission of PeopleSoft.

This documentation is subject to change without notice, and PeopleSoft does not warrant that the material contained in this documentation is free of errors. Any errors found in this document should be reported to PeopleSoft in writing.

The copyrighted software that accompanies this document is licensed for use only in strict accordance with the applicable license agreement which should be read carefully as it governs the terms of use of the software and this document, including the disclosure thereof.

PeopleSoft, PeopleTools, PS/nVision, PeopleCode, PeopleBooks, PeopleTalk, and Vantive are registered trademarks, and Pure Internet Architecture, Intelligent Context Manager, and The Real-Time Enterprise are trademarks of PeopleSoft, Inc. All other company and product names may be trademarks of their respective owners. The information contained herein is subject to change without notice.

Open Source Disclosure

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

Apache Software Foundation

This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>). Copyright (c) 1999-2000 The Apache Software Foundation. All rights reserved.

THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

OpenSSL

Copyright (c) 1998-2003 The OpenSSL Project. All rights reserved.

THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

SSLeay

Copyright (c) 1995-1998 Eric Young. All rights reserved.

THIS SOFTWARE IS PROVIDED BY ERIC YOUNG "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Loki Library

Copyright (c) 2001 by Andrei Alexandrescu. This code accompanies the book:

Alexandrescu, Andrei. "Modern C++ Design: Generic Programming and Design Patterns Applied". Copyright (c) 2001. Addison-Wesley. Permission to use, copy, modify, distribute and sell this software for any purpose is hereby granted without fee, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation.

Contents

General Preface

About This PeopleBook	xix
PeopleSoft Application Prerequisites.....	xix
PeopleSoft Application Fundamentals.....	xix
Documentation Updates and Printed Documentation.....	xx
Obtaining Documentation Updates.....	xx
Ordering Printed Documentation.....	xx
Additional Resources.....	xxi
Typographical Conventions and Visual Cues.....	xxii
Typographical Conventions.....	xxii
Visual Cues.....	xxiii
Country, Region, and Industry Identifiers.....	xxiii
Currency Codes.....	xxiv
Comments and Suggestions.....	xxiv
Common Elements Used in PeopleBooks.....	xxiv

Preface

Form Design Aid Preface.....	xxvii
PeopleSoft Products.....	xxvii
PeopleSoft Form Design Aid.....	xxvii
Additional Resources.....	xxvii

Chapter 1

Getting Started with PeopleSoft Tools Form Design Aid.....	1
PeopleSoft Tools Form Design Aid Overview.....	1
PeopleSoft Tools Form Design Aid Implementation.....	1

Chapter 2

Working with Forms.....	3
Understanding Forms.....	3
Understanding Form Interconnections.....	4
Configuring Forms at Design Time.....	4
Understanding Forms.....	10

Understanding Form Creation.....	10
Creating a Form.....	12
Creating a Modal Form Interconnection.....	12
Creating a Modeless Form Interconnection.....	13
Working with Quick Form.....	14
Prerequisite.....	14
Using Quick Form.....	14

Chapter 3

Working with Form Controls.....	15
Understanding Form Controls.....	15
Understanding Form Control Design-Time Considerations.....	18
Attaching Data Items to a Control.....	40
Understanding the Relationship between Data Dictionary Items and Controls.....	40
Attaching a Data Item to a Control.....	40
Overriding a Default Data Dictionary Trigger.....	41
Associating a Data Item Description with a Field.....	41
Understanding the Relationship between Data Item Descriptions and Fields.....	41
Displaying the Title of a Data Item Associated with a Field.....	41
Grouping Controls.....	42
Setting the Tab Sequence of Controls on a Form.....	42
Understanding Tab Sequence.....	42
Changing the Tab Sequence on a Form.....	42

Chapter 4

Working with FDA Compare.....	43
Understanding FDA Compare.....	43
Understanding the FDA Compare Color Scheme.....	43
Using FDA Compare.....	44
Forms Used To Use FDA Compare.....	44
Merging Specifications Using FDA Compare.....	44

Chapter 5

Understanding Browse Portlet Forms.....	47
Understanding the Browse Portlet Form.....	47
Understanding Browse Portlet Form Events.....	48
Generating Portlets for Collaborative Portal.....	48

Prerequisites.....	49
Understanding Portlet Generation.....	49
Deploying an FDA-Created Portlet.....	49
Updating an FDA-Created Portlet after Initial Installation.....	49
Understanding Portlet Updates.....	50
Adding a New Portlet Application.....	50
Deleting an Existing Portlet Application.....	50

Chapter 6

Understanding Edit Portlet Forms.....	51
Edit Portlet Forms.....	51
Edit Portlet Form Design-Time Considerations.....	52
Edit Portlet Form Events.....	52

Chapter 7

Understanding Find/Browse Forms.....	55
Find/Browse Forms.....	55
Find/Browse Events.....	55
Find/Browse Runtime Processing.....	55

Chapter 8

Understanding Fix/Inspect Forms.....	59
Fix/Inspect Forms.....	59
Fix/Inspect Events.....	59
Fix/Inspect Runtime Processing.....	59

Chapter 9

Understanding Header Detail Forms.....	65
Header Detail Forms.....	65
Header Detail Design-Time Considerations.....	65
Header Detail Events.....	65
Header Detail Runtime Processing.....	66

Chapter 10

Understanding Headerless Detail Forms.....	71
Headerless Detail Forms.....	71
Headerless Detail Design-Time Considerations.....	71
Headerless Detail Events.....	71
Headerless Detail Runtime Processing.....	72

Chapter 11

Understanding Message Forms.....	77
Message Forms.....	77
Message Form at Design-Time Considerations.....	77
Understanding Message Form Events.....	77
Message Form Runtime Processing.....	77

Chapter 12

Understanding Parent/Child Browse Forms.....	79
Parent/Child Browse Forms.....	79
Parent/Child Browse Events.....	79
Parent/Child Browse Runtime Processing.....	79

Chapter 13

Using Power Browse Forms.....	83
Power Browse Forms.....	83
Power Form Hierarchical Structures.....	83
Power Browse Form Design-Time Considerations.....	86
Power Browse Events.....	86
Power Browse Form Runtime Processing.....	87
Transaction Boundaries for Power Forms and Subforms.....	88

Chapter 14

Understanding Power Edit Forms.....	91
Power Edit Forms.....	91
Power Edit Form Design-Time Considerations.....	91
Power Edit Events.....	92
Power Edit Form Runtime Processing.....	93

Chapter 15

Understanding Search & Select Forms.....	99
Search & Select Forms.....	99
Search & Select Events.....	99
Search & Select Runtime Processing.....	99

Chapter 16

Understanding Wizard Forms.....	103
Wizard Forms.....	103

Chapter 17

Understanding Calendar Controls.....	105
Calendar Controls.....	105
Calendar Control Design-Time Considerations.....	106
Calendar Control Events.....	106
Calendar Control Runtime Processing.....	107
Calendar Control System Functions.....	108
Add Calendar Activity.....	109
Delete Calendar Activity.....	111
Modify Calendar Activity.....	112
Select Calendar View.....	114

Chapter 18

Understanding Check Box Controls.....	115
Check Box Controls.....	115
Check Box Control Design-Time Considerations.....	115
Check Box Events.....	115

Chapter 19

Understanding Combo Box Controls.....	117
Understanding the Combo Box Control.....	117
Loading the Combo Box Control.....	117
Loading a Combo Box from a UDC.....	118
Loading a Combo Box from Cache.....	118
Loading a Combo Box with the Add Item System Function.....	119
Understanding Combo Box Control Design-Time Considerations.....	119

Understanding Combo Box Control Events.....	119
Understanding Combo Box Control Runtime Processing.....	120
Combo Box Control System Functions.....	124
Add Item.....	124
Get Description.....	125
Get Index of Key.....	126
Get Item at Index.....	126
Get Item Count.....	127
Get Key at Index.....	127
Load from Cache.....	127
Remove Item by Index.....	128
Remove Item by Key.....	129
Select Item.....	129
Embedded Combo Box System Functions.....	130
Add Item.....	130
Get Description.....	131
Get Index of Key.....	131
Get Item at Index.....	132
Get Item Count.....	133
Get Key at Index.....	133
Load from Cache.....	134
Remove Item by Index.....	135
Remove Item by Key.....	135
Select Item.....	136

Chapter 20

Understanding Edit Controls.....	137
Edit Controls.....	137
Edit Control Events.....	138
Edit Control Runtime Processing.....	138
Edit Control System Functions.....	138
Set Edit Control Color.....	139
Set Edit Control Font.....	139

Chapter 21

Understanding Grid Controls.....	141
Grid Controls.....	141
Grid Control Design-Time Considerations.....	142

Designing the Grid.....	142
Adding Columns to the Grid Control.....	143
Setting Property Values for the Grid Control.....	143
Showing Multiple Currencies per Column.....	147
Grid Control Events.....	147
Grid Control Runtime Processing.....	149
How Runtime Processes the Grid Control.....	149
Impact of Interactivity Levels.....	159
Grid Control System Functions.....	160
Change Row Selection.....	162
Clear Grid Buffer.....	163
Clear Grid Cell Error.....	163
Clear QBE Column.....	163
Clear Selection.....	164
Clear Sequencing.....	164
Copy Grid Row to Grid Buffer.....	164
Delete Grid Row.....	165
Disable Grid.....	165
Display Customized Grid Option.....	166
Display Export to Excel Option.....	166
Display Export to Word Option.....	166
Display Import from Excel Option.....	167
Enable Grid.....	167
Get Grid Row.....	167
Get Max Grid Rows.....	168
Get Next Selected Row.....	168
Get Selected Grid Row Count.....	169
Get Selected Grid Row Number.....	169
Hide Grid Column.....	170
Hide Grid Row.....	170
Insert Grid Buffer Row.....	171
Insert Grid Row.....	171
Set Data Dictionary Item.....	172
Set Data Dictionary Item Overrides.....	172
Set Grid Cell Error.....	172
Set Grid Color.....	173
Set Grid Column Heading.....	173
Set Grid Font.....	174
Set Grid Row Bitmap.....	174
Set Grid Row Format.....	175

Set Lower Limit.....	175
Set QBE Column Compare Style.....	177
Set Selection.....	177
Set Selection Append Flag.....	178
Set Sequencing.....	179
Show Grid Column.....	179
Show Grid Row.....	180
Suppress Grid Line.....	180
Update Grid Buffer Row.....	180
Was Grid Cell Value Entered.....	181

Chapter 22

Understanding Image Controls.....	183
Image Controls.....	183
Image Control Design-Time Considerations.....	183

Chapter 23

Understanding Media Object Controls.....	185
Media Object Controls.....	185
Media Object Control Design-Time Considerations.....	185
Media Object System Functions.....	186
Access Media Object.....	186
Activate Item.....	186
Clear Characterization Cache.....	187
Delete Item.....	187
Disable Characterization Cache.....	187
Get OLE Item.....	188
Insert OLE Object.....	188
Insert Text.....	189
Insert URL.....	190
Hide the Viewer Icon Panel.....	190
Lock the Viewer Splitter Bar.....	191
Set Characterization Cache.....	191
Set Cursor Position.....	191
Set Grid Text Indicator.....	192
Set Text Color.....	192

Chapter 24

Understanding Parent Child Controls.....	193
Parent Child Controls.....	193
Tree Nodes.....	194
Lean Manufacturing.....	194
Parent Child Control Design-Time Considerations.....	194
Parent Child Control Properties.....	194
Parent Child Control and Power Forms.....	195
Lean Manufacturing Properties.....	195
Parent Child Control Events.....	196
Selecting Tree Nodes.....	197
Performing Drag-and-Drop or Copy/Cut/Paste.....	197
Expanding and Collapsing Nodes.....	199
Clicking Bitmaps.....	201
Parent Child Control System Functions.....	201
Add Action.....	202
Attach Path To Segment.....	203
Change Row Selection.....	203
Clear Grid Buffer.....	204
Clear Grid Cell Error.....	204
Clear QBE Column.....	204
Contact Tree Node.....	205
Copy Grid Row To Grid Buffer.....	205
Delete All Actions.....	205
Delete All Tree Nodes.....	206
Delete Grid Row.....	206
Disable Grid.....	206
Enable Grid.....	207
Expand Tree Node.....	207
Get Grid Row.....	208
Get Max Grid Rows.....	208
Get Next Selected Row.....	208
Get Node ID.....	209
Get Node Level.....	210
Get Related Node ID.....	210
Get Row Number.....	211
Get Selected Context Action.....	211
Get Selected Grid Row Count.....	212
Get Selected Grid Row Number.....	212
Get Tree Node Handle.....	213

Hide Grid Column.....	213
Insert Grid Buffer Row.....	213
Insert Grid Buffer Row By Node ID.....	214
Set Action.....	215
Set Data Dictionary Item.....	216
Set Data Dictionary Overrides.....	216
Set Drag Cursor.....	217
Set Grid Cell Error.....	217
Set Grid Color.....	217
Set Grid Column Heading.....	218
Set Grid Font.....	218
Set Grid Row Bitmap.....	219
Set QBE Column Compare Style.....	219
Set Tree Bitmap Scheme.....	220
Set Tree Node Bitmap.....	220
Set Tree Node Bold.....	221
Set Tree Node Clickable Bitmap.....	221
Set Tree Node Handle.....	222
Set Tree Root Node ID.....	222
Show Grid Column.....	223
Show N Levels.....	223
Suppress Fetch On Node Expand.....	223
Suppress Grid Line.....	224
Suppress Node Indent/Outdent.....	224
Suppress Node Move Up/Down.....	224
Update Grid Buffer Row.....	225
Was Grid Cell Value Entered.....	225

Chapter 25

Understanding Push Button Controls.....	227
Push Button Controls.....	227
Push Button Events.....	227

Chapter 26

Understanding Radio Button Controls.....	229
Radio Button Controls.....	229
Radio Button Design-Time Considerations.....	229
Radio Button Events.....	229

Chapter 27

Understanding Saved Query Controls.....	231
Saved Query Controls.....	231
Saved Query Control Design-Time Considerations.....	232
Runtime Processing for the Saved Query Control.....	232

Chapter 28

Understanding Static Text Controls.....	233
Static Text Controls.....	233

Chapter 29

Understanding Subforms and Subform Aliases.....	235
Understanding Subforms.....	235
Understanding Subform Design-Time Considerations.....	236
Understanding Subform Events.....	237
Understanding Subform Runtime Processing.....	238
Control Initialization.....	238
Subform Push Buttons.....	238
Creating Subforms.....	240
Understanding Subform Creation.....	240
Creating a Subform without a Power Form.....	240
Creating a Subform on a Power Form.....	240
Creating a Subform as a Tab Page.....	241
Reusing Subforms.....	241
Understanding Subform Reuse.....	241
Reusing a Subform on a Power Form.....	242
Working with Data Structures and Subforms.....	242
Mapping a Parent's Variables to a Child Subform.....	242
Working with Functions and Subforms.....	242
Adding a Function to a Subform.....	243
Subform System Functions.....	243
Call Function.....	243
Enable Subform.....	243
Disable Subform.....	244
Hide Subform.....	244
Show Subform.....	244
Update Parent.....	244
Notify Parent.....	244

Get Error Count.....	245
Get Warning Count.....	245
Get Subform ID.....	245
Notify Child.....	246
Trigger Default Action.....	246
Expand Subform.....	246
Collapse Subform.....	247

Chapter 30

Understanding Tab and Tab Page Controls.....	249
Understanding Tab and Tab Page Controls.....	249
Creating Tab Controls.....	249
Creating a Tab Control.....	249
Tab Control System Functions.....	250
Disable Tab Page.....	250
Enable Tab Page.....	250
Hide Tab Page.....	250
Set Current Tab Page.....	251
Set Tab Page Text.....	251

Chapter 31

Understanding Text Block Controls.....	253
Text Block Controls.....	253
Text Block Control Design-Time Considerations.....	253
Text Block Events.....	253
Text Block System Functions.....	253
Add Segment.....	254
Get Last Clicked Segment.....	254
Get Segment Information.....	255
Remove Segment.....	255
Update Segment.....	255

Chapter 32

Understanding Text Search Controls.....	257
Text Search Controls.....	257

Chapter 33

Understanding Tree Controls.....	259
Tree Controls.....	259
Tree Control Events.....	260
Tree Control System Functions.....	261
Bulk Tree Load.....	261
Contract Tree Node.....	261
Delete Tree Node.....	262
Expand Tree Node.....	262
Get Node Information.....	262
Get Node Level.....	263
Get Tree Node Handle.....	264
Insert Tree Node.....	264
Set Bitmap Scheme.....	265
Set Node Bitmap.....	266
Set Node Information.....	266
Set Node Text.....	267
Set Tree Node Handle.....	267

Chapter 34

Understanding Wizard Controls.....	269
Wizard Controls.....	269
Wizard Control Design-Time Considerations.....	270
Implementing Re-entry Save.....	271
Wizard Control Events.....	272
Wizard Control Runtime Processing.....	273
Wizard Control Transaction Processing.....	284
Wizard Control System Functions.....	285
Get Current Wizard Page ID.....	285
Get Wizard Page Index.....	285
Set Selected Wizard Page.....	286
Set Wizard Form Mode.....	286
Set Wizard Page Index.....	287
Set Wizard Page Status.....	287
Suppress Wizard Page Validation and Save.....	288

Appendix A

System Functions in Form Design Aid.....	289
System Functions.....	289
Control.....	289
Clear Control Error.....	289
Disable Control.....	290
Enable Control.....	290
Go to Url.....	290
Hide Control.....	291
Set Control Error.....	291
Set Control Text.....	291
Set Data Dictionary Item.....	292
Set Data Dictionary Overrides.....	292
Set Statusbar Text.....	292
Show Control.....	293
Was Value Entered.....	293
General.....	294
Cancel User Transaction.....	294
Continue Custom Data Fetch.....	294
Copy Currency Information.....	294
Launch Batch Application.....	294
Launch Processing Options Dialog.....	296
Press Button.....	296
Run Executable.....	297
Set Control Focus.....	297
Set Form Title.....	298
Set Time Zone On Form.....	298
Stop Processing.....	298
Suppress Add.....	298
Suppress Default Visual Assist Form.....	298
Suppress Delete.....	299
Suppress Find.....	299
Suppress Update.....	299
Time Between.....	299
Was Form Record Fetched.....	300
Messaging.....	300
Send Message Extended.....	300
Mail Merge & Doc Gen (Web Only).....	303
Delete Document.....	304
Display Document.....	304

Download Template.....305

Download Template for Doc Gen.....305

Get XML Data Model.....306

Run Doc Gen and Display.....307

Run Mail Merge and Display.....308

Run Multiple Mail Merge.....308

Upload Template.....309

Upload Template for Doc Gen.....310

Glossary of PeopleSoft Terms.....311

Index331

About This PeopleBook

PeopleBooks provide you with the information that you need to implement and use PeopleSoft applications.

This preface discusses:

- PeopleSoft application prerequisites.
- PeopleSoft application fundamentals.
- Documentation updates and printed documentation.
- Additional resources.
- Typographical conventions and visual cues.
- Comments and suggestions.
- Common elements in PeopleBooks.

Note. PeopleBooks document only page elements, such as fields and check boxes, that require additional explanation. If a page element is not documented with the process or task in which it is used, then either it requires no additional explanation or it is documented with common elements for the section, chapter, PeopleBook, or product line. Elements that are common to all PeopleSoft applications are defined in this preface.

PeopleSoft Application Prerequisites

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

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

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

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

PeopleSoft Application Fundamentals

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

The application fundamentals PeopleBook consists of important topics that apply to many or all PeopleSoft applications across one or more product lines. Whether you are implementing a single application, some combination of applications within the product line, or the entire product line, you should be familiar with the contents of the appropriate application fundamentals PeopleBooks. They provide the starting points for fundamental implementation tasks.

Documentation Updates and Printed Documentation

This section discusses how to:

- Obtain documentation updates.
- Order printed documentation.

Obtaining Documentation Updates

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

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

See Also

PeopleSoft Customer Connection, <https://www.peoplesoft.com/corp/en/login.jsp>

Ordering Printed Documentation

You can order printed, bound volumes of the complete PeopleSoft documentation that is delivered on your PeopleBooks CD-ROM. PeopleSoft makes printed documentation available for each major release shortly after the software is shipped. Customers and partners can order printed PeopleSoft documentation by using any of these methods:

- Web
- Telephone
- Email

Web

From the Documentation section of the PeopleSoft Customer Connection website, access the PeopleBooks Press website under the Ordering PeopleBooks topic. The PeopleBooks Press website is a joint venture between PeopleSoft and MMA Partners, the book print vendor. Use a credit card, money order, cashier's check, or purchase order to place your order.

Telephone

Contact MMA Partners at 877 588 2525.

Email

Send email to MMA Partners at peoplesoftpress@mmapartner.com.

See Also

PeopleSoft Customer Connection, <https://www.peoplesoft.com/corp/en/login.jsp>

Additional Resources

The following resources are located on the PeopleSoft Customer Connection website:

Resource	Navigation
Application maintenance information	Updates + Fixes
Business process diagrams	Support, Documentation, Business Process Maps
Interactive Services Repository	Interactive Services Repository
Hardware and software requirements	Implement, Optimize + Upgrade, Implementation Guide, Implementation Documentation & Software, Hardware and Software Requirements
Installation guides	Implement, Optimize + Upgrade, Implementation Guide, Implementation Documentation & Software, Installation Guides and Notes
Integration information	Implement, Optimize + Upgrade, Implementation Guide, Implementation Documentation and Software, Pre-built Integrations for PeopleSoft Enterprise and PeopleSoft EnterpriseOne Applications
Minimum technical requirements (MTRs) (EnterpriseOne only)	Implement, Optimize + Upgrade, Implementation Guide, Supported Platforms
PeopleBook documentation updates	Support, Documentation, Documentation Updates
PeopleSoft support policy	Support, Support Policy
Prerelease notes	Support, Documentation, Documentation Updates, Category, Prerelease Notes
Product release roadmap	Support, Roadmaps + Schedules
Release notes	Support, Documentation, Documentation Updates, Category, Release Notes
Release value proposition	Support, Documentation, Documentation Updates, Category, Release Value Proposition
Statement of direction	Support, Documentation, Documentation Updates, Category, Statement of Direction

Resource	Navigation
Troubleshooting information	Support, Troubleshooting
Upgrade documentation	Support, Documentation, Upgrade Documentation and Scripts

Typographical Conventions and Visual Cues

This section discusses:

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

Typographical Conventions

This table contains the typographical conventions that are used in PeopleBooks:

Typographical Convention or Visual Cue	Description
Bold	Indicates PeopleCode function names, business function names, event names, system function names, method names, language constructs, and PeopleCode reserved words that must be included literally in the function call.
<i>Italics</i>	Indicates field values, emphasis, and PeopleSoft or other book-length publication titles. In PeopleCode syntax, italic items are placeholders for arguments that your program must supply. We also use italics when we refer to words as words or letters as letters, as in the following: Enter the letter <i>O</i> .
KEY+KEY	Indicates a key combination action. For example, a plus sign (+) between keys means that you must hold down the first key while you press the second key. For ALT+W, hold down the ALT key while you press the W key.
Monospace font	Indicates a PeopleCode program or other code example.
“ ” (quotation marks)	Indicate chapter titles in cross-references and words that are used differently from their intended meanings.

Typographical Convention or Visual Cue	Description
. . . (ellipses)	Indicate that the preceding item or series can be repeated any number of times in PeopleCode syntax.
{ } (curly braces)	Indicate a choice between two options in PeopleCode syntax. Options are separated by a pipe ().
[] (square brackets)	Indicate optional items in PeopleCode syntax.
& (ampersand)	When placed before a parameter in PeopleCode syntax, an ampersand indicates that the parameter is an already instantiated object. Ampersands also precede all PeopleCode variables.

Visual Cues

PeopleBooks contain the following visual cues.

Notes

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

Note. Example of a note.

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

Important! Example of an important note.

Warnings

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

Warning! Example of a warning.

Cross-References

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

Country, Region, and Industry Identifiers

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

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

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

Country Identifiers

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

Region Identifiers

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

- Asia Pacific
- Europe
- Latin America
- North America

Industry Identifiers

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

- USF (U.S. Federal)
- E&G (Education and Government)

Currency Codes

Monetary amounts are identified by the ISO currency code.

Comments and Suggestions

Your comments are important to us. We encourage you to tell us what you like, or what you would like to see changed about PeopleBooks and other PeopleSoft reference and training materials. Please send your suggestions to:

PeopleSoft Product Documentation Manager PeopleSoft, Inc. 4460 Hacienda Drive Pleasanton, CA 94588

Or send email comments to doc@peoplesoft.com.

While we cannot guarantee to answer every email message, we will pay careful attention to your comments and suggestions.

Common Elements Used in PeopleBooks

Address Book Number

Enter a unique number that identifies the master record for the entity. An address book number can be the identifier for a customer, supplier, company, employee, applicant, participant, tenant, location, and so on. Depending on the application, the field on the form might refer to the address book number as the customer number, supplier number, or company number, employee or applicant id, participant number, and so on.

As If Currency Code	Enter the three-character code to specify the currency that you want to use to view transaction amounts. This code allows you to view the transaction amounts as if they were entered in the specified currency rather than the foreign or domestic currency that was used when the transaction was originally entered.
Batch Number	Displays a number that identifies a group of transactions to be processed by the system. On entry forms, you can assign the batch number or the system can assign it through the Next Numbers program (P0002).
Batch Date	Enter the date in which a batch is created. If you leave this field blank, the system supplies the system date as the batch date.
Batch Status	<p>Displays a code from user-defined code (UDC) table 98/IC that indicates the posting status of a batch. Values are:</p> <p><i>Blank:</i> Batch is unposted and pending approval.</p> <p><i>A:</i> The batch is approved for posting, has no errors and is in balance, but it has not yet been posted.</p> <p><i>D:</i> The batch posted successfully.</p> <p><i>E:</i> The batch is in error. You must correct the batch before it can post.</p> <p><i>P:</i> The system is in the process of posting the batch. The batch is unavailable until the posting process is complete. If errors occur during the post, the batch status changes to E.</p> <p><i>U:</i> The batch is temporarily unavailable because someone is working with it, or the batch appears to be in use because a power failure occurred while the batch was open.</p>
Branch/Plant	Enter a code that identifies a separate entity as a warehouse location, job, project, work center, branch, or plant in which distribution and manufacturing activities occur. In some systems, this is called a business unit.
Business Unit	Enter the alphanumeric code that identifies a separate entity within a business for which you want to track costs. In some systems, this is called a branch/plant.
Category Code	Enter the code that represents a specific category code. Category codes are user-defined codes that you customize to handle the tracking and reporting requirements of your organization.
Company	Enter a code that identifies a specific organization, fund, or other reporting entity. The company code must already exist in the F0010 table and must identify a reporting entity that has a complete balance sheet.
Currency Code	Enter the three-character code that represents the currency of the transaction. PeopleSoft EnterpriseOne provides currency codes that are recognized by the International Organization for Standardization (ISO). The system stores currency codes in the F0013 table.
Document Company	<p>Enter the company number associated with the document. This number, used in conjunction with the document number, document type, and general ledger date, uniquely identifies an original document.</p> <p>If you assign next numbers by company and fiscal year, the system uses the document company to retrieve the correct next number for that company.</p>

If two or more original documents have the same document number and document type, you can use the document company to display the document that you want.

Document Number

Displays a number that identifies the original document, which can be a voucher, invoice, journal entry, or time sheet, and so on. On entry forms, you can assign the original document number or the system can assign it through the Next Numbers program.

Document Type

Enter the two-character UDC, from UDC table 00/DT, that identifies the origin and purpose of the transaction, such as a voucher, invoice, journal entry, or time sheet. PeopleSoft EnterpriseOne reserves these prefixes for the document types indicated:

P: Accounts payable documents.

R: Accounts receivable documents.

T: Time and pay documents.

I: Inventory documents.

O: Purchase order documents.

S: Sales order documents.

Effective Date

Enter the date on which an address, item, transaction, or record becomes active. The meaning of this field differs, depending on the program. For example, the effective date can represent any of these dates:

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

Fiscal Period and Fiscal Year

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

G/L Date (general ledger date)

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

Form Design Aid Preface

This preface discusses the Form Design Aid PeopleBook.

PeopleSoft Products

This PeopleBook refers to this PeopleSoft product line: PeopleSoft EnterpriseOne Tools.

PeopleSoft Form Design Aid

This PeopleBook covers Form Design Aid (FDA), a member of the PeopleSoft EnterpriseOne Tools suite. FDA is used to create, modify, and rebrand EnterpriseOne applications. Its chapters describe the tool in general and then defines each form type and each control type in terms of how to set each up in FDA and how runtime processes events and application logic.

Additional Resources

The following resources are located on the PeopleSoft Customer Connection website:

Resource	Navigation
<i>EnterpriseOne PeopleTools 8.94 Installation and Configuration for Collaborative Portal</i>	Customer Connection
<i>EnterpriseOne 8.93 APIs</i>	Customer Connection

CHAPTER 1

Getting Started with PeopleSoft Tools Form Design Aid

This chapter provides an overview of preparing to use Form Design Aid (FDA).

PeopleSoft Tools Form Design Aid Overview

Use FDA to create or modify EnterpriseOne applications. Applications are composed of forms, and a form is the interface between a user and a table. This interface should present the data logically and contain the functions that are necessary to enter and manipulate data.

Other Sources of Information

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

See Also

About These PeopleBooks Preface

PeopleSoft Tools Form Design Aid Implementation

To use FDA to work with EnterpriseOne applications, these tasks must be completed first:

- You must have a valid EnterpriseOne user account.

Depending on how security has been configured, you might need one or more roles assigned to you so that you can access Object Management Workbench (OMW), the EnterpriseOne databases, and so forth.

- OMW must be configured with transfer activity rules and allowed actions so that application development can occur.
- At a minimum, you must have a default project in OMW to which you have been added in the role of Developer.

CHAPTER 2

Working with Forms

This chapter provides an overview of forms, interconnectivity, and modes, and describes how to:

- Create forms.
- Work with Quick Form.

Understanding Forms

Use Form Design Aid (FDA) to create one or more forms for an application. A form is a graphical user interface where user interacts with the system. A form can be used to search and display data, as well as enter new data and modify existing data.

A single application can contain one or more forms. Usually, a find/browse form is the first form that appears in the application. It enables the user to locate a specific record with which to work. Upon selecting a record, a subsequent form such as a fix/inspect form can be used to provide details of the record. With the introduction of power forms, applications can use one single power form to locate a specific record and detail its detail records on one form.

A form has these elements:

- Form type

The form type establishes the basic purpose of a form. Each form type has default controls and processes.

- Business view

In an application, a business view (BV) links forms and tables efficiently by providing access only to that data required by the application. For example, if you have two tables with twenty columns each and the application only needs to access one column from one table and two columns from the other, you can make a BV that contains only those three columns. the application is more efficient because searches are limited only to those three columns, but the application still updates the actual tables when necessary. You must associate all forms, except the message form, with a BV.

- Controls

All objects on a form are controls. Controls include grids, check boxes, radio buttons, push buttons, subforms, and more.

- Properties

Properties define the appearance and function of the application, the forms in the application, and each control on each form.

- Data structure

A data structure defines the data that can be passed between forms. Data in the form data structure can be passed in or out of the form.

- Event rules

Event rules (ER) can contain processing instructions for specific events. Events are actions that occur on a form, such as clicking a button or using the TAB key to move out of a field. Use ER to attach business logic to any event. Events are triggered either as a result of user interaction with a control, such as clicking a button, or as a result of a system-controlled process, such as loading a grid.

Understanding Form Interconnections

You can call a form from a form. This kind of form interconnection falls into two categories, modal and modeless:

- Modal interconnections enable the user to view only one form at a time. After the child form begins, the user can not access the parent form until the child form is closed.

Additionally, the data connection between the parent and child is static. Input data structure items are populated when the child form is launched, and output data structure items are populated when the child form closes. Parent form ER that follows a modal form interconnection call executes after the child form is closed.

- Modeless interconnections enable the user to view multiple forms at the same time. After the child form is started, user can switch back and forth between the parent form and the child form. Additionally, data changes on a parent or child form are immediately reflected in all other open forms in the connection.

Modal is the default interconnection type. Modal interconnections are appropriate when you want to lead the user through a particular process in which a number of values must be input in a specific order. In this case, you want the user to completely fill out each form before moving on to the next one. Add and copy functions also lend themselves to modal processing because you want the user to complete the function before going on to others.

Modeless interconnections are valuable when the user needs to view or update a series of data records. Avoid using modeless form interconnections if both the parent and child forms should be presented to the user at the same time. A power form is more appropriate in this case.

The parent form in a modeless interconnect must be a find/browse form. The child form type can be fix/inspect or transaction forms (header and headerless detail) When user updates a record on the transaction form, the parent find/browse form automatically reflects the change.

When user closes the parent form, the system closes all its modeless children forms.

Parent form ER that follow a modeless form interconnection call execute immediately instead of waiting for the child form to return.

Configuring Forms at Design Time

The property settings for a form control its appearance, how it displays errors, and how it interacts with its underlying business views. When you first create a form, the system prompts you to configure its properties. You can change the properties of a form later in the design process.

Some property values are common to all form types, although many are shared by just a few. This table lists the property values for all form types. The descriptions given in some cases are for general use only. If a particular property significantly impacts a given form type, then those impacts are discussed in detail in the section devoted to that form type.

Property	Form Type	Description
Business View Name	Find/Browse Fix/Inspect Header Detail Headerless Detail Search & Select Parent/Child Browse Power Edit Power Browse Edit Subform Browse Subform Edit Portlet Browse Portlet	The BV that is attached to the form.
Data Structure	All form types	The data structure underlying the form that maps incoming and outgoing data.
Enable In-Your-Face-Error Display	Find/Browse Fix/Inspect Header Detail Headerless Detail Search & Select Message Form Parent/Child Browse Power Edit Power Browse	An option which, when selected, displays error text in red at the top of forms displayed in Web applications. Typically, the system indicates an application error by highlighting the Errors and Warnings link in the upper right area of the application. Selecting In-Your-Face-Errors has no noticeable effect on performance.
End Form on Add	Fix/Inspect Header Detail Headerless Detail	An option which, when selected, causes the system to close the form and return to the previous form after a user adds a record and clicks the OK button. If you want greater control over form flow, clear this option.

Property	Form Type	Description
Entry Point	Find/Browse Header Detail Headerless Detail Search & Select Message Form Parent/Child Browse Power Edit Power Browse	An option which, when selected, flags the form as being the one that you want users to see when they first launch the application. If you do not assign one of the forms as the entry point, the runtime engine loads the first form it finds in the application. FDA does not permit you to set more than one form as an entry point. Fix/Inspect forms cannot be used as entry points.
Fetch on Form Business View	Find/Browse Fix/Inspect Header Detail Search & Select Parent/Child Browse Edit Subform Browse Subform Edit Portlet Browse Portlet	An option which, when selected, causes the system to perform fetches based on information in the business view underlying the form. This option is unavailable if the form has no business view. Even though this setting is available for all form types, it only applies to Fix/Inspect, header/detail, headerless/detail, Power Edit forms with no grids, and subforms.
Fetch on Grid Business View	Find/Browse Headerless Detail Header Detail Search & Select Parent/Child Power Edit Power Browse	An option which, when selected, causes the system to update the grid during runtime based on information in the tables underlying the grid when the user clicks the OK button. This option is unavailable for entry unless the form contains a grid control.

Property	Form Type	Description
Form Guide	Find/Browse Fix/Inspect Header Detail Headerless Detail Search & Select Message Form Parent/Child Browse Power Edit Power Browse	The height and width in pixels to set the form guides. FDA indicates the optimum form size for different platforms by superimposing light blue lines that indicate height and width along the top and left-hand side of the form.
Form Name	All form types	The system name for the form. FDA names the form based on PeopleSoft EnterpriseOne naming standards (forms start with W, subforms with S), the application name, and the creation sequence (the first form is appended with an A, the second with a B, and so forth). You cannot change this property.
Form Type	Find/Browse Fix/Inspect Header Detail Headerless Detail Search & Select Message Form Parent/Child Browse Power Edit Power Browse	The type (find/browse, search & select, and so forth) of the current form. You cannot change this property.
Height Width	All form types	The height and width of the form in dialog units. You can change the form size by resizing it manually with the mouse or by using these properties to set the size precisely.

Property	Form Type	Description
Mapping Links	Power Edit Power Browse Edit Subform Browse Subform Edit Portlet Browse Portlet	The mapping of data between a parent and its child subforms.
Tile Wallpaper	Find/Browse Fix/Inspect Header Detail Headerless Detail Search & Select Message Form Parent/Child Browse Power Edit Power Browse	This option, when selected, displays multiple copies of the background image in a tile-like manner. If you select this option, then you cannot select the Wallpaper property. Additionally, you cannot select this property unless you have set either the Wallpaper File or the Wallpaper Full Name File property.
Title	All form types	The text name of the form. This name appears at the top of the form when users work with the application. By default, FDA enters the form type in this field. You should change it.
Total Controls on a Form	Find/Browse Fix/Inspect Header Detail Headerless Detail Search & Select Message Form Parent/Child Browse Power Edit Power Browse	A field that shows the current number of controls (including subforms) on the selected form. You cannot change this property

Property	Form Type	Description
Transaction	Fix/Inspect Header Detail Headerless Detail Power Edit Power Browse Edit Subform Edit Portlet	An option which, when selected, causes runtime to commit all changes at one time instead of individually. If the form makes a single database change or a group of unrelated changes, do not enable Transaction. However, if the form has a group of inserts that rely on each other and if the system should revert to the previously committed changes if a change fails or for another condition, then select Transaction. Transaction works differently for subforms and portlet forms.
Update on Form Business View	Fix/Inspect Header Detail Power Edit With No Grid Edit Subform Edit Portlet	An option which, when selected, causes the system to update the tables underlying the form (except those underlying the grid control) during runtime when the user clicks the OK button. This option is unavailable for entry unless you have attached a BV to the form.
Update on Grid Business View	Header Detail Headerless Detail Power Edit with Grid	An option which, when selected, causes the system to update the tables underlying the grid during runtime when the user clicks the OK button. This option is unavailable for entry unless the form contains a grid control.
Wallpaper	Find/Browse Fix/Inspect Header Detail Headerless Detail Search & Select Message Form Parent/Child Browse Power Edit Power Browse	An option which, when selected, displays a single image on the form as its wallpaper. If you enable this option, then you cannot enable the Tile Wallpaper property. Additionally, you cannot select this property unless you have set either the Wallpaper File or the Wallpaper Full File Name property.

Property	Form Type	Description
Wallpaper File	Find/Browse Fix/Inspect Header Detail Headerless Detail Search & Select Message Form Parent/Child Browse Power Edit Power Browse	The name of the image to use in conjunction with the Tile Wallpaper and Wallpaper properties. Use this property if the image is in the path code for the application.
Wallpaper Full File Name	Find/Browse Fix/Inspect Header Detail Headerless Detail Search & Select Message Form Parent/Child Browse Power Edit Power Browse	The name of the image to use in conjunction with the Tile Wallpaper and Wallpaper properties. Use this property if the image is not in the path code for the application.

Understanding Forms

This section provides an overview of form creation and describes how to:

- Create a form.
- Create a modal form interconnection.
- Create a modeless form interconnection.

Understanding Form Creation

Use FDA to create one or more forms that appear in an application. These forms are the visual interface for the end user of the application and enable that user to view, add, or modify data that is stored in one or more tables.

After you create the form, you can modify the system data for the form, such as its metadata, Help ID, and so forth.

Recommended FDA Configuration

FDA offers a variety of tool bars and panes that you can place on the desktop while you work. This lists provides recommendations for using these objects:

- Display the Main Toolbar, Layout, and Insert Controls tool bars at all times.
- Use the Insert Controls tool bar to insert controls on a form.
- Display the Application Tree View whenever possible.

The pane helps you keep the entire application in mind, and it provides an easy way to open other objects in the application.

- Display the Property Browser at all times and use it to set the properties for all objects.
- Display the Tab Sequence Toolbar when working with power forms and subforms.

This table lists the objects, why you would use them, and how to display them:

Object	Notes	Navigation in FDA
Main Toolbar	Provides standard actions such as open, cut, and paste.	View, Toolbars, Main Toolbar
Layout tool bar	Provides functions to align and space form controls precisely.	View, Toolbars, Layout
Insert Controls tool bar	enables you to place controls on a form with the push of a button.	View, Toolbars, Insert Controls
Status Bar	Displays a status bar at the bottom of the tool.	View, Status Bar
Application Tree View	Displays a pane showing the objects in the current application and their relationship to each other. You can open an object in this pane by double-clicking it. The pane can be filtered for language. You can also configure the tree to display its hierarchy by control or by business unit.	View, Application Tree View
Property Browser	Displays a pane showing the properties of the selected object. You can change object properties in this pane.	View, Property Browser
Data Dictionary Browser	Displays a pane where you can search for data dictionary (DD) objects. You can drag and drop DD objects from this pane to the form.	View, Data Dictionary Browser

Object	Notes	Navigation in FDA
Business View Columns Browser	Displays a pane where you can search for columns in a BV. You can drag and drop objects from this pane to the form.	View, Business View Columns Browser
Tab Sequence Toolbar	enables you to manipulate the tab sequence functionality, including changing the object being sequenced on the form.	View, Toolbars, Tab Sequence Toolbar

Creating a Form

This task provides a general overview of the typical form creation process. Many of the steps in this task are described in greater detail in other topics.

To create a form:

1. In Object Management Workbench (OMW), create an application or open an existing one, and then start FDA.
2. On PeopleSoft Form Design Aid, select the type of form you want to create.
3. Configure the form properties as appropriate.
4. If required, attach one or more BVs to the form by selecting Form, Business View, Add Business View.
Headerless detail forms permit you to attach two BVs to them; all other forms, except for message forms, permit you to attach only one. You cannot attach a BV to a message form.
5. Add and configure controls, as required.
Set the Data Item Information property for a control to attach DD items or BV columns to it. To attach ER to a control, right-click the control and select Event Rules. To set text variables for a control, right-click the control and select Text Variables.
6. Arrange the controls on the form so that they line up and are equally spaced, and then resize the form to fit.
7. Add menu and tool bar exits to the form.
8. Save and test the form.

Creating a Modal Form Interconnection

To create a modal form interconnection:

1. On Event Rules Design, select an event.
2. Click the Form Interconnect icon.
3. On Work with Applications, choose the application to which you are connecting.
Work with Forms displays available forms for the chosen application.
4. Select the form to which you want to connect (the target).
5. Select the version of the form to which you want to connect.

The Data Item column displays data items in the data structure of the target form. The keys in the primary unique index for the primary table of the BV are automatically set up as the data structure.

6. In the Available Objects column, choose the object that you want to pass and move it to the Data Structure-Value Column.
7. Indicate the direction of data flow between Value and Data Items.
As you click the direction arrow, it toggles through these options:
 - Data flows from the source to the target.
 - Data flows from the target to the source.
 - Data flows from the source to the target and, upon exiting the target, data flows back to the source.
 - Upon exiting the target, data flows back to the source.
 - No data flows either way.
8. Select the Include in Transaction option to include this interconnection for transaction processing.
This option is available for entry only if you are calling from a fix/inspect, header detail, or headerless detail form.
9. Click one of these buttons to add notes:
 - Structure Notes
 - Parameter Notes
10. After the data structure is defined, click the OK button.
Event Rules Design displays the form interconnect with this statement:
`Call (Application <name> Form <name>)`

Creating a Modeless Form Interconnection

To create a modeless form interconnection:

1. On Event Rules Design for the find/browse form that you want to use (the source), select an event.
2. Click the Form Interconnect button.
3. On Work with Applications, select the application to which you are connecting.
Work with Forms displays available forms for the chosen application.
4. Choose the form to which you want to connect (the target).
The Form Interconnect - Values to Pass window displays the data structure for the target form.
5. Choose the version of the fix/inspect form to which you want to connect.
The Data Item column displays data items in the data structure of the target form. The keys in the primary unique index, for the primary table of the BV, are automatically set up as the data structure.
6. Select the Modeless option.
7. In the Available Objects column, select objects that you want to pass and move them to the Data Structure-Value column.
8. Indicate the direction of data flow between Value and Data Items.
As you click the direction arrow, it toggles through these options:
 - Data flows from the source to the target.
 - Data flows from the target to the source.

- Data flows from the source to the target and, upon exiting the target, data flows back to the source.
 - No data flow.
9. Click one of these buttons to add notes:
 - Structure Notes
 - Parameter Notes
 10. After you define the data structure, click the OK button.

Event Rules Design displays the form interconnect with this statement:

```
Call (Application <name> Form <name>)
```

Working with Quick Form

Quick Form enables you to place multiple database fields on a form faster than choosing each data item individually. Select one or more data items, and Quick Form automatically places the fields 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.

Prerequisite

Attach a BV to the form.

Using Quick Form

To use Quick Form:

1. On PeopleSoft Form Design Aid, select Form, Quick Form.
2. On Quick Form layout, choose the number of columns of controls you want on the form and whether you want the columns arranged horizontally or vertically.
3. Choose the data items from the BV that you want to display and move them to the columns in form pane.
4. Use the buttons under the columns in the form pane to order the data items.
5. Click the OK button to place the fields on the form.

Quick Form remains open so that you can adjust the arrangement by changing the columns per row and the vertical and horizontal placement.

CHAPTER 3

Working with Form Controls

This chapter provides an overview of EnterpriseOne form controls and describes how to:

- Attach data items to a control.
- Associate a data item description with a field.
- Group controls.
- Set the tab sequence of controls on a form.

Understanding Form Controls

Use form controls to provide specific functions within an application such as these:

- Insert field controls on forms to display data, enter data, calculate data, store data permanently or temporarily, or pass data between fields and forms.
- Place check boxes on forms to provide for multiple selections, or radio buttons to indicate mutually exclusive selections.

A maximum of 250 controls can exist on a form. Form Design Aid (FDA) warns you if you are near the 250-control limit. Each form includes specific default controls, depending on the type of form that you are creating. However, you might need to add additional controls when you design the form. Choose from standard Windows graphical controls as well as PeopleSoft custom controls.

This table lists the controls, the forms on which they can be used, and their purpose:

Control	Valid Form Types	Description
Calendar	All form types	Use a calendar control to provide standard calendar capabilities to users which can be tied to some system events.
Check Box	All Form Types	Use one or more check boxes to provide the user with options that are not mutually exclusive.
Combo Box	All Form Types	Use combo box to provide user a drop down list of items.
Edit	All Form Types	Use edit fields to display data and to enable users to enter information for a specific instance of a data item.

Control	Valid Form Types	Description
Grid control	Browse Portlet Edit Portlet Find/Browse (default control) Header Detail (default control) Headerless Detail (default control) Power Browse (default control) Power Edit Reusable Browse Subform Reusable Edit Subform Search & Select (default control) Wizard (page only)	Use grids to display data and to enable users to enter information. Unlike an edit control, grid controls can show multiple data items and multiple table rows at once.
Group	All Form Types	Use this control to group other controls together visually.
Image	All Form Types	Use image controls to place a static or animated graphic on a form.
Media Object	All Form Types	Use media object controls to enable users to enter rich text and attach files to a form.
Parent Child	Browse Portlet Edit Portlet Reusable Browse Subform Reusable Edit Subform Parent/Child Browse (default control) Power Edit Wizard (page only)	Use parent child controls to present a hierarchical grid view or a tree view.
Push Button	All Form Types	Use a push button to initiate an action or a set of actions.
Radio Button	All Form Types	Use radio buttons to provide the user with sets of options. The radio buttons in each set are mutually exclusive.

Control	Valid Form Types	Description
Saved Query Control	Browse Portlet Edit Portlet Find/Browse Header Detail Headerless Detail Parent/Child Browse Power Edit Power Browse Reusable Browse Subform Reusable Edit Subform Search & Select Wizard (page only)	Use a saved query control to enable users to create and save data queries and to provide them with a set of premade queries from which to choose.
Static Text	All Form Types	Use static text as labels on the form.
Subform	Browse Portlet Edit Portlet Reusable Browse Subform Reusable Edit Subform Power Edit Power Browse Wizard (Wizard control only)	Use subforms to provide one BV and a group of controls associated with it. Place multiple subforms on a power form to provide multiple, collective data views on one form. The subform created this way will not be reusable by other forms or applications.
Subform Alias	Browse Portlet Edit Portlet Reusable Browse Subform Reusable Edit Subform Power Browse Power Edit Wizard	Use a subform alias to place a reusable subform on the form. A reusable subform is a subform with a data view and a set of controls associated with it.
Tab Control	All Form Types	Use tab controls to present a large number of controls on one or more tab pages. Power forms can have any number of tab controls, but all other forms are restricted to one. You cannot use tab control on a subform that is a tab page.

Control	Valid Form Types	Description
Tab Page	Only apply to tab controls	Use a tab page control to define one page in a tab control.
Text Block Control	All Form Types	Use text block controls to display free-form HTML text and plain text elements.
Text Search Control	Browse Portlet Edit Portlet Find/Browse Header Detail Headerless Detail Parent/Child Browse Power Edit Power Browse Reusable Browse Subform Reusable Edit Subform Search & Select Wizard (page only)	Use text search controls to enable full text searches against a generated index (as opposed to searching against the underlying BV).
Tree Control	All Form Types	Use tree controls to display a tree structure.
Wizard	Wizard (default control)	Use wizard controls to create self-directed applications. This specialized control is available only on wizard forms.

Understanding Form Control Design-Time Considerations

Some control property values are common to all controls, although many are shared by just a few. This table lists the property values for all control types. The descriptions given in some cases are for general use only. If a particular property significantly impacts a given form type, then those impacts are discussed in detail in the chapter devoted to that control. In addition to the standard controls you can add to a form using the Insert menu in FDA, this table includes grid columns which have their own property values separate from the grid itself and the standard menu properties such as OK, Save, Cancel, and so forth because they can act as push button controls. They are referred to as buttons in this table; for example, Select button.

Property	Control	Description
Allow Image Items	Media Object	An option to permit image items in the media object.
Allow OLE Items	Media Object	An option to permit OLE objects in the media object.
Allow RTF Text	Media Object	An option to permit RTF text to be included in the media object.
Allow Text Items	Media Object	An option to permit plain text objects in the media object. The plain text in the file is stored in the database.
Allowed in Saved Query	Edit Grid column Saved Query Control	An option to enable users to include the control as a filter value for a saved query. The property is available for entry only when a BV item is associated with the control.
Alternate Grid Row Format String	Grid	An HTML string that provides values for formatting the grid differently from the default system grid formatting. The system uses this string only if the Use Alternate Grid Row Formatproperty is selected. You can also choose the formatting at runtime with the Set Grid Row Format system function. Set it to <code><DEFAULT></code> to use the system grid formatting and <code><ALTERNATE></code> to use this HTML string instead. The system function can switch between formats regardless of whether Use Alternate Grid Row Format is enabled.
Always Hidden	Parent Child	An option to hide the grid portion of the control.
Automatic Scroll Horizontal	Edit	A property that indicates whether the user can see text that exceeds the width of the field. If you permit scrolling, you can choose to have the system automatically scroll ten characters to the right when the user types text, or you can display a horizontal scroll bar.
Automatic Scroll Vertical	Edit	A property that indicates whether the user can see text that exceeds the height of the field when the Lines property is set to <i>Multiple</i> . If you permit scrolling, you can choose to have the system automatically scroll down a page when the user presses ENTER, or you can display a vertical scroll bar.

Property	Control	Description
Automatically Find on Entry	Grid Parent Child	An option to cause runtime to populate the grid automatically when the form is entered.
Business View Name	Grid Parent Child Subform Subform Alias	A property that indicates the BV underlying the control. In all cases except one, the BV for the control is the same as the one for the form. On a header detail form, the grid control may have a BV that is different than the one underlying the controls that comprise the header.
Button Type	Push Button	A property that indicates the button type: OK, Cancel, Yes, No, and so forth. For form types except Message, the only option is Other.
Calendar Day View Visible	Calendar	An option to enable users to access the view of the calendar that shows a single day at a time.
Calendar Month View Visible	Calendar	An option to enable users to access the view of the calendar that shows an entire month at once.
Calendar Week View Visible	Calendar	An option to enable users to access the view of the calendar that shows an entire week at once.
Checked Value	Check Box Grid column	A property that is the value that the control returns when a user selects the control. The property applies to a grid column only when its Display Style property is set to <i>Check Box</i> .
Clickable	Grid column Image Static Text	An option to cause runtime to fire the Text Clicked event when the user clicks the control.
Client Edge	Group Box	An option to give the group box the appearance of depth.
Collapsible	Subform Subform Alias	An option to enable users to hide the content of the subform, displaying only its header.
Column Header One	Grid column	A property that shows the text to be displayed in the first line of the column heading.

Property	Control	Description
Column Header Two	Grid column	A property that shows the text to be displayed in the second line of the column heading.
Column Moved to Tree	Parent Child	A property that enables you to control which columns appear in the tree portion of the control.
Column Order	Grid Parent Child	A property that controls the order in which the grid columns appear in the grid, from left to right in English.
Column Sort Order	Grid Parent Child	A property that controls the order in which data returned to the grid is ordered for display.
Control ID	All control types	A property that shows the system ID of the current control. It cannot be changed.
Data Item Information	Check Box Combo Box Edit Grid column Radio Button Static Text	A property that shows the DD item or BV column associated with the control. BV column choices come from the BV associated with the form.
Data Structure	Subform Subform Alias	A property that shows the data structure being used to pass data between parent and child. If no data structure is attached to the form, the property is unavailable for entry.
Default cursor on add mode	Edit Grid column Media Object	An option to designate this field as the one in which the cursor appears initially when the form appears in Add mode. You can select this option for only one edit control on any given form.
Default cursor on update mode	Edit Grid column Media Object	An option to designate this field as the one in which the cursor appears initially when the form appears in Update mode. You can select this option for only one edit control on any given form.
Disable Copy	Parent Child	An option to prevent users from using the copy function in a cut/copy/paste operation.

Property	Control	Description
Disable Drag and Drop (Cut/Copy/Paste)	Parent Child	An option to prevent users from using the cut/copy/paste function.
Disable Move (Cut)	Parent Child	An option to prevent users from using the move/cut function in a cut/copy/paste operation.
Disable Page-at-a-Time Process	Grid Parent Child	An option to disable page-at-a-time processing. Page-at-a-time processing enables the runtime engine to fetch a single page of data only on the initial search call. If the user pages down, then runtime fetches only enough data to fill the next page. When disabled, runtime fetches and loads into memory all of the data at once. The number of rows that constitute a page of data is based on the Grid Row Countproperty.
Disable QBE	Grid column	An option to disable the QBE cell above a given column.

Property	Control	Description
Disabled	Calendar Check Box Close button Combo Box Delete button Edit Find button Grid Grid column Group Box Media Object OK button Parent Child Push Button Radio Button Saved Query Control Select button Static Text Subform Subform Alias Text Search Control	An option to disable the control, preventing user interaction with it (although it can still be seen). You can also enable and disable controls during runtime with the Enable Control and Disable Control system functions.
Display Customized Grid	Grid	An option to enable users the option to customize the grid through grid formats.
Display Export to Excel	Grid	An option to enable users the option to send the contents of the control to an Excel spreadsheet.
Display Export to Word	Grid	An option to enable users the option to send the contents of the control to a Microsoft Word file.
Display Import from Excel	Grid	An option to enable users the option to bring the contents of an Excel spreadsheet into the control.
Display Style	Grid column	An option to make cells in the column act (and appear) as check boxes.

Property	Control	Description
Do Not Clear After Add	Edit Grid column	An option to retain the data in the field after runtime performs an add function. After performing an add, runtime usually clears all form fields.
Editable	Subform Subform Alias	A property that indicates whether the subform type can be edited. Reusable edit subforms can be edited, reusable browse subforms cannot. Embedded subforms are editable or not based on its context within the parent form.
Expand All/Collapse All	Parent Child	An option to provide a button for the user to expand or collapse the entire tree.
Fetch on Form/Subform Businessview	Subform Subform Alias	An option to cause the system to perform fetches from the subform business view. This option is disabled unless you have attached a BV to the subform.
File Name	Image	A property that displays the name of the image file to be displayed in the control.
Filter Criteria	Edit Radio Button	A property that indicates whether the control value should be incorporated into the database fetch. If you want to use the value, then you must also choose the relational operator by which the value should be evaluated. In some cases, you can designate that the relational value should be chosen by the user at runtime instead.
Filter Criteria - Checked Filter	Check Box	A property that indicates whether the control should be incorporated in the database fetch as a filter criteria when the check box is selected. If you want to use this control as a filter, then you must also define how the filter value will be used by providing a relational operator or designate that the relational value should be set by the user at runtime instead.

Property	Control	Description
Filter Criteria - Unchecked Filter	Check Box	A property that indicates whether the control should be incorporated in the database fetch as a filter criteria when the check box is cleared. If you want to use this control as a filter, then you must also define how the filter value will be used by providing a relational operator or designate that the relational value should be set by the user at runtime instead.
Flat	Group Box	An option to give the group box border the appearance of height.
Form Name	Subform Subform Alias	A property that shows the system name for the subform. FDA names the subform based on PeopleSoft EnterpriseOne naming standards (subforms start with S), the application name, and the creation sequence (the first subform is appended with an A, the second with a B, and so forth). You cannot change this property.
Full File Name	Image	A property that displays the name and location of the image file to be displayed in the control.
Grid Row Count	Grid	A property that shows the number of rows in a grid that constitute a “page.”

Property	Control	Description
Height	Calendar Check Box Close button Combo Box Delete button Edit Find button Grid Group Box Image Media Object OK button Parent Child Push Button Radio Button Saved Query Control Select button Static Text Subform Tab Control Text Box Control Text Search Control Tree Control Wizard	A property that shows the height of the control in dialog units. You can change the height by typing a different value.
Hide HTML Row Selector	Grid	An option to hide the row selector when the form is viewed in HTML. You might choose to hide the selector either because it is not needed or because you want to prevent users from selecting entire rows.
Hide in Grid	Parent Child	A property that controls whether the designated tree column should be hidden in the grid. You should set this property to prevent the same column being displayed in two places, both in the tree and the grid.

Property	Control	Description
Hide Query By Example	Grid Parent Child	An option to hide the query-by-example (QBE) line above the grid. You might choose to hide the QBE either because it is not needed or because you want to prevent the users from defining query criteria on grid columns.
Indent and Outdent	Parent Child	An option to enable the user to change the horizontal position of a node in the tree.
Justification	Check Box Edit Radio Button Static Text	A property that indicates whether the text of the control will be left-, center-, or right-justified.
Key Relations	Parent Child	A property that defines the values to use for certain key fields when the system builds queries. When you map a child key to a parent key, then the system sets the value of the parent variable equal to the associated child value for purposes of filtering the query.

Property	Control	Description
Left	Calendar Check Box Close button Combo Box Delete button Edit Find button Grid Group Box Image Media Object OK button Parent Child Push Button Radio Button Saved Query Control Select button Static Text Subform Subform Alias Tab Control Text Box Control Text Search Control Tree Control	<p>A property that shows the distance from the left edge of the form to the left edge of the control in dialog units. You can change the horizontal placement of the control by typing a different value.</p>
Lines	Edit	<p>An option to enable the field to display multiple lines of text. If you select this option, usually you also set the Automatic Scroll View property such that users can view the text if it exceeds the height of the control.</p>
Load Text by Instance	Subform Alias	<p>An option that defines how jargon will be loaded for reusable subforms. If selected, the system fetches jargon based on the form that is using the subform. If cleared, the system fetches jargon based on the application where the subform is defined.</p>

Property	Control	Description
Location Indicator Feature	Parent Child	An option to enable or disable location indicator functionality for the parent child control. If selected, the system enables the user to choose to display a location indicator for each tree node.
Maintain Aspect Ratio	Image	An option to maintain the original width-to-height ratio when the image is resized.
Mapping Links	Subform Subform Alias	A property that shows the mapping of data between a parent and its child forms. You must define this mapping to pass data between parent and child forms.
Menubar Separator	Close button Delete button Find button OK button Select button	An option to display a line above the name of the control when viewed in a menu list.
Modal Frame	Group Box	An option to give the group box the appearance of height.
Move Up and Down	Parent Child	An option to enable the user to change the vertical position of a node in the tree.
Multi-Line Edit	Grid	An option to post rows in groups of three to five to the database in the background. When disabled, in low interactivity mode, runtime posts each time the user tabs out of the row, forcing the user to wait for a refresh before continuing. This option does not apply to high interactivity.
Multiple Select	Grid Parent Child	An option to enable users to choose multiple lines to affect with a single operation, such as cut-and-paste.
New Text Item on Open	Media Object	An option to cause the control to create a new text item automatically when the user first opens the control.
No Adds On Update Grid	Grid Parent Child	An option to prevent users from adding new records to the control. Users can still edit existing rows, however.

Property	Control	Description
No display if currency is OFF	Check Box Edit Grid column Radio Button Static Text	An option to hide the control if currency is disabled.
Node ID Column	Parent Child	An option to use this column as a unique identifier for that row. The system functions, Insert Grid Buffer Row By Node ID and Get Related Node ID , require a node ID column to work correctly.
Number of columns joined to header	Grid Parent Child	A property that enables backwards compatibility of certain legacy applications. Note. Set this property value to zero for new applications.
Overrides Button	Check Box Combo Box Edit Grid column Radio Button	A button to set DD overrides for the control.
Parent	Subform Subform Alias	A property that indicates the form that acts as the parent to the current subform.
Password	Edit	An option to cause the system to display an asterisk in place of the character the user actually typed. This feature is most often used to help protect passwords.
Prevent Resizing	Image	An option to prevent the image from being resized during design time.
Position of Saved Query links	Saved Query Control	A property that indicates where the saved query links appear relative to the main body of the control itself.
Process All Rows in Grid	Grid	An option to cause runtime to apply row changed and row exited logic to all rows, no matter their state.

Property	Control	Description
Product Synch Mapping	Parent Child	A property that indicates how the data in the column will be used if the parent child control on which the grid resides has the Product Synch Mode property enabled.
Product Synch Mode	Parent Child	An option to use the control for process mapping applications (that is, rapid manufacturing).
Progress Indicator	Wizard	An option to display a progress indicator during runtime. You must also indicate its type, if the indicator is displayed.
Read Only	Edit Media Object	An option to prevent users from changing the value in the control.
Reclaim Whitespace	Grid	An option to cause runtime to shrink the control vertically so it displays only those rows which contain data (HTML only).
Re-entry Save	Wizard	An option to enable users to save their input, quit, and then re-launch the wizard later, starting at the point where they saved.
Required	Combo Box	An option to force users to enter a value into this control before being able to execute a form-level action such as OK or Save (except Cancel).
Required entry field	Edit Grid column	An option to force users to enter a value into this control before being able to execute a form-level action such as OK or Save (except Cancel).
Reusable	Subform Subform Alias	A property that indicates whether the subform is reusable. Subforms that were created independently of a parent (such as, reusable browse subform and reusable edit subform) are reusable and may be referenced by an alias on any number of power forms. Subforms that were created as a control on a parent form cannot be referenced with an alias and are therefore not reusable.

Property	Control	Description
Save Data	Check Box Combo Box Edit Grid column Radio Button Text Box Control	An option to cause the system to save the value in this control locally in case of a communication failure between the client and server upon commit.
Show details of all tree nodes	Parent Child	An option to display one tree node for each grid row.
Show Header	Subform Subform Alias	An option to display the header of the subform during runtime.
Sort Direction	Grid column	A property that displays the order in which column data will be sorted (ascending or descending).
Sortable by End User	Grid column	An option to enable the user to reorder the grid contents based on this column.
Sorted	Combo Box	An option to sort the items in the drop-down box alphabetically.
Static Edge	Group Box	An option to give the group box border the appearance of depth.
Subform Application Name	Subform Alias	A property that shows the name of the application that contains the reusable subform to which the alias points.
Support Multiple Currencies	Grid column	An option to enable the column to handle and display amounts in differing currency types.
Suppress Validation and Save	Wizard	An option to prevent runtime from validating and saving the data in the control.

Property	Control	Description
Tab Stop	Calendar Check Box Combo Box Edit Grid Group Box Media Object Parent/Child Push Button Radio Button Saved Query Control Static Text Text Search Control	An option to enable users to press TAB to move the focus to the control.
Task List	Wizard	An option to display the task list during runtime.
Text is Overridden	Check Box Combo Box Edit Grid column Radio Button Static Text	If the control is associated with a BV column or DD item, an option to change the title (the text that users can see) in this instance.

Property	Control	Description
Title	Calendar Check Box Combo Box Edit Grid Group Box Image Media Object Parent Child Push Button Radio Button Saved Query Control Static Text Subform Tab Control Tab Page Text Block Control Text Search Control Tree Control Wizard	A property that shows the text that the user can see. If the control is associated with a particular BV column or DD item, then you might need to choose an overrides option to enable the field if you want to change its displayed text in a given instance.
Tool Tip	Image	A property that shows the text that appears in the tool tip for the image. The tool tip is the text that appears when the user hovers over the object.
Toolbar	Close button Delete button Find button OK button Select button	An option to display the control as a button on the standard application tool bar.

Property	Control	Description
Toolbar Separator	Close button Delete button Find button OK button Select button	An option to display a line to the left of the control button when it appears on the standard application tool bar.
Top	Calendar Check Box Close button Combo Box Delete button Edit Find button Grid Group Box Image Media Object OK button Parent Child Push Button Radio Button Saved Query Control Select button Static Text Subform Subform Alias Tab Control Text Search Control Text Box Control Tree Control	A property that shows the distance from the top edge of the form to the top edge of the control in dialog units. You can change the vertical placement of the control by typing a different value.

Property	Control	Description
Transaction	Subform	An option to cause runtime to commit all changes at one time instead of individually. If the subform makes a single database change or a group of unrelated changes, do not select Transaction. However, if the subform has a group of inserts that rely on each other and if the system should revert to the previously committed changes if a change fails or for another condition, then select Transaction.
Unchecked Value	Check Box Grid column	A property that is the value that the control returns when a user clears the control. The property applies to a grid column only when its Display Style property is set to <i>Check Box</i> .
Update Mapping Link	Push Button	An option to cause the parent to push data to its child before runtime performs any button processing.
Update Mode	Grid	An option to make the grid input-capable. Runtime implements this option only when the grid resides on input-type forms such as fix/inspect or header detail.
Update on Form/Subform Businessview	Subform	An option to cause the system to update the tables underlying the subform (except those underlying the grid control) during runtime when the user clicks the OK button. If you want greater control over subform updates, clear this option. This option is unavailable for entry unless you have attached a BV to the subform.
Use Alternate Grid Row Format	Grid	An option to use the HTML string in the Alternate Grid Row Format property to format the grid instead of using the default system formatting for grids.

Property	Control	Description
UTC Display Format	Edit Grid control	<p>A property that controls how to display time and date fields. This property value is available for entry if the BV column or DD item associated with the control has a type of U-Time. This type of data represents date and time in Coordinated Universal Time (UTC) format. The offset is from Greenwich Mean.</p> <p>This field is not a mask. Whatever format you choose comprises the only data that is saved to the database. Therefore, if you choose a format that displays only the date and not the time, then only the date is written to the database.</p>
Value	Radio Button	A property that is the value that the control returns when a user selects the control.

Property	Control	Description
Visible	Calendar Check Box Combo Box Edit Grid Grid column Group Box Image Media Object Parent Child Push Button Radio Button Saved Query Control Static Text Subform Subform Alias Tab Control Text Box Control Text Search Control Tree Control Wizard	An option to enable users to see the control. You can also hide and show the control during runtime with the system functions, Hide Control and Show Control .

Property	Control	Description
Width	Calendar Check Box Close button Combo Box Delete button Edit Find button Grid Grid column Group Box Image Media Object OK button Parent Child Push Button Radio Button Saved Query Control Select button Static Text Subform Tab Control Text Box Control Text Search Control Tree Control Wizard	A property that shows the width of the control in dialog units. You can change the width by typing a different value.
Wildcard	Edit	At option to enable users to use the asterisk character as a wildcard when performing searches.
Wrap Text	Grid column	An option to enable text to wrap if it exceeds the column width. Otherwise, the system truncates column text.

Attaching Data Items to a Control

This section provides an overview of the relationship between data items and controls and describes how to:

- Attach a data item to a control.
- Override a default DD trigger.

Understanding the Relationship between Data Dictionary Items and Controls

On a form, you can use any data item that is in the BV and any data item from the DD. BV items are associated with the database; they are retrieved from and updated to the database tables. DD items are not connected to the database; their values are not retrieved from nor updated to the database tables by the system. On a form, BV items appear with a blue box in the left corner, and DD items appear with a yellow box in the left corner. Use DD items as work fields for ER or fields that are not directly connected to the database table.

You can associate data item values with these control types:

- Check boxes
- Edit controls
- Grid controls (columns)
- Parent child controls (columns)
- Radio buttons

Radio buttons must be associated with DD items. All radio buttons with the same DD item are considered to be a group. Selecting one radio button within the group automatically clears all other radio buttons in the same group.

- Combo boxes

The rest of the controls types either cannot be associated with a data item, such as push buttons, or essentially act as containers for controls, such as subforms and tab pages, which may in turn be associated with a data item, according to their type.

If you want the user-entered value for a control to update a database record, then the control must be associated with a BV item. For example, a check box can be associated with the database field called Taxable. In the check box properties, the Checked value should be *Y* and the Unchecked value should be *N*. You can use these values in ER as well.

Each DD item has a set of properties (such as Default Value) and behaviors (such as Format Rules and Edit Rules). When you create a DD item, you define its default properties and behaviors at that time. You can override previously-defined DD items at the application level, which enables you to further customize how data items behave at runtime. You can disable default properties and behaviors for a specific item, as well as override to use different values and triggers.

Attaching a Data Item to a Control

To attach a data item to a control:

1. Choose the control, and then click Data Item Information in the Property Browser and click the ellipses button that appears in the Data Item Information field.
2. On Control Properties, perform one of these tasks:

- To attach a data item from the BV attached to the form, click the Business View Items tab and double-click a data item.
 - To attach a data item from the DD, click the Data Items tab, search for the data item you want, and then double-click it.
3. To override the data item name as it appears on the form, click the General tab, click Override Text, and then enter the name you want to use in the Event Rules Title field.

The Text is Overridden property in the Property Browser changes to Yes. To return the display name to the DD name, change Text is Overridden to No.

Overriding a Default Data Dictionary Trigger

To override default DD properties and triggers.

1. On the form, double-click a control with an attached DD item.
2. On Edit Properties, click the Overrides tab, and then click Data Dictionary Overrides.
3. On Data Dictionary Overrides, to disable one or more triggers, choose the ones you want to disable in the Disable box.
4. To override a trigger, click the applicable trigger type and complete the form that appears.

For example, if you click Default Value, the Override Default Value form appears and you would make your changes there.

Associating a Data Item Description with a Field

This section provides an overview of the relationship between data item descriptions and fields and discusses how to display the title of a data item associated with a field.

Understanding the Relationship between Data Item Descriptions and Fields

When you insert an edit control on the form and associate a data item with it, FDA does not automatically associate a description with the field. If you want to display the row description for the data item automatically on the form, you must associate the data item description with the field.

When you associate a description with an edit control, a description of the value in the control automatically appears next to it. Associating a description is optional but is useful as a visual aid on the form.

For example, suppose that the address book number, for example, 1001 appears in an edit control or in an edit control. When the user presses TAB to move out of the control, the address book name, XYZ Company, appears next to the control as associated description.

Displaying the Title of a Data Item Associated with a Field

To display the title of a data item associated with a field:

1. Click the control, and then select Edit, Associate Description.

FDA creates a static text block and attaches it to the cursor so that you can place it where you want.

2. Click the approximate location on the form where you want the description to appear.

Grouping Controls

Sometimes, you want a set of controls to be taken as a whole by the user. The classic example is when you want the user to choose one item from a list of items, and this situation is typically resolved by creating a radio button for each option and then grouping them so that the user can choose only one at a time.

To provide the user with a visual cue that the controls are grouped, you can place a group box around the controls. Use the Title property for the group box to label the group on the form.

Setting the Tab Sequence of Controls on a Form

This section provides an overview of tab sequence and describes how to change the tab sequence on a form.

Understanding Tab Sequence

The tab sequence of the controls determines the order in which the cursor travels through the controls on the form. Each control that is designated as a tab stop is numbered to reflect how the cursor will travel. For example, when the user opens the form, the cursor appears on the control labeled number one. When the user presses TAB, the cursor moves sequentially to the control (tab stop) labeled number two 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 you place the controls on the form.

These properties can override the first tab stop:

- Default cursor on add mode
- Default cursor on update mode

Changing the Tab Sequence on a Form

To change the tab sequence:

1. On the form with which you are working, select Layout, Tab Sequence.
FDA displays each control so you see its number in the tab sequence instead of its name.
2. Click the controls in the order in which you want the cursor to travel.

The first control that you click becomes number one, the second becomes number two, and so on. To reset the numbers to their original values, right-click anywhere in the window.

Note. Subforms contain their own sequence orders for objects on the subform. Therefore, you set the tab sequence on each subform within the subform itself. The subform, then, becomes a single object in the tab sequence for the power form.

CHAPTER 4

Working with FDA Compare

This chapter provides overviews of FDA Compare and its color scheme and discusses how to use FDA Compare.

Understanding FDA Compare

The FDA Compare tool in Form Design Aid (FDA) enables you to compare one version of an application to another. You can compare them on the application level to determine whether forms have been added, deleted, or rearranged and whether the properties have changed. You also can compare the forms in the applications to each other to see whether controls have been added, deleted, or rearranged and whether the properties have changed.

Additionally, you can compare two different applications as well. This feature is useful when you have made a new application by copying an existing one and then modifying it. Then, when you upgrade, you can not only compare the base application to its new counterpart, but also you can compare a custom application.

While working with the target object, you can use all FDA functions except creating new forms. While comparing, you can change the target object to match the source object. If an object exists in the source but not in the target, you can copy it to the target. If an object exists in both but is different in some way, you can merge the specifications from the source to the target to make them identical.

As a software developer, you might use FDA Compare as the final step before checking in the changes to ensure that you made all of the changes you intended to make. In this way, you can also make sure that you did not move a control or make a property change unintentionally.

As an administrator, you might use FDA Compare to see the changes between a software update and the pristine or current implementation. If you have performed a number of customized modifications, you can more carefully implement the software changes without fear of ruining the customization.

Understanding the FDA Compare Color Scheme

FDA Compare uses colors and letters to highlight the differences between the source and target objects. This table lists the default colors and their meanings:

Color	Meaning
Black	The object exists in both the source and target and is the same in both versions.

Color	Meaning
Red	The object exists in both the source and target but is not the same in both versions.
Green	The object exists only in the target.
Blue	The object exists only in the source.

In the form workspace, the controls that differ between the versions are marked with a symbol in a color indicating the type of change. When you click a control, the browsers highlight the changes in specific property values with the same color coding. To change the color scheme, select View, User Options.

Note. Nodes in the Application Tree View browser appear in a color if one or more of their children are different. Therefore, while the node itself may not be different, you will find that an object is different when you navigate deeper into the tree.

Using FDA Compare

This section describes how to use FDA Compare.

Forms Used To Use FDA Compare

Form Name	Form ID	Navigation	Usage
Select Source Object	NA	Form Design Aid, File, Compare Mode	Enable or disable compare mode (toggle). When you enable compare mode, the system prompts you to identify the application that you want to use as the basis for the comparison.

Merging Specifications Using FDA Compare

To merge specifications using FDA compare:

1. In FDA, open the target application.

This is the application that you have been working on or that your company uses. It is the version of the application that you might want to change when you compare it to the source, which is a base version of the application.

2. Select a source application (the one to use as the basis for the comparison).
3. To change the view properties in the source or target property browser, select one of these options from the drop-down menu at the bottom of the pane:

- All

Displays all the properties and values in alphabetic order for the object including the translation properties.

- Standard

Displays only the standard properties in alphabetic order.

- Translation

Displays only the translation properties in alphabetic order.

4. Compare the source application properties (on the left) to the local target properties (on the right).
5. To merge properties from the source object to the target object, right-click in the property source property browser and select Merge to Target from the menu.

CHAPTER 5

Understanding Browse Portlet Forms

This chapter provides overviews of browse portlet forms and events and discusses how to:

- Generate portlets for Collaborative Portal.
- Update an FDA-created portlet after initial installation

Understanding the Browse Portlet Form

Portlet forms are the form types you use to create portlets intended for use in the Enterprise Portal, Collaborative Portal, or any other framework which supports the JSR 168 specification. A *portlet* is a single piece of portal content; it becomes the window through which users can interact with the application while using a portal. A *portal* is an application that aggregates portlets into pages and provides authentication, authorization and administration tools. Planet PeopleSoft is an example of a portal. The JSR 168 specification is a Java standard portlet API supported by technology companies including PeopleSoft, IBM, SUN, SAP, Plumtree, and Apache.

Note. Portlet forms are supported in PeopleSoft EnterpriseOne version 8.11 and might not function correctly in older versions of the software.

EnterpriseOne Tools provides two types of portlet forms: browse portlet and edit portlet. Browse portlet facilitates the display of and queries for information. Edit portlet facilitates user edit functions-adding, changing, and deleting table information.

Portlet forms have these general characteristics:

- All regular controls can be placed on portlet forms.
- Multiple tab controls are permitted.
- Toolbar and form/row exits are not permitted on a portlet form.
- You must add action buttons to a portlet form.

No default action buttons are defined by Form Design Aid (FDA).

- Default actions can be placed anywhere on the portlet form.
- Portlet forms do not contain scroll bars; you must display all controls within the form.
- Portlet forms support the Fetch on Business View and Update on Business View form property options.
- You can nest a reusable subform in portlet forms.

If you include a form interconnection in a portlet, the portal containing it will automatically resize the portlet to its maximum size when the user triggers the interconnection. When the user closes the application, the portlet returns to its normal size.

Portlet Personalization

One feature of portlet forms that is unique among the FDA form types is portlet personalization. Users can toggle between standard mode (where they interact with the application) and personalization mode, where they can choose different options that affect how the application runs while in standard mode.

For example, say that the application by default displays all accounts that are overdue by 30 days or more. Users might be able to choose instead to view accounts that are overdue by 60 days instead, or by accounts in a certain region that are overdue. Alternatively, they might have an option which displays the information in a summarized format instead of showing all the data you show by default. Personalization values are user-specific.

You control what options users have. Every option you provide, be it as simple as a filtering function or as complex as invoking additional functionality, is a data dictionary (DD) item passed in with the data structure. The DD items appear when the user toggles to personalization mode, and the user can trigger them by choosing the options and then returning to standard mode. If you do not include any such items in the data structure, then personalization mode is disabled for that portlet.

Because the available options are based on the form's data structure, those values are passed in by runtime to the form if they have been set.

Understanding Browse Portlet Form Events

Runtime provides three events that are specific to portlet form types:

- **Portlet is Initialized**

This event occurs when runtime initializes the portlet form and before it initializes any forms which the portlet might contain.

- **Personalization Applied**

This event occurs immediately after initialization to apply previously saved personalization data. It occurs again when the user clicks the Save button in personalization mode.

- **Portlet is Exited**

This event occurs when the portlet is unloaded by the container. Typically, this occurs as the user logs out or the session times out.

In addition to the portlet-specific events, browse portlets support these general events:

- **Grid Record is Fetched**
- **Write Grid Line-Before**
- **Last Grid Record Has Been Read**
- **Notified By Child**

Generating Portlets for Collaborative Portal

This section provides an overview of generating portlets for Collaborative Portal and discusses how to deploy an FDA-created portlet.

Prerequisites

Before you complete the tasks in this section:

- Create or obtain a portlet application that was created in FDA.
- Generate the application as usual using eGenerator.

Understanding Portlet Generation

Generating an application based on portlet form types is similar to generating an application based on other form types, the difference being that you must also generate a portlet descriptor file, `portlet.xml`, that tells the portal which applications are available for the portlet (the `WebClient_Portal.war` file). This is an overview of the high-level tasks you must complete to create and view a portlet using FDA:

1. Create an application in FDA using the portlet form types.
2. Generate the application (that is, the portlet) with eGenerator.
3. Generate the `portlet.xml` file into `WebClient_Portal.war` file with eGenerator.
4. Install the `.war` file in Collaborative Portal.
5. To view it, add the portlet to a page in the portal.

This subsection describes how to deploy an FDA-created portlet so that you can install it in Collaborative Portal.

Deploying an FDA-Created Portlet

To deploy an FDA-created portlet:

1. Launch eGenerator and select Generate, Portlet Deployment.
2. Choose the `.war` file to generate.

Typically, the `.war` file for the local version of Collaborative Portal resides in `C:\B9\system\Generator\WebClient_Portal.war`.

eGenerator reads the local specifications for EnterpriseOne and lists all portlets it discovers.

3. Clear the check box next to any portlet that you do not want to generate, and click the Start button.

If multiple versions of an application exist, each version is listed.

4. Select the version you want to generate.

eGenerator creates a new portlet descriptor file and bundles it into a new `WebClient_Portal.war` file.

Updating an FDA-Created Portlet after Initial Installation

This section provides an overview of updating an FDA-created portlet after initial installation and discusses how to:

- Add a new portlet application.
- Delete an existing portlet application.

Understanding Portlet Updates

When you modify an existing FDA-created portlet in FDA, you need only regenerate the EnterpriseOne application to promote the application to the Collaborative Portal. However, if you are removing existing portlets or are adding new FDA-created portlets to Collaborative Portal, you must use eGenerator to update the portlet descriptor, portlet.xml, in the WebClient_Portal.war.

Adding a New Portlet Application

To add a new FDA-created portlet:

1. Launch eGenerator and select Generate, Portlet Deployment.
2. Choose the .war file to update.

Choose the same war file that you deployed previously.

3. Clear the check box next to any portlets that you do not want to generate, and click the Start button.

You should always select all portlets you want to be deployed to Collaborative Portal because after installation, the new WebClient_Portal.war replaces any previous version that exists on Collaborative Portal server.

eGenerator modifies the portlet descriptor file, portlet.xml, in the .war file.

Deleting an Existing Portlet Application

Regenerating the portlet.xml by removing the portlet definitions does not remove the corresponding portlets from Collaborative Portal when you perform a portlet application update.

See *EnterpriseOne PeopleTools 8.94 Installation and Configuration for Collaborative Portal*

CHAPTER 6

Understanding Edit Portlet Forms

This chapter discusses edit portlet forms.

See Also

Chapter 5, “Understanding Browse Portlet Forms,” Generating Portlets for Collaborative Portal, page 48

Edit Portlet Forms

Portlet forms are the form types you use to create portlets intended for use in the Enterprise Portal, Collaborative Portal, or any other framework which supports the JSR 168 specification. A *portlet* is a single piece of portal content; it becomes the window through which users can interact with the application while using a portal. A *portal* is an application that aggregates portlets into pages and provides authentication, authorization and administration tools. Planet PeopleSoft is an example of a portal. The JSR 168 specification is a Java standard portlet API supported by technology companies including PeopleSoft, IBM, SUN, SAP, Plumtree, and Apache.

Note. Portlet forms are supported in PeopleSoft EnterpriseOne version 8.11 and might not function correctly in older versions of the software.

EnterpriseOne Tools provides two types of portlet forms: browse portlet and edit portlet. Browse portlet facilitates the display of and queries for information. Edit portlet facilitates user edit functions-adding, changing, and deleting table information.

Portlet forms have these general characteristics:

- All regular controls can be placed on portlet forms.
- Multiple tab controls are permitted.
- Toolbar and form/row exits are not permitted on a portlet form.
- You must add action buttons to a portlet form.

No default action buttons are defined by Form Design Aid (FDA).

- Default actions can be placed anywhere on the portlet form.
- Portlet forms do not contain scroll bars; you must display all controls within the form.
- Portlet forms support the Fetch on Business View and Update on Business View form property options.
- You can nest a reusable subform in portlet forms.

If you include a form interconnect in a portlet, the portal containing it will automatically resize the portlet to its maximum size when the user triggers the interconnect. When the user closes the application, the portlet returns to its normal size.

Portlet Personalization

One feature of portlet forms that is unique among the FDA form types is portlet personalization. Users can toggle between standard mode (where they interact with the application) and personalization mode, where they can choose different options that affect how the application runs while in standard mode.

For example, say that the application by default displays all accounts that are overdue by 30 days or more. Users might be able to choose instead to view accounts that are overdue by 60 days instead, or by accounts in a certain region that are overdue. Alternatively, they might have an option which displays the information in a summarized format instead of showing all the data you show by default. Personalization values are user-specific.

You control what options users have. Every option you provide, be it as simple as a filtering function or as complex as invoking additional functionality, is a data dictionary (DD) item passed in with the data structure. The DD items appear when the user toggles to personalization mode, and the user can trigger them by choosing the options and then returning to standard mode. If you do not include any such items in the data structure, then personalization mode is disabled for that portlet.

Because the available options are based on the form's data structure, those values are passed in by runtime to the form if they have been set.

Edit Portlet Form Design-Time Considerations

In addition to standard form properties, you can control the transaction boundary for edit portlets. You can choose to process each action (that is, insert, update, or delete) on the form separately (*Transaction Disabled*), or you can process all of the actions at once (*Portlet Only Transaction*). Choose the latter if processing each action separately would cause data integrity issues. Transaction processing for edit portlets is identical to that for subforms.

See Also

[Chapter 5, "Understanding Browse Portlet Forms," Generating Portlets for Collaborative Portal, page 48](#)

Edit Portlet Form Events

Runtime provides three events that are specific to portlet form types:

- **Portlet is Initialized**

This event occurs when runtime initializes the portlet form and before it initializes any forms which the portlet might contain.

- **Personalization Applied**

This event occurs immediately after initialization to apply previously saved personalization data. It occurs again when the user clicks the Save button in personalization mode.

- **Portlet is Exited**

This event occurs when the portlet is unloaded by the container. Typically, this occurs as the user logs out or the session times out.

In addition to the portlet-specific events, browse portlets support these general events:

- **Grid Record is Fetched**
- **Write Grid Line-Before**
- **Last Grid Record Has Been Read**
- **Add Record to DB - Before**
- **Add Record to DB - After**
- **Update Record to DB - Before**
- **Update Record to DB - After**
- **Write Grid Line-After**
- **Post Commit**
- **Notified By Child**

CHAPTER 7

Understanding Find/Browse Forms

This chapter discusses find/browse forms.

Find/Browse Forms

Find/Browse forms are used to query business views (BVs) and to select records from BVs for operations.

Find/Browse Events

These events can occur on the find/browse form during runtime:

- **Dialog is Initialized**
- **Post Dialog is Initialized**
- **Grid Record is Fetched**
- **Write Grid Line-Before**
- **Write Grid Line-After**
- **Last Grid Record Has Been Fetched**
- **Call is Alerting**
- **End Dialog**
- **XAPI Subscribe Event**

Find/Browse Runtime Processing

This section describes how runtime processes find/browse forms.

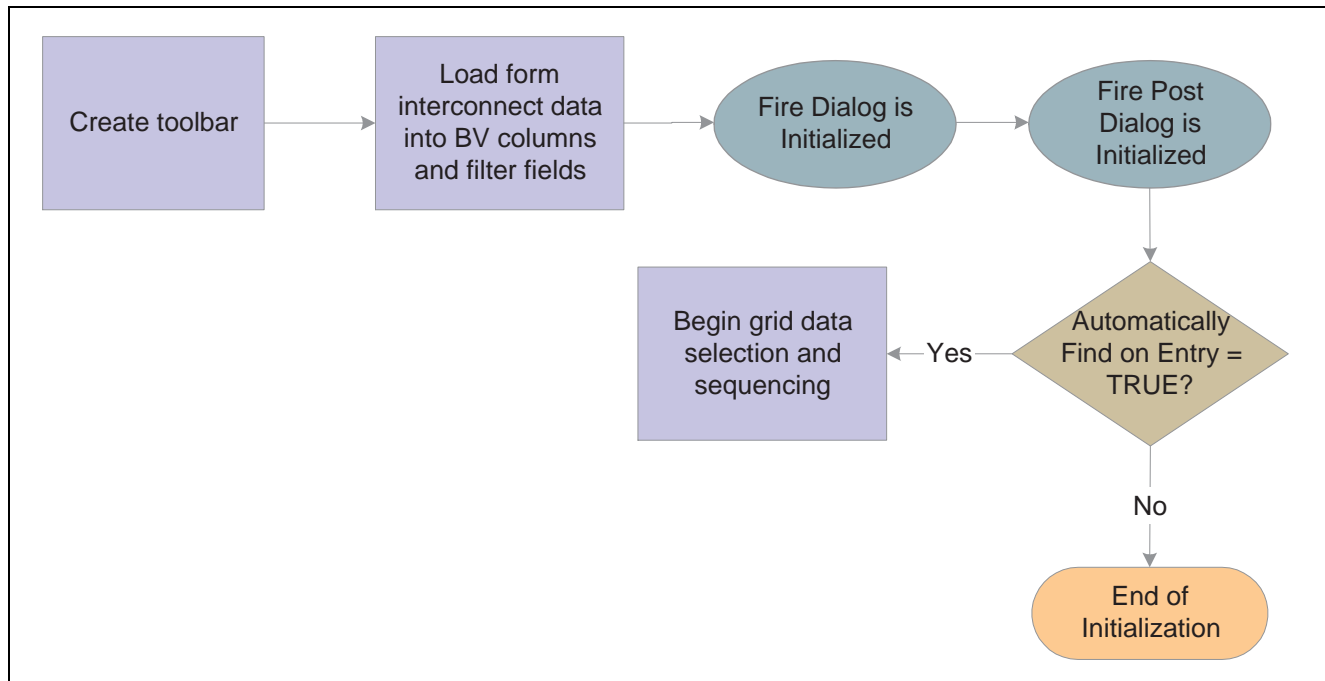
Dialog Initialization

When a find/browse form is called, runtime initializes these items in this order:

1. Thread handling
2. Error handling process
3. Business view columns (BC)

4. Form controls (FC)
5. Grid fields
6. Static text
7. Helps
8. Event rules structures

This flowchart illustrates the tasks that runtime performs after initializing these objects to complete dialog initialization:



Find/Browse dialog initialization

Grid data selection and sequencing occurs at the control level.

See [Chapter 21, “Understanding Grid Controls,” Grid Control System Functions, page 160](#).

The system creates an internal structure that represents the data selection and data sequencing requirements specified by the user. The system then passes this to the database engine to perform the actual database select and sequencing. The data used for selection is based on values from filter fields and query-by-example (QBE) columns. The system holds the data until the data is retrieved.

Find Button

The Find button is a standard button on find/browse forms that appears by default. When the user clicks it, runtime fires the **Button Clicked** event. If no errors exist in the filter fields, runtime performs data selection and sequencing for the grid. After reloading the grid with the fetched data, runtime fires the **Post Button Clicked** event.

Select Button

The Select button is a standard button on find/browse forms that appears by default. When the user clicks it, runtime fires the **Button Clicked** event. If no errors occur, runtime writes the values from the selected row to the BC and fires the **Post Button Clicked** event. Then it fires the **End Dialog** event and initiates the dialog close process.

Close Button

The Close button is a standard button on find/browse forms that appears by default. When the user clicks it, runtime fires the **Button Clicked** and **Post Button Clicked** events in immediate succession. If no errors occur, runtime attempts to close all of the modeless child forms, if any exist. If any of these child forms cannot be closed, the Close button process is terminated. Otherwise, runtime fires the **End Dialog** event and initiates the dialog close process.

Dialog Close

Find/Browse can be closed either by the user (typically by clicking the Select or Close buttons) or by the system. After performing any control-level close processing that might need to occur, runtime closes the form. If the event has not already occurred, runtime fires the **End Dialog** event. Then it performs these tasks in this order:

1. Load form interconnect data from BC for database commit.
2. Persist saved queries and grid formats
3. Terminate error and thread handling.
4. Terminate helps.
5. Destroy the window.

CHAPTER 8

Understanding Fix/Inspect Forms

This chapter discusses fix/inspect forms.

Fix/Inspect Forms

A fix/inspect form is used to update and insert single database records. This form type can also be used to display static information to the user, as well as prompt the user for information. Sign-on screens, for example, prompt the user to enter sign-on information.

Fix/Inspect Events

These events can occur on the fix/inspect form during runtime:

- **Dialog is Initialized**
- **Post Dialog is Initialized**
- **Clear Screen Before Add**
- **Add Record to DB - Before**
- **Add Record to DB - After**
- **Update Record to DB - Before**
- **Update Record to DB - After**
- **Post Commit**
- **End Dialog**
- **XAPI Subscribe Event**

Fix/Inspect Runtime Processing

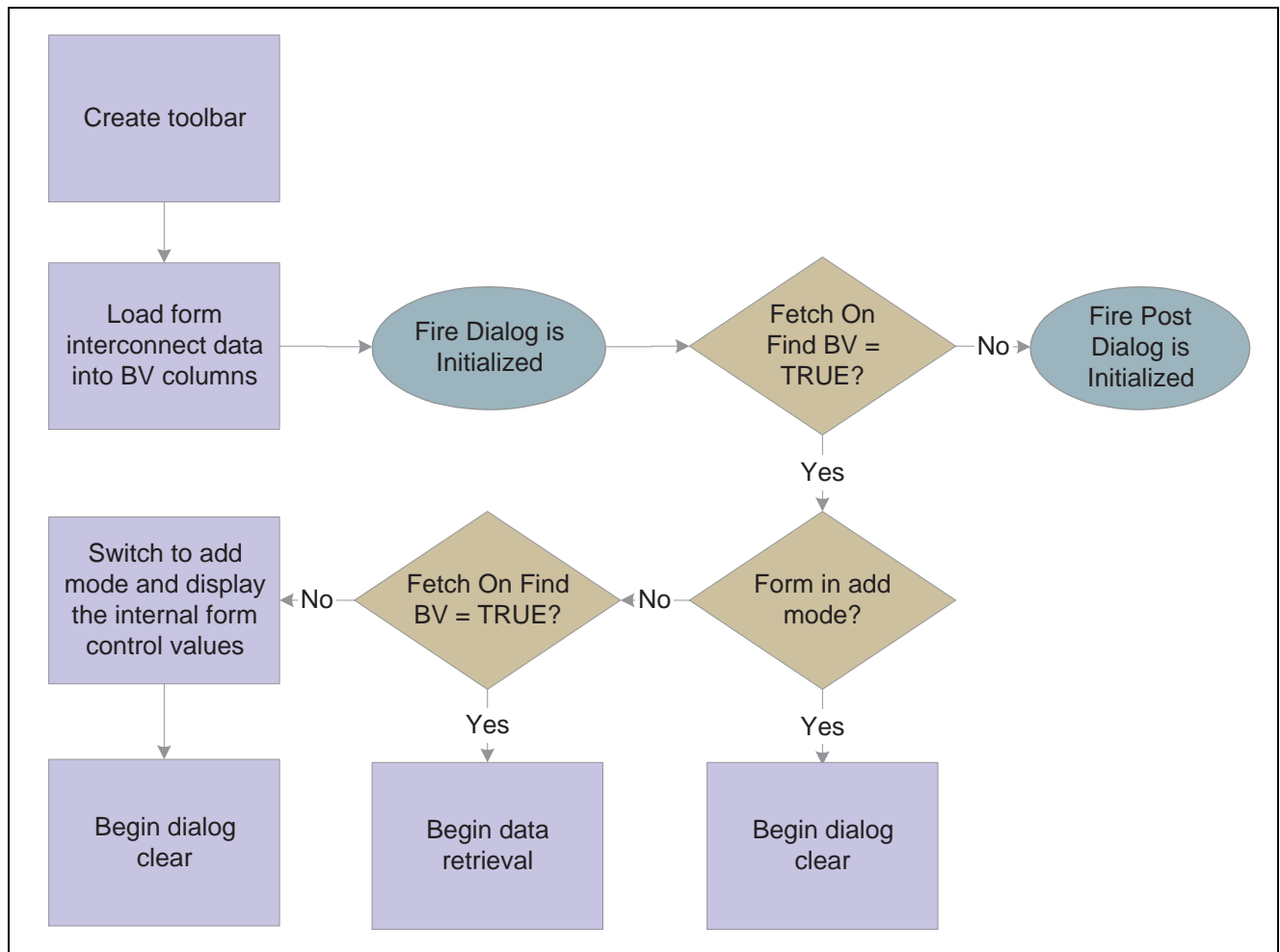
This section discusses how runtime processes fix/inspect forms.

Dialog Initialization

When a fix/inspect form is called, runtime initializes these items in this order:

1. Thread handling
2. Error handling process
3. Business view columns (BC)
4. Form controls (FC)
5. Static text
6. Helps
7. Event rules structures

This flowchart illustrates the tasks that runtime performs after initializing these objects to complete dialog initialization:



Fix/Inspect dialog initialization

Note. FC and BC are sharing internal memory, so copying into the BV is effectively copying into the internal FC memory locations.

Dialog Clear

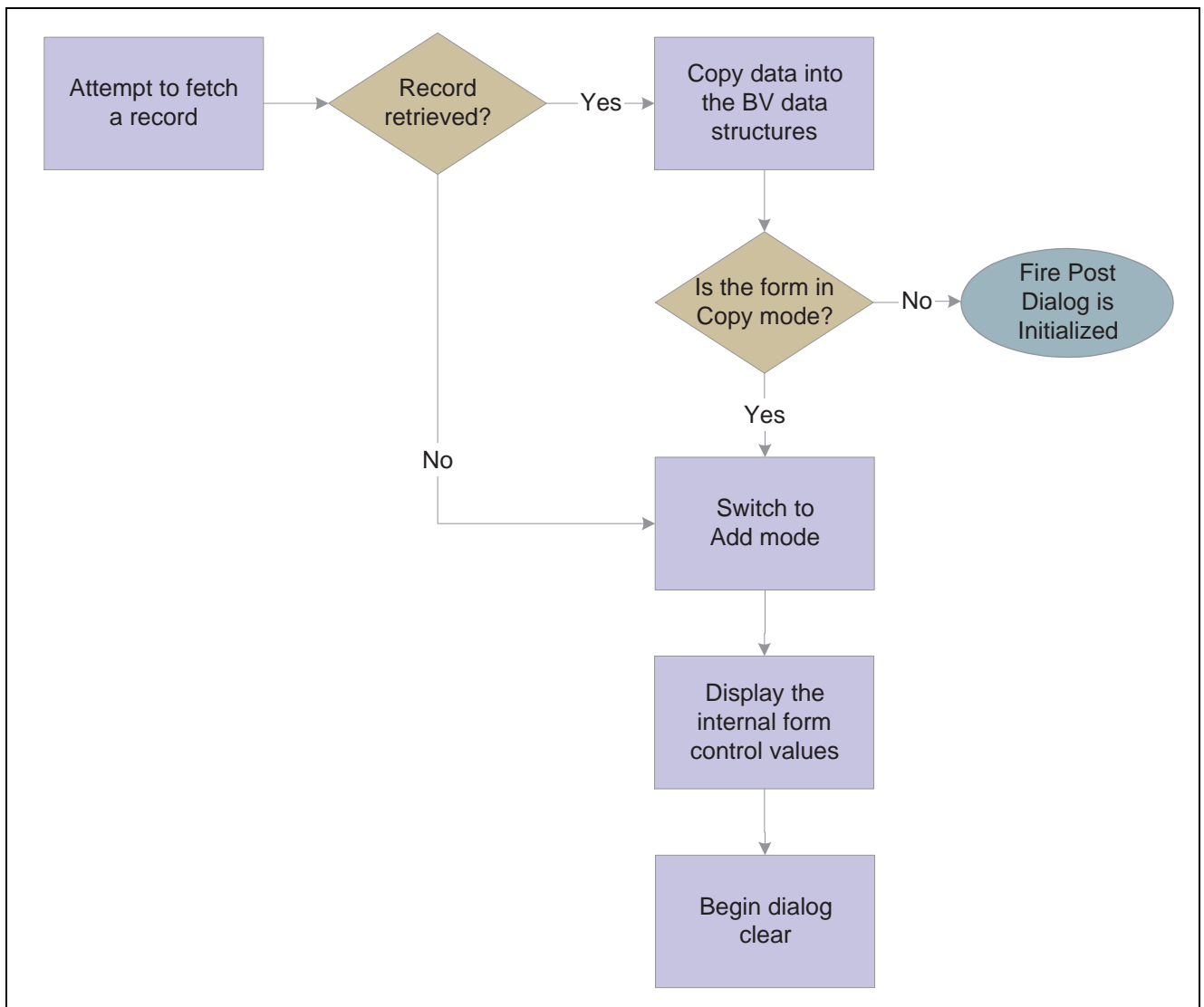
This list describes how runtime clears the form in preparation to display retrieved data:

1. If the form was called in Copy mode, clear the key (primary index) controls for which the Do not clear after add option was enabled.
2. If the form was not called in Copy mode, clear all FCs for which the Do not clear after add option has been disabled.
3. Fire the **Clear Screen Before Add** event.
4. Fire the **Post Dialog is Initialized** event.

Data Retrieval

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

This flowchart illustrates the tasks that runtime performs to retrieve and display data on the fix/inspect form:

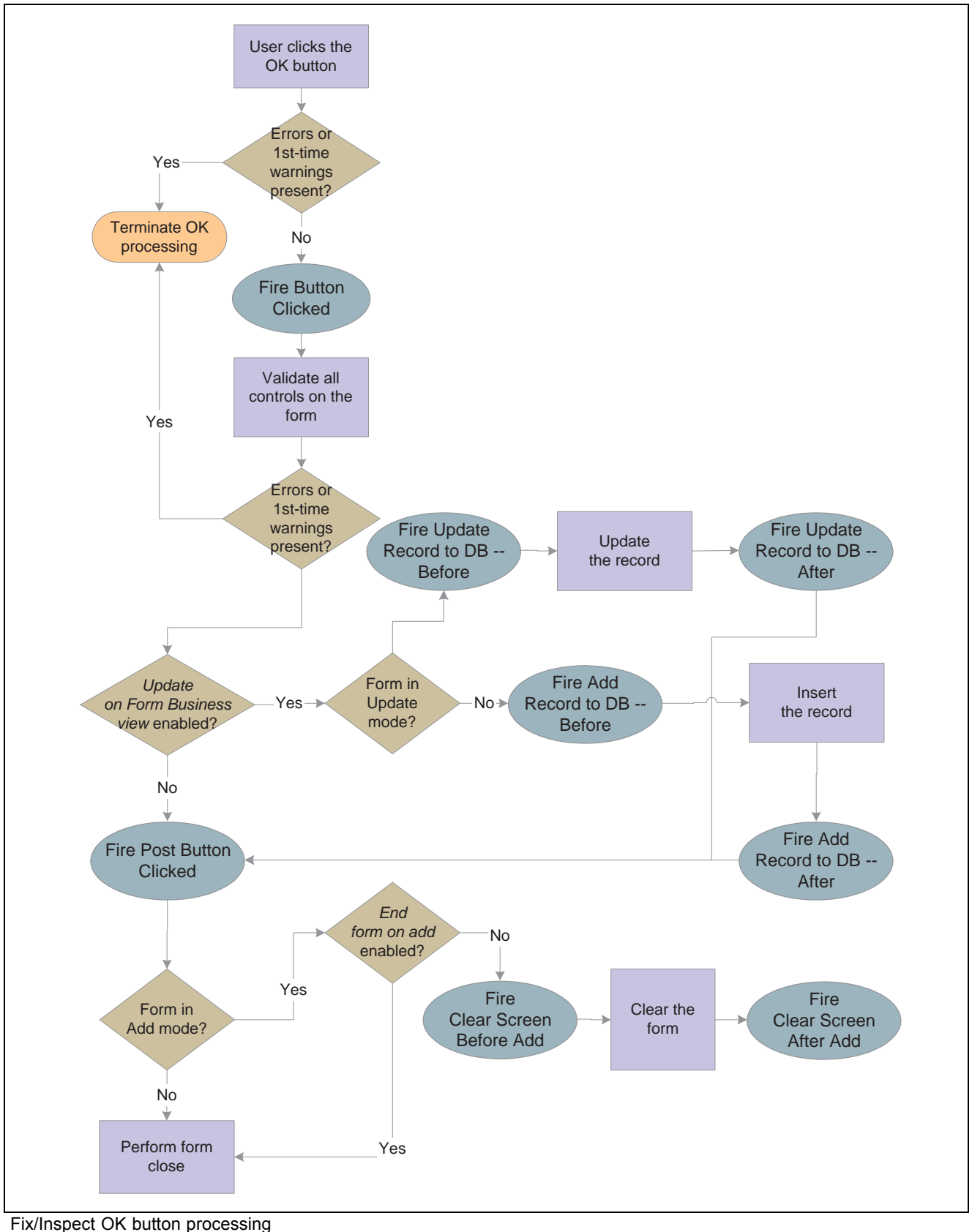


Fix/Inspect data retrieval

OK Button

The OK button is a standard button on fix/inspect forms that appears by default. It causes runtime to validate the information on the form and update or add it to the database through JDEKRNL function calls.

This flowchart illustrates the tasks that constitute runtime processing for the OK button:



Cancel Button

The Cancel button is a standard button on fix/inspect forms that appears by default. When the user clicks it, runtime fires the **Button Clicked** and **Post Button Clicked** events in immediate succession. If no errors occur, runtime attempts to close all of the modeless child forms, if any exist. If any of these child forms cannot be closed, the Cancel button process is terminated. Otherwise, runtime fires the **End Dialog** event and initiates the dialog close process.

Dialog Close

Fix/Inspect can be closed either by the user (typically by clicking the OK or Cancel buttons) or by the system. After performing any control-level close processing that might need to occur, runtime closes the form. If the event has not already occurred, runtime fires the **End Dialog** event. Then it performs these tasks in this order:

1. Load form interconnect data from BC for database commit.
2. Terminate error and thread handling.
3. Terminate helps.
4. Free all form structures.
5. Destroy the window.

CHAPTER 9

Understanding Header Detail Forms

This chapter discusses header detail forms.

Header Detail Forms

A header detail form is used when a relationship exists between the information in the *header* (the fields above the grid) and the information in the *detail area* (the grid itself). The header portion uses one business view (BV), and the detail portion of the form can use another BV. This form type (as well as the headerless detail form type) is referred to as a *transaction form*.

Header Detail Design-Time Considerations

These property values are particularly significant in the design of the header detail form:

- Fetch on Form Businessview
- Fetch on Grid Businessview
- Update on Form Businessview
- Update on Grid Businessview

The fetch properties are important because of their affect in data display during runtime. You must select Fetch on Form Businessview to populate the controls in the header portion of the form. Likewise, you must select Fetch on Grid Businessview to populate the grid.

Similarly, select Update on Form Businessview if you intend to permit users to commit their modifications to records in the header controls to the database. Select Update on Grid Businessview to enable the commitment of modified grid records.

Header Detail Events

These events can occur on the header detail form during runtime:

- **Dialog is Initialized**
- **Post Dialog is Initialized**
- **Grid Record is Fetched**

- **Last Grid Record Has Been Read**
- **Clear Screen Before Add**
- **Clear Screen After Add**
- **Write Grid Line-Before**
- **Write Grid Line-After**
- **Add Record to DB - Before**
- **Add Record to DB - After**
- **Update Record to DB - Before**
- **Update Record to DB - After**
- **Post Commit**
- **End Dialog**
- **XAPI Subscribe Event**

Header Detail Runtime Processing

This section describes how runtime processes header detail forms. This section discusses form-level runtime processing only. Much of the runtime processing for the header detail “form” actually occurs on the level of the grid control, however.

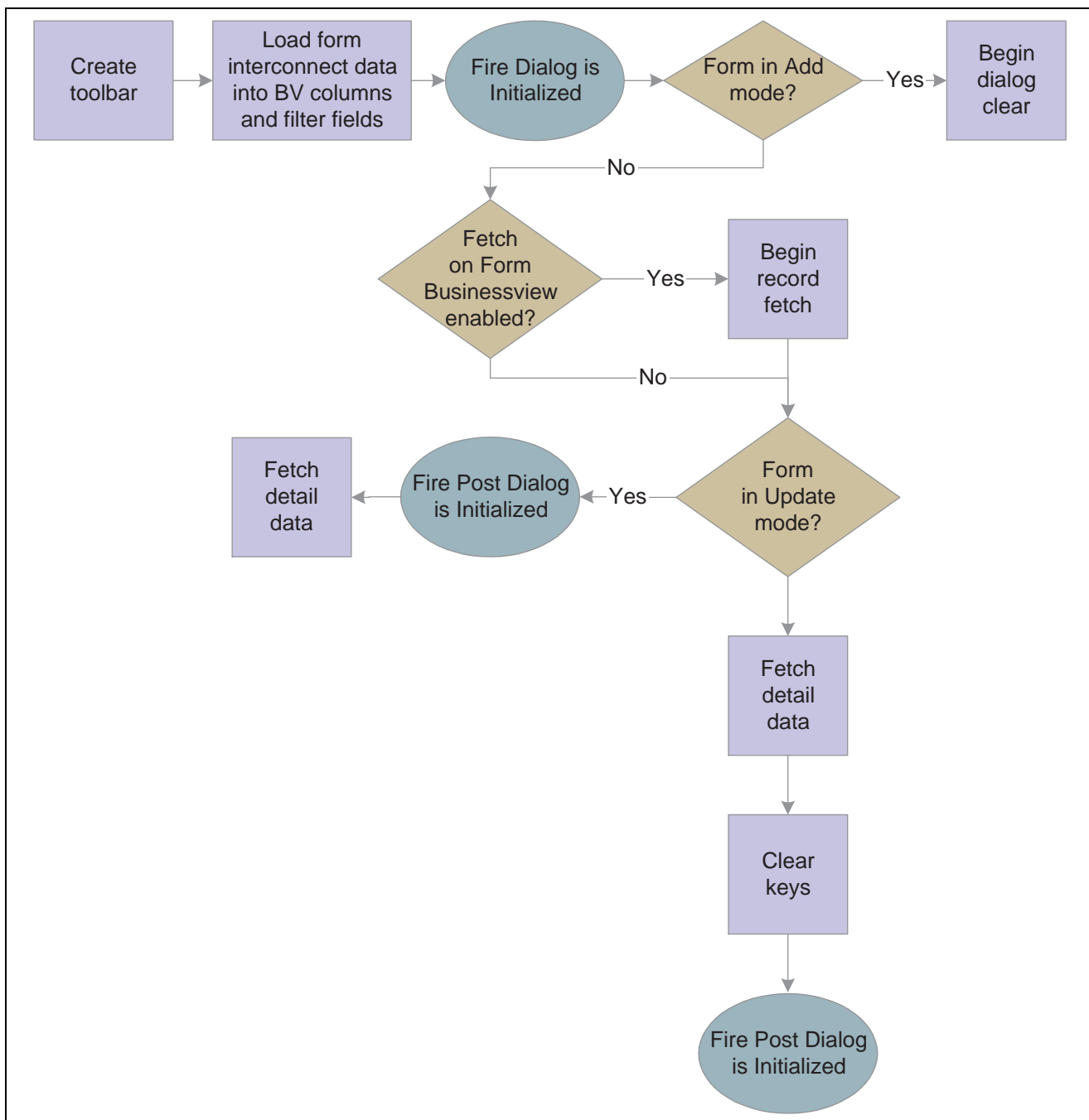
See [Chapter 21, “Understanding Grid Controls,” Grid Control Runtime Processing, page 149](#).

Dialog Initialization

When a header detail form is called, runtime initializes these items in this order:

1. Thread handling
2. Error handling process
3. BV columns
4. Form controls
5. Grid fields
6. Static text
7. Helps
8. Event rules (ER) structures

This flowchart illustrates the tasks that runtime performs after initializing these objects to complete dialog initialization:



Header detail form initialization

Dialog Clear

This list describes how runtime clears the form in preparation to display retrieved data:

1. If the form was called in Copy mode, clear the key (primary index) controls for which the Do not clear after add option was enabled.
2. If the form was not called in Copy mode, clear all form controls for which the Do not clear after add option has been disabled.
3. Fire the **Clear Screen Before Add** event.

4. Fire the **Post Dialog is Initialized** event.

Data Retrieval

After dialog initialization, runtime populates the form only if it is in Update mode. Then, it seeks to populate the header if the Fetch on Form Businessview option is enabled. If enabled, runtime fetches the header record and populates the controls in the header.

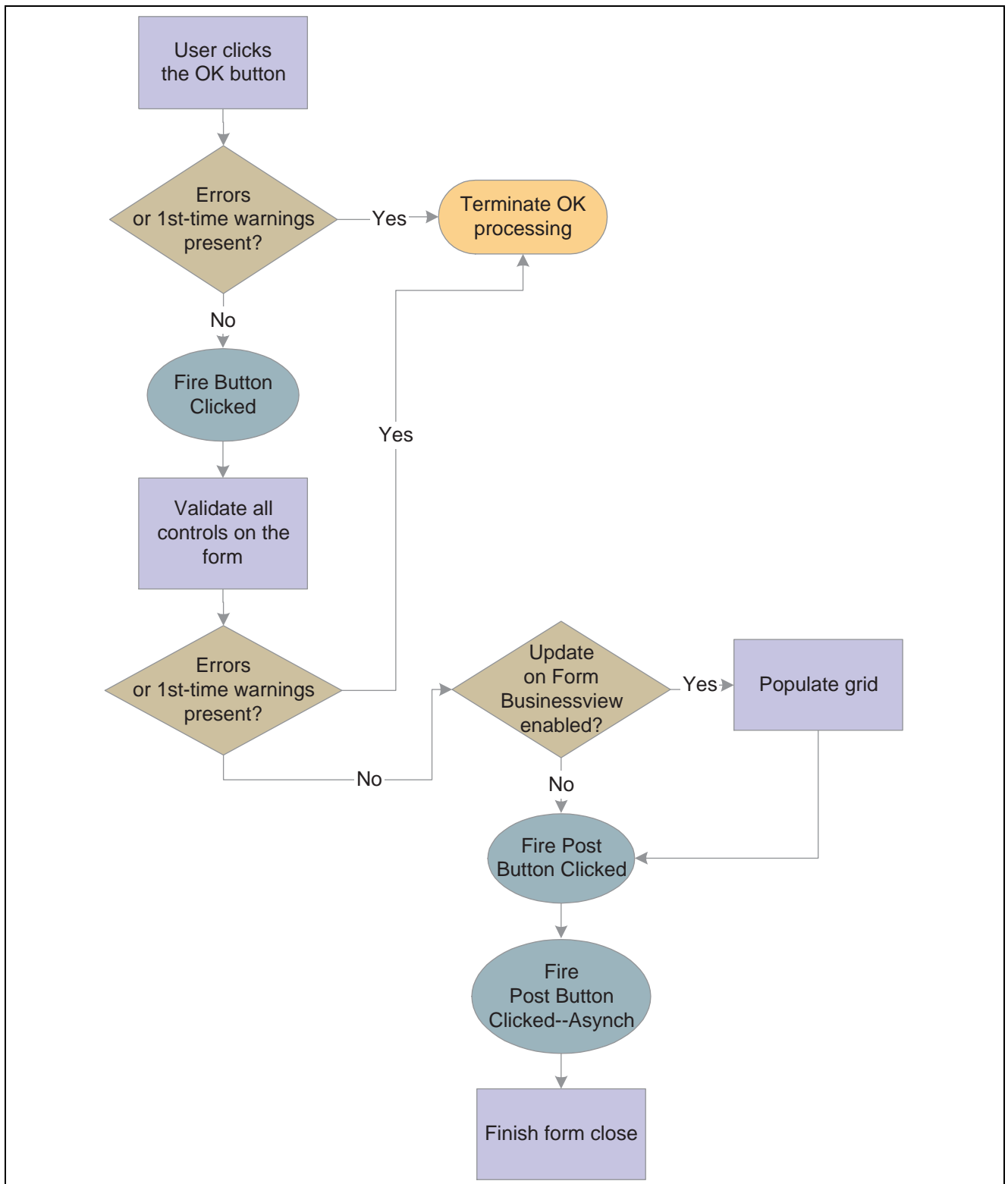
Next, runtime performs a fetch and populates the grid if the Automatically Find on Entry and the Fetch on Grid Businessview options are both enabled. The fetch and populate process used is standard for grid controls.

See [Chapter 21, “Understanding Grid Controls,” How Runtime Processes the Grid Control, page 149.](#)

OK Button

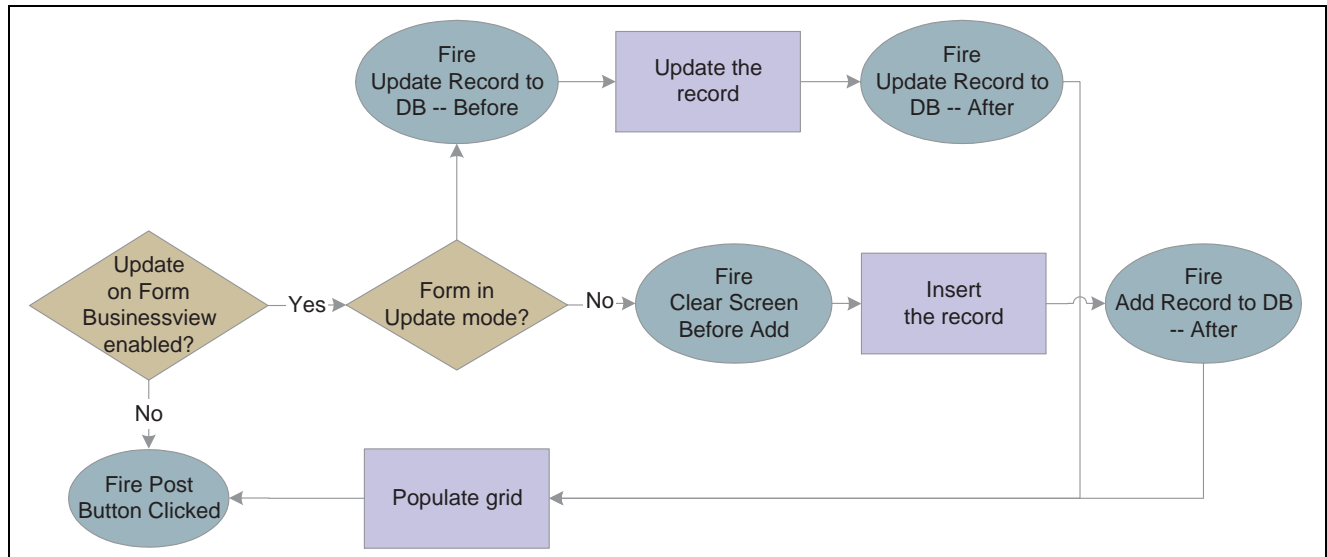
OK is a standard button on header detail forms that appears by default. It causes runtime to validate the information on the form and update or add it to the database through JDEKRNL function calls.

This flowchart illustrates the initial tasks that constitute runtime processing for the OK button:



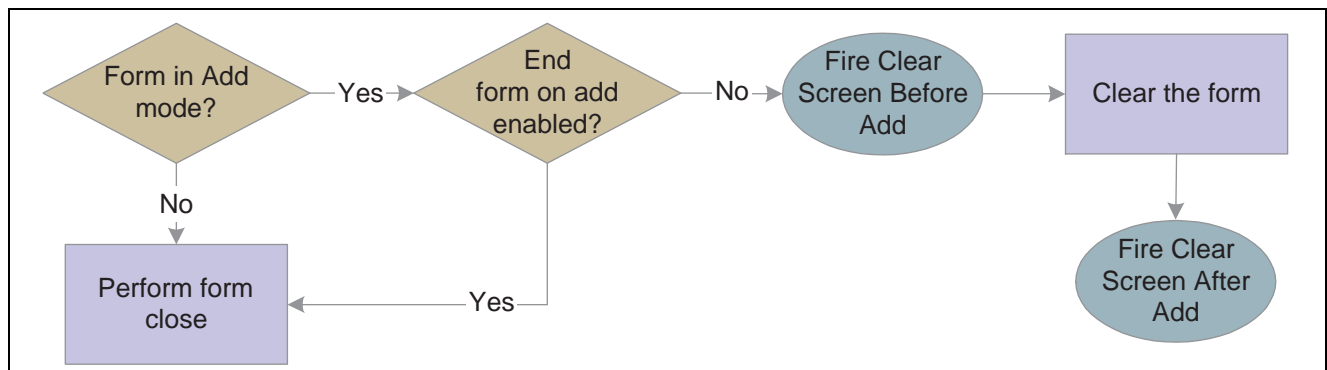
Header detail OK button processing, part 1 of 3

This flowchart expands on how runtime populates the grid during OK button processing:



Header detail OK button processing, part 2 of 3

This flowchart illustrates how runtime completes the form close process during OK button processing:



Header detail OK button processing, part 3 of 3

Cancel Button

The Cancel button is a standard button on header detail forms that appears by default. When the user clicks it, runtime fires the **Button Clicked** and **Post Button Clicked** events in immediate succession. If no errors occur, runtime cancels any media objects that might be open. Also, if a manual transaction is in process, runtime attempts to cancel it as well. Then, runtime fires the **End Dialog** event and initiates the dialog close process.

Dialog Close

Header detail can be closed either by the user (typically by clicking the OK or Cancel buttons) or by the system. After performing any control-level close processing that might need to occur, runtime closes the form. If the event has not already occurred, runtime fires the **End Dialog** event. Then it performs these tasks in this order:

1. Load form interconnect data from BV columns for database commit.
2. Terminate error and thread handling.
3. Terminate helps.
4. Free all form structures.
5. Destroy the window.

CHAPTER 10

Understanding Headerless Detail Forms

This chapter discusses headerless detail forms.

Headerless Detail Forms

A headerless detail form is used to update and enter records that have information that is common to all records in a selected group. This form type (as well as the header detail form type) is referred to as a *transaction form*.

Headerless Detail Design-Time Considerations

These property values are particularly significant in the design of the headerless detail form:

- Fetch on Grid Businessview
- Update on Grid Businessview

The fetch property is important because of its affect in data display during runtime. You must enable Fetch on Grid Businessview to populate the grid. Similarly, enable Update on Grid Businessview to enable the commitment of modified grid records.

Headerless Detail Events

These events can occur on the headerless detail form during runtime:

- **Dialog is Initialized**
- **Post Dialog is Initialized**
- **Grid Record is Fetched**
- **Last Grid Record Has Been Read**
- **Clear Screen Before Add**
- **Clear Screen After Add**
- **Write Grid Line-Before**
- **Write Grid Line-After**
- **Post Commit**

- **End Dialog**
- **XAPI Subscribe Event**

Headerless Detail Runtime Processing

This section describes how runtime processes headerless detail forms. This section discusses form-level runtime processing only. Much of the runtime processing for the headerless detail “form” actually occurs on the level of the grid control, however.

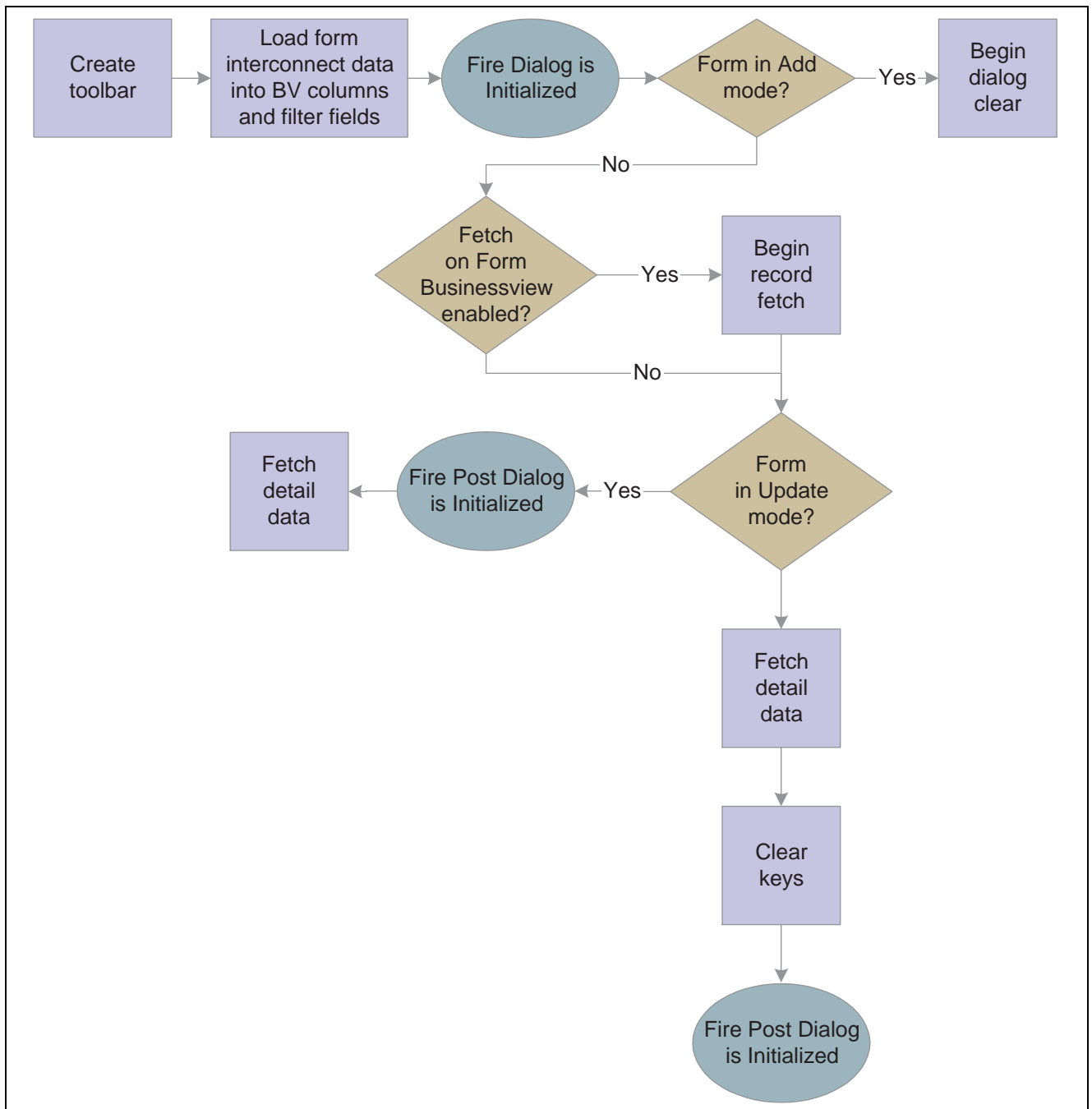
See [Chapter 21, “Understanding Grid Controls,” Grid Control Runtime Processing, page 149](#).

Dialog Initialization

When a headerless detail form is called, runtime initializes these items in this order:

1. Thread handling
2. Error handling process
3. Business view columns (BCs)
4. Form controls (FCs)
5. Grid fields
6. Static text
7. Helps
8. Event rules (ER) structures

This flowchart illustrates the tasks that runtime performs after initializing these objects to complete dialog initialization:



Headerless detail dialog initialization

Dialog Clear

This list describes how runtime clears the form in preparation to display retrieved data:

1. If the form was called in Copy mode, clear the key (primary index) controls for which the Do not clear after add option was enabled.
2. If the form was not called in Copy mode, clear all FCs for which the Do not clear after add option has been disabled.
3. Fire the **Clear Screen Before Add** event.

4. Fire the **Post Dialog is Initialized** event.

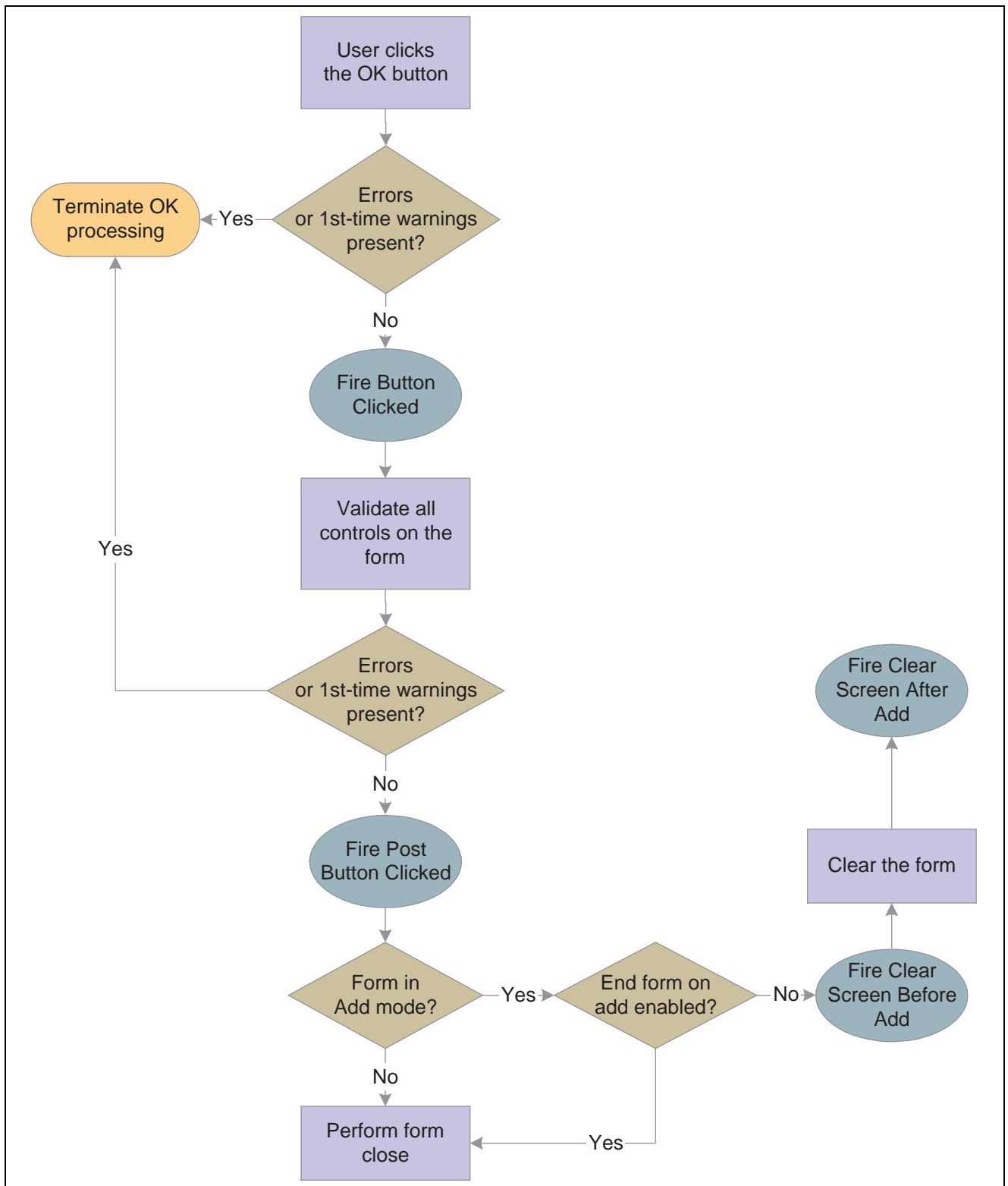
Find Button

The Find button is a standard button on headerless detail forms that appears by default. When the user clicks it, runtime fires the **Button Clicked** event. If no errors exist in the filter fields, runtime performs data selection and sequencing for the grid control. After reloading the grid with the fetched data, runtime fires the **Post Button Clicked** event.

OK Button

OK is a standard button on headerless detail forms that appears by default. It causes runtime to validate the information on the form and update or add it to the database through JDEKRNL function calls.

This flowchart illustrates the tasks that constitute runtime processing for the OK button:



Headerless detail OK button processing

Delete Button

The Delete button is a standard button on headerless detail forms that appears by default. The actual delete from the database does not occur at this point. Runtime verifies the intention to delete when the user clicks the Delete button, and then commits the deletion when the user clicks the OK button. Consequently, if the user clicks the Cancel button, the records are not purged from the database.

Cancel Button

The Cancel button is a standard button on headerless detail forms that appears by default. When the user clicks it, runtime fires the **Button Clicked** and **Post Button Clicked** events in immediate succession. If no errors occur, runtime cancels any media objects that might be open. Also, if a manual transaction is in process, runtime attempts to cancel it as well. Then, runtime fires the **End Dialog** event and initiates the dialog close process.

Dialog Close

Headerless detail can be closed either by the user (typically by clicking the OK or Cancel buttons) or by the system. After performing any control-level close processing that might need to occur, runtime closes the form. If the event has not already occurred, runtime fires the **End Dialog** event. Then it performs these tasks in this order:

1. Load form interconnect data from BC for database commit.
2. Terminate error and thread handling.
3. Terminate helps.
4. Free all form structures.
5. Destroy the window.

CHAPTER 11

Understanding Message Forms

This chapter discusses message forms.

Message Forms

The message form type is a form that appears as a secondary window to inform the user of something or to ask a question. It parallels the behavior of a Windows message box. The form does not have a tool bar or a status bar and can only contain static text and buttons.

Message forms permit only limited use of processing option (PO) values and business functions. Therefore, do not use this form type for complex logic.

Message Form at Design-Time Considerations

Message forms are unique among EnterpriseOne forms in that they include a default push button control on the form. You can configure this button to be OK, Cancel, Yes, or No. You can also make it of type Other and equip it with special functions of your own. Because they have no tool bar, message forms do not have the standard menu buttons available to the other forms. Hence, you must provide all functions with manually-added controls.

Understanding Message Form Events

These events can occur on the message form during runtime:

- **Dialog is Initialized**
- **XAPI Subscribe Event**

Message Form Runtime Processing

This section describes how the runtime engine processes the message form.

Dialog Initialization

When a message form is called, runtime initializes these items in this order:

1. Form controls
2. Static text
3. Helps
4. Event rules (ER) structures

Then, it fires **Dialog is Initialized**.

Dialog Close

Message forms close when the user clicks the default button. Then runtime performs these tasks in this order:

1. Terminate helps.
2. Free all form structures.
3. Destroy the window.

CHAPTER 12

Understanding Parent/Child Browse Forms

This chapter discusses parent/child browse forms.

Parent/Child Browse Forms

Similar to find/browse forms, parent/child browse forms are used to query business views (BVs) and select records from BVs for operations. However, instead of a default grid control, parent/child browse forms contain a default parent child control instead.

Parent/Child Browse Events

These events can occur on the parent/child browse form during runtime:

- **Dialog is Initialized**
- **Post Dialog is Initialized**
- **Grid Record is Fetched**
- **Write Grid Line-Before**
- **Write Grid Line-After**
- **Last Grid Record Has Been Read**
- **XAPI Subscribe Event**
- **End Dialog**

Parent/Child Browse Runtime Processing

This section discusses how runtime processes parent/child browse forms.

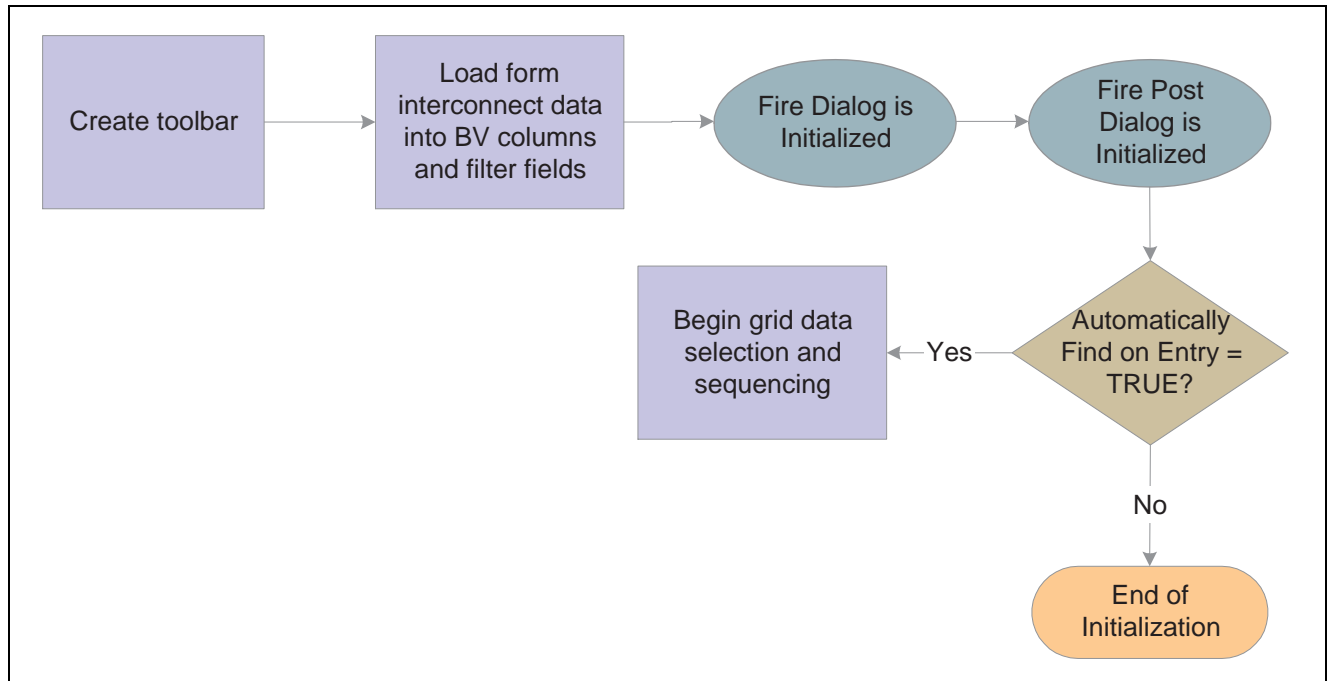
Dialog Initialization

When a parent/child browse form is called, runtime initializes these items in this order:

1. Thread handling
2. Error handling process

3. Business view columns (BCs)
4. Form controls (FCs)
5. Grid fields
6. Static text
7. Helps
8. Event rules (ER) structures

This flowchart illustrates the tasks that runtime performs after initializing these objects to complete dialog initialization:



Parent/Child browse dialog initialization

Data selection and sequencing occurs at the control level and ultimately leads to the population of the parent/child control, provided runtime encounters no errors.

Find Button

The Find button is a standard button on parent/child forms that appears by default. When the user clicks it, runtime fires the **Button Clicked** event. If no errors exist in the filter fields, runtime performs data selection and sequencing for the grid and tree controls. After reloading the grid and tree with the fetched data, runtime fires the **Post Button Clicked** event.

Select Button

Select is a standard item that is automatically placed on parent/child browse forms. No default processing exists for Select on parent/child browse forms. It acts as a user-defined item.

Close Button

The Close button is a standard button on parent/child forms that appears by default. When the user clicks it, runtime fires the **Button Clicked** and **Post Button Clicked** events in immediate succession. If no errors occur, runtime attempts to close all of the modeless child forms, if any exist. If any of these child forms cannot be closed, the Close button process is terminated. Otherwise, runtime fires the **End Dialog** event and initiates the dialog close process.

Dialog Close

Parent/Child can be closed either by the user (typically by clicking the Close button) or by the system. After performing any control-level close processing that might need to occur, runtime closes the form. If the event has not already occurred, runtime fires the **End Dialog** event. Then it performs these tasks in this order:

1. Load form interconnect data from BCs for database commit.
2. Terminate error and thread handling.
3. Terminate helps.
4. Free all form structures.
5. Destroy the window.

CHAPTER 13

Using Power Browse Forms

This chapter discusses power browse forms.

Power Browse Forms

Power forms are Web-only application forms that, through the use of the subform control, enable users to view multiple, interrelated views of data, grids, and tab pages on one form and to pass logic between them. The tab pages can have their own business views (BVs), and these BVs can communicate with each other and can update based on data selection and changes that occur in other BVs on the form. In this way, you can simplify navigation tasks for users.

Power forms have these general properties:

- All regular controls except a parent child control can be placed on a power form.
- Multiple tab controls are permitted.
- The maximize grid feature is supported for all grids.
- All power form (and subform) errors and warnings are shown from the error button on the power form.
- Power forms contain vertical and horizontal scroll bars.

EnterpriseOne offers two types of power forms: browse and edit. Use a power browse form to browse data on a form. You cannot use a power browse form to change data on the form. This form is similar to a find/browse form. A power browse form with a BV should always have a grid with at least one column. You can hide the grid.

See Also

[Chapter 7, “Understanding Find/Browse Forms,” page 55](#)

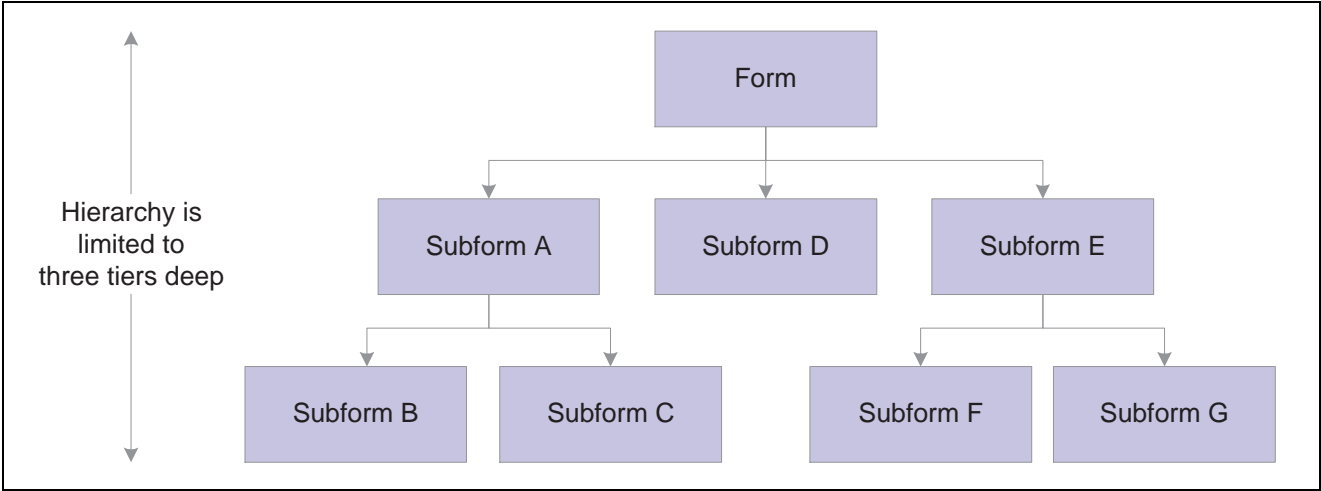
Power Form Hierarchical Structures

Hierarchical relationships govern the processing between containers in the application. The power form enables you to create hierarchies based on BVs using power forms and subform controls. A power form always resides at the top of the hierarchy. It is a parent form, and can have multiple subform children. Subforms also can function as parent forms, and they can have multiple children. All subforms must have at least one parent power form or subform. At most, you can have three levels, including the initial power form, in a power form hierarchy.

Note. Data flow follows the parenting structure in the logical hierarchy.

A subform can communicate only with its parent or its children. Logic from the event rules (ER) on the subform, or its subcontrols, does not flow to or from the other subforms. Related information does not appear in the available objects.

This flowchart illustrates the general hierarchy scheme of power forms:

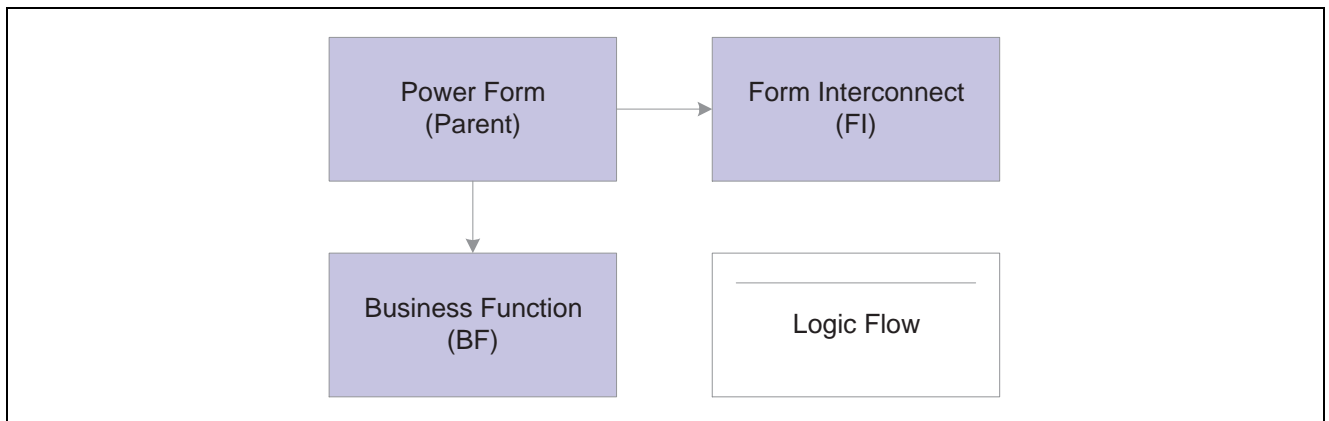


Power form hierarchical scheme example

Object	Passes Logic To and From
Power Form	Subforms A, D, and E
Subform A	Power Form, Subforms B and C
Subforms B and C	Subform A, Power Form
Subform D	Power Form
Subform E	Power Form, Subforms F and G
Subforms F and G	Subform E, Power Form

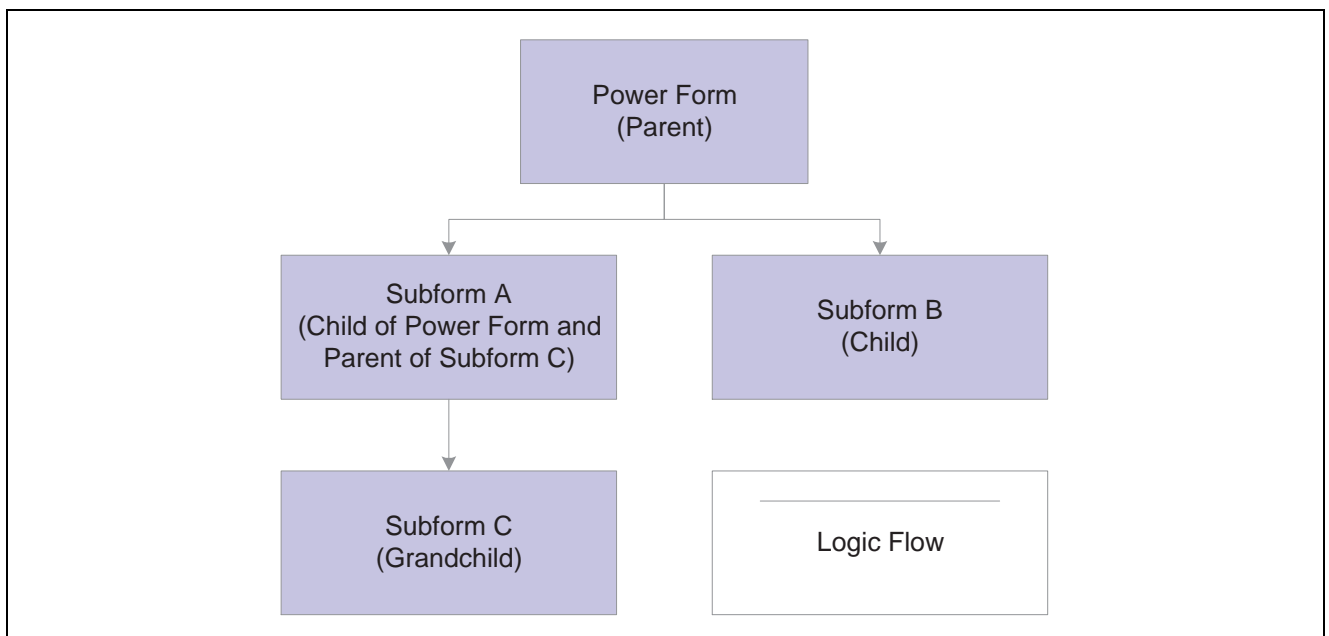
Examples of the Logic Flow of Power Forms

This flowchart illustrates the flow of logic between a power form and a business function, and the power form and a form interconnect:



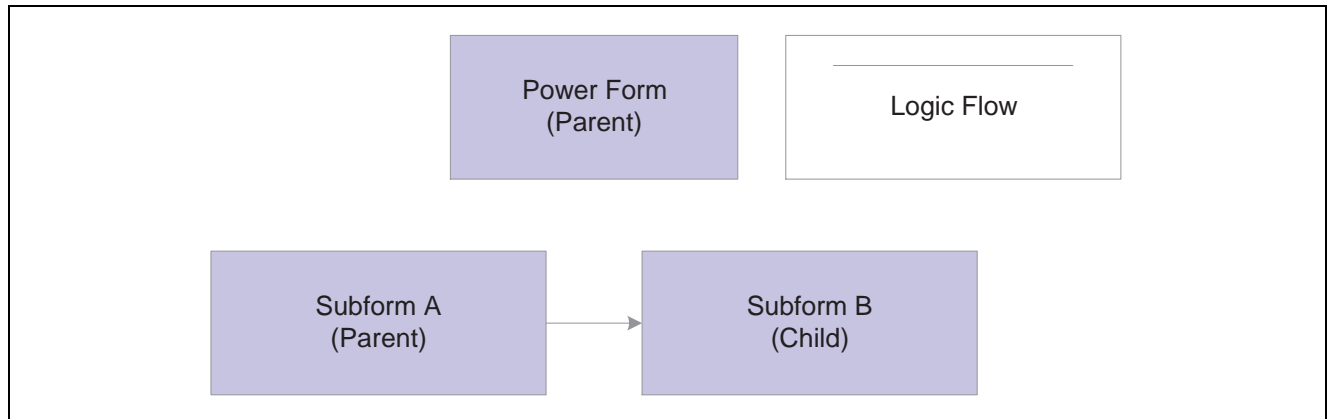
Power form logic flow example 1

This flowchart illustrates the flow of logic between a power form as a parent, two subforms as children (subforms A and B), and a subform as a parent and a child (Subform A):



Power form logic flow example 2

This flowchart illustrates the flow of logic between subform A and subform B. The flow of logic occurs only between subform A (parent) and subform B (child). The subforms do not communicate with the power form:



Power form logic flow example 3

Power Browse Form Design-Time Considerations

These property values are particularly significant in the design of power browse forms:

- Mapping Links
- Transaction

Use the Mapping Links property to identify the data to pass between parent and child. For each data item, specify whether the data is to flow from parent to child, child, to parent, or both directions. You must set up a mapping for every child of the parent. Without the mapping, the power browse form probably will not work correctly.

Power Browse Events

These events can occur on power browse forms during runtime:

- **Dialog is Initialized**
- **Post Dialog is Initialized**
- **Grid Record is Fetched**
- **Write Grid Line-Before**
- **Write Grid Line-After**
- **Last Grid Record Has Been Fetched**
- **Notified by Child**
- **End Dialog**

Power Browse Form Runtime Processing

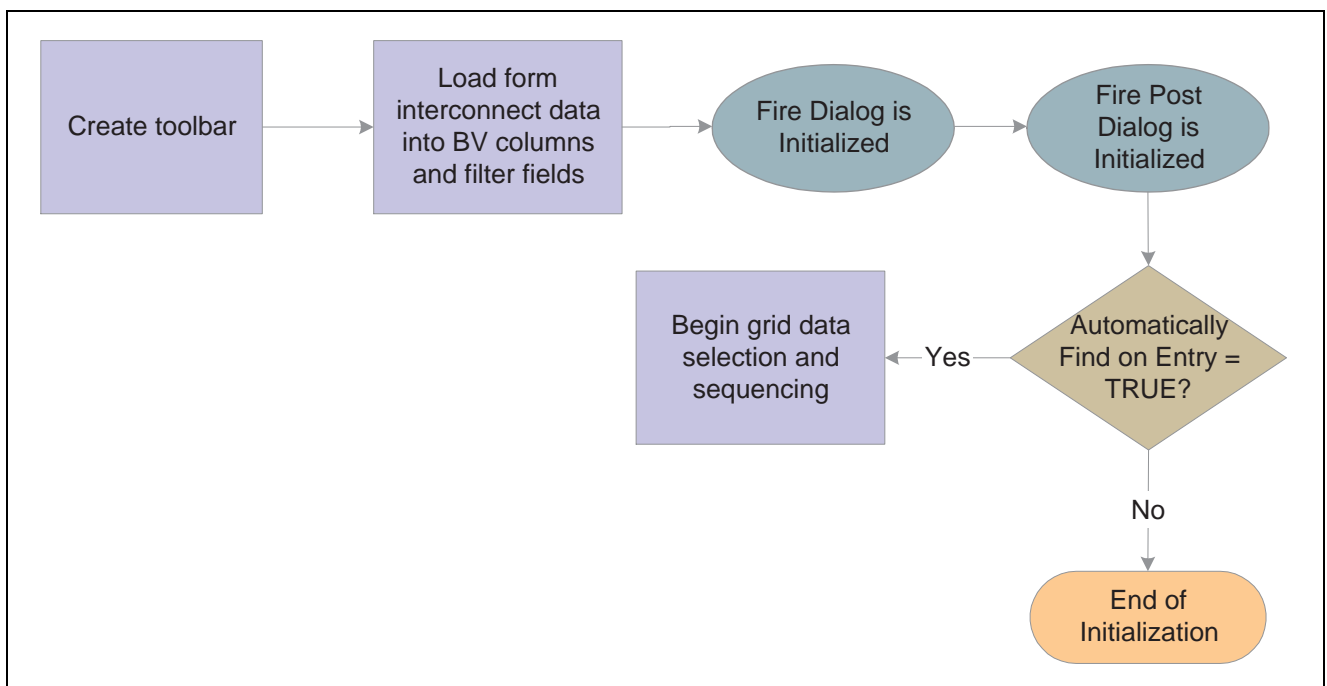
This section describes how runtime processes power browse forms.

Dialog Initialization

When a power browse form is called, runtime initializes these items in this order:

1. Thread handling
2. Error handling process
3. BV columns
4. Form controls (FC)
5. Grid fields
6. Static text
7. Helps
8. ER structures

This flowchart illustrates the tasks that runtime performs after initializing these objects to complete dialog initialization:



Power browse dialog initialization

Grid data selection and sequencing occurs at the control level and ultimately leads to the population of the grid, provided runtime encounters no errors.

See [Chapter 21, “Understanding Grid Controls,” How Runtime Processes the Grid Control, page 149.](#)

The system creates an internal structure that represents the data selection and data sequencing requirements specified by the user. The system then passes this to the database engine to perform the actual database select and sequencing. The data used for selection is based on values from filter fields and query-by-example (QBE) columns. The system holds the data until the data is retrieved.

Find Button

The Find button is a standard button on power browse forms that appears by default. When the user clicks it, runtime fires the **Button Clicked** event. If no errors exist in the filter fields, runtime performs data selection and sequencing for the grid control. After reloading the grid with the fetched data, runtime fires the **Post Button Clicked** event.

Select Button

The Select button is a standard button on power browse forms that appears by default. When the user clicks it, runtime fires the **Button Clicked** event. If no errors occur, runtime writes the values from the selected row to the BC and fires the **Post Button Clicked** event. Then it fires the **End Dialog** event and initiates the dialog close process.

Close Button

The Close button is a standard button on power browse forms that appears by default. When the user clicks it, runtime fires the **Button Clicked** and **Post Button Clicked** events in immediate succession. If no errors occur, runtime attempts to close all of the modeless child forms, if any exist. If any of these child forms cannot be closed, the Close button process is terminated. Otherwise, runtime fires the **End Dialog** event and initiates the dialog close process.

Dialog Close

Power browse can be closed either by the user (typically by clicking the Select or Close buttons) or by the system. After performing any control-level close processing that might need to occur, runtime closes the form. If the event has not already occurred, runtime fires the **End Dialog** event. Then it performs these tasks in this order:

1. Load form interconnect data from BV columns for database commit.
2. Terminate error handling.
3. Terminate helps.
4. Free all form structures.
5. Destroy the window.

Transaction Boundaries for Power Forms and Subforms

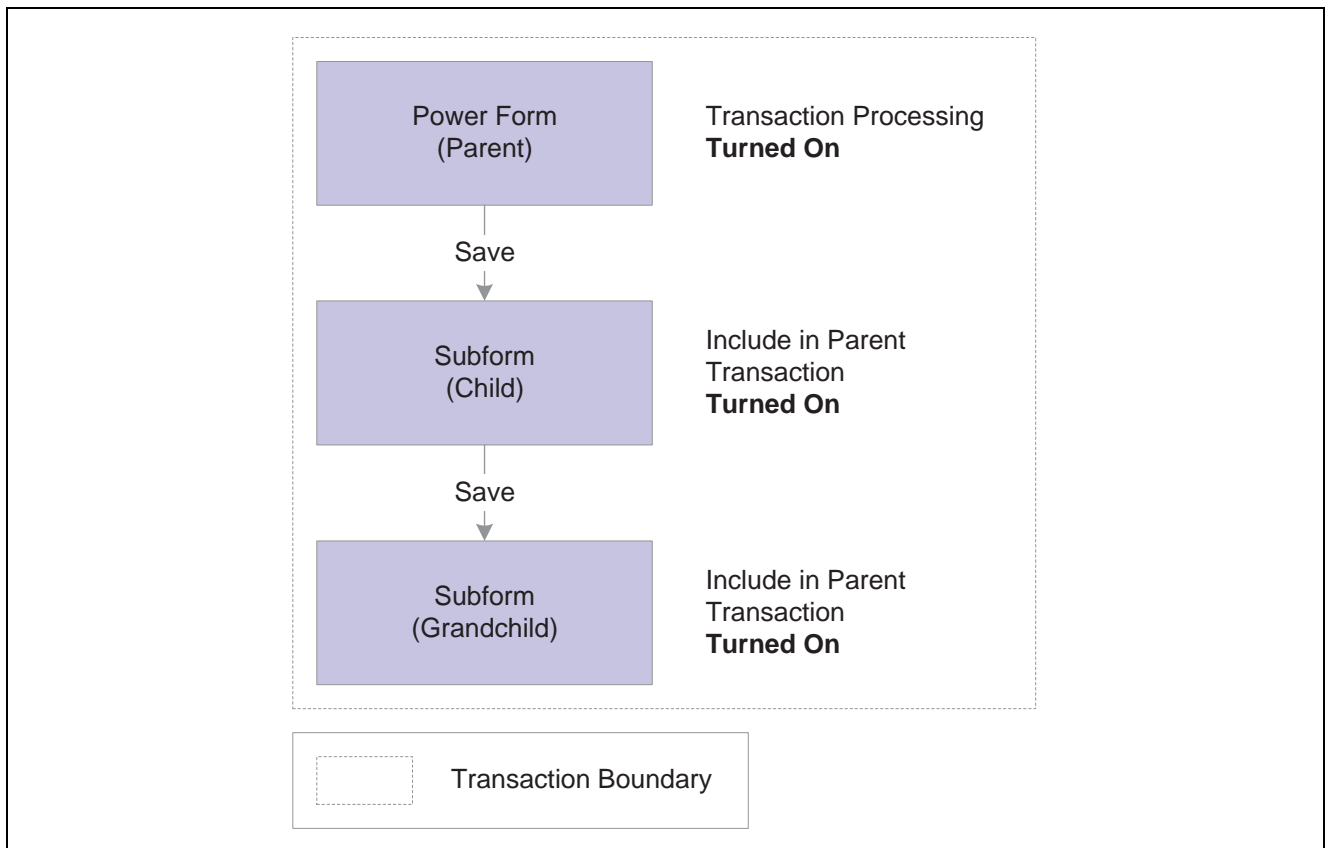
Power forms include the Transaction property, which functions the same as the other form types. By default, it is scoped to OK button processing; however, the scope can be extended to business functions, table I/O, and form interconnects.

Subforms are scoped to Save button processing; however, the scope can be extended to business functions, table I/O, and form interconnects. You can use these transaction processing settings for subforms:

Setting	Result
Transaction Disabled (default)	No transaction processing occurs for this form.

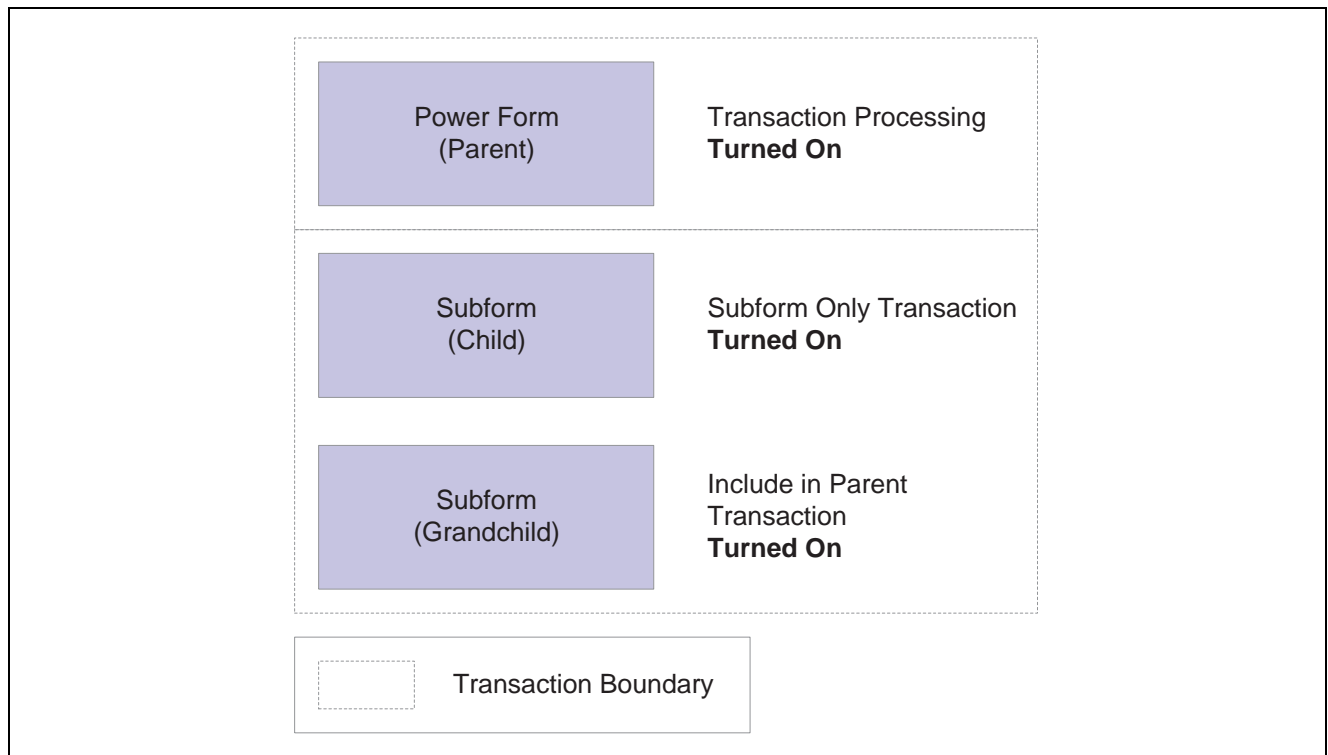
Setting	Result
Include in Parent Transaction	If the form is called within the parent's transaction boundary (OK or Save processing), the transaction will be included in the parent's transaction boundary.
Subform Only Transaction	Save processing for the Subform has its own transaction boundary, which is independent of all boundaries.

This flowchart illustrates how to include all of the subforms in the form save transaction. If you rollback data on any one form, the system rolls back the changes on all of them:



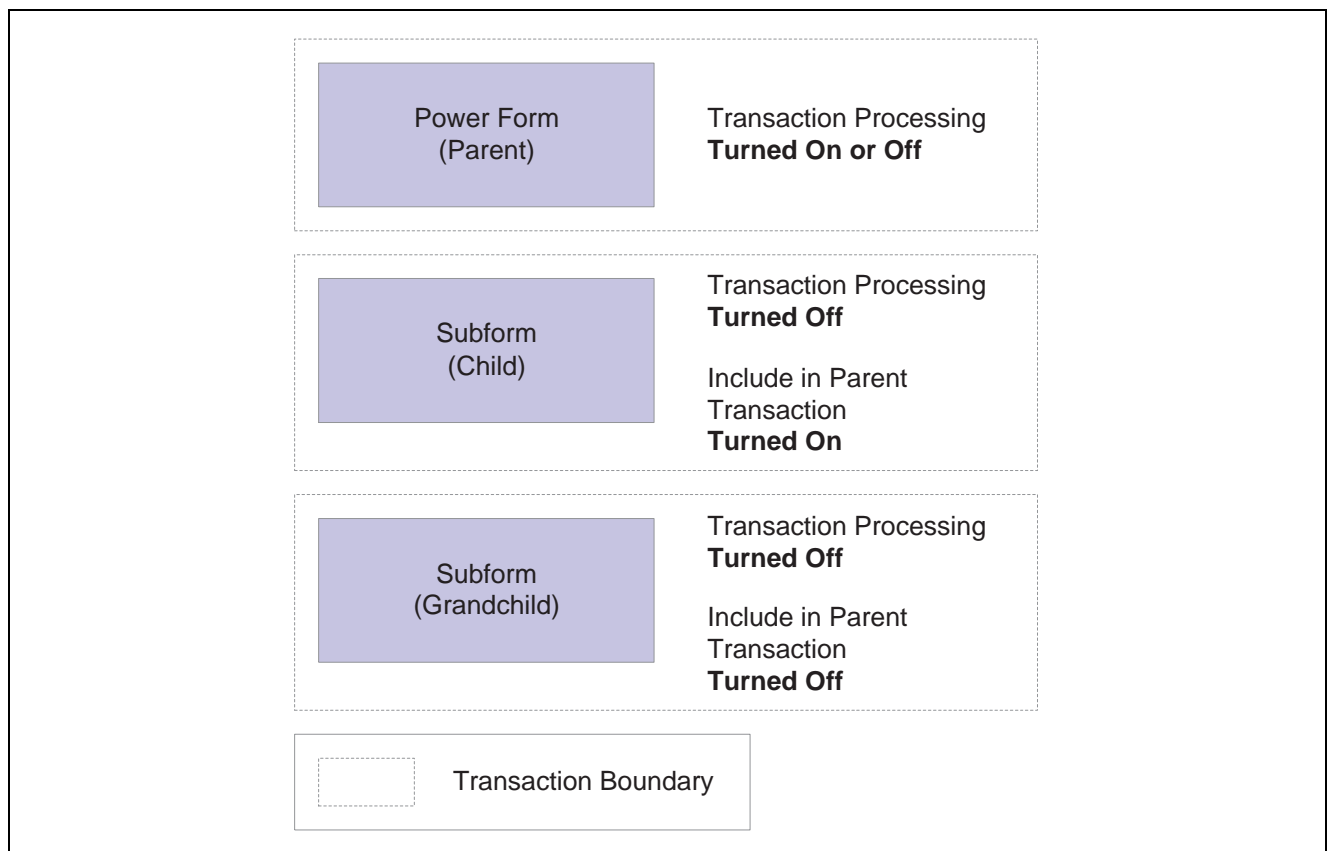
Including parent, child, and grandchild in the transaction boundary

Here, the child is not included in the transaction for the form, which means that a rollback on the parent level will not affect any of the subforms. However, all of the subforms are included in the same transaction, so any rollback on the subforms will affect all of the subforms:



Placing parent in a separate transaction boundary from its child and grandchild

In all these scenarios, transactions are local and will not affect appear on any other form:



Placing parent, child, and grandchild in separate transaction boundaries

CHAPTER 14

Understanding Power Edit Forms

This chapter provides overviews of power edit forms.

Power Edit Forms

Power forms are Web-only application forms that, through the use of the subform control, enable users to view multiple, interrelated views of data, grids, and tab pages on one form and to pass logic between them. The tab pages can have their own business views (BVs), and these BVs can communicate with each other and can update based on data selection and changes that occur in other BVs on the form. In this way, you can simplify navigation tasks for users.

Power forms have these general properties:

- All regular controls except a parent child control can be placed on a power form.
- Multiple tab controls are permitted.
- The maximize grid feature is supported for all grids.
- All power form (and subform) errors and warnings are shown from the error button on the power form.
- Power forms contain vertical and horizontal scroll bars.

EnterpriseOne offers two types of power forms: browse and edit. A power edit form with a grid enables users to update and enter multiple records simultaneously. Similar to a headerless detail form, a power edit form has only one BV.

Power edit forms contain a grid control by default. If you leave the grid on the form, runtime processes the form similarly to how it processes headerless detail forms. However, if you remove the grid control, runtime processes it similarly to how it processes fix/inspect forms.

See Also

[Chapter 8, “Understanding Fix/Inspect Forms,” page 59](#)

[Chapter 10, “Understanding Headerless Detail Forms,” page 71](#)

[Chapter 13, “Using Power Browse Forms,” page 83](#)

Power Edit Form Design-Time Considerations

These property values are particularly significant in the design of the power edit form:

- Mapping Links

- Transaction

Use the Mapping Links property to identify the data to pass between parent and child. For each data item, specify whether the data is to flow from parent to child, child, to parent, or both directions. You must set up a mapping for every child of the parent. Without the mapping, the power edit form probably will not work correctly.

Power Edit Events

These events can occur on the power edit form during runtime if the form contains a grid control:

- **Dialog is Initialized**
- **Post Dialog is Initialized**
- **Grid Record is Fetched**
- **Last Grid Record Has Been Read**
- **Clear Screen Before Add**
- **Clear Screen After Add**
- **Write Grid Line-Before**
- **Write Grid Line-After**
- **Post Commit**
- **Notified by Child**
- **End Dialog**

These events can occur on the power edit form during runtime if the form does not contain a grid control:

- **Dialog is Initialized**
- **Post Dialog is Initialized**
- **Clear Screen Before Add**
- **Clear Screen After Add**
- **Add Record to DB - Before**
- **Add Record to DB - After**
- **Update Record to DB - Before**
- **Update Record to DB - After**
- **Post Commit**
- **Notified by Child**
- **End Dialog**

Power Edit Form Runtime Processing

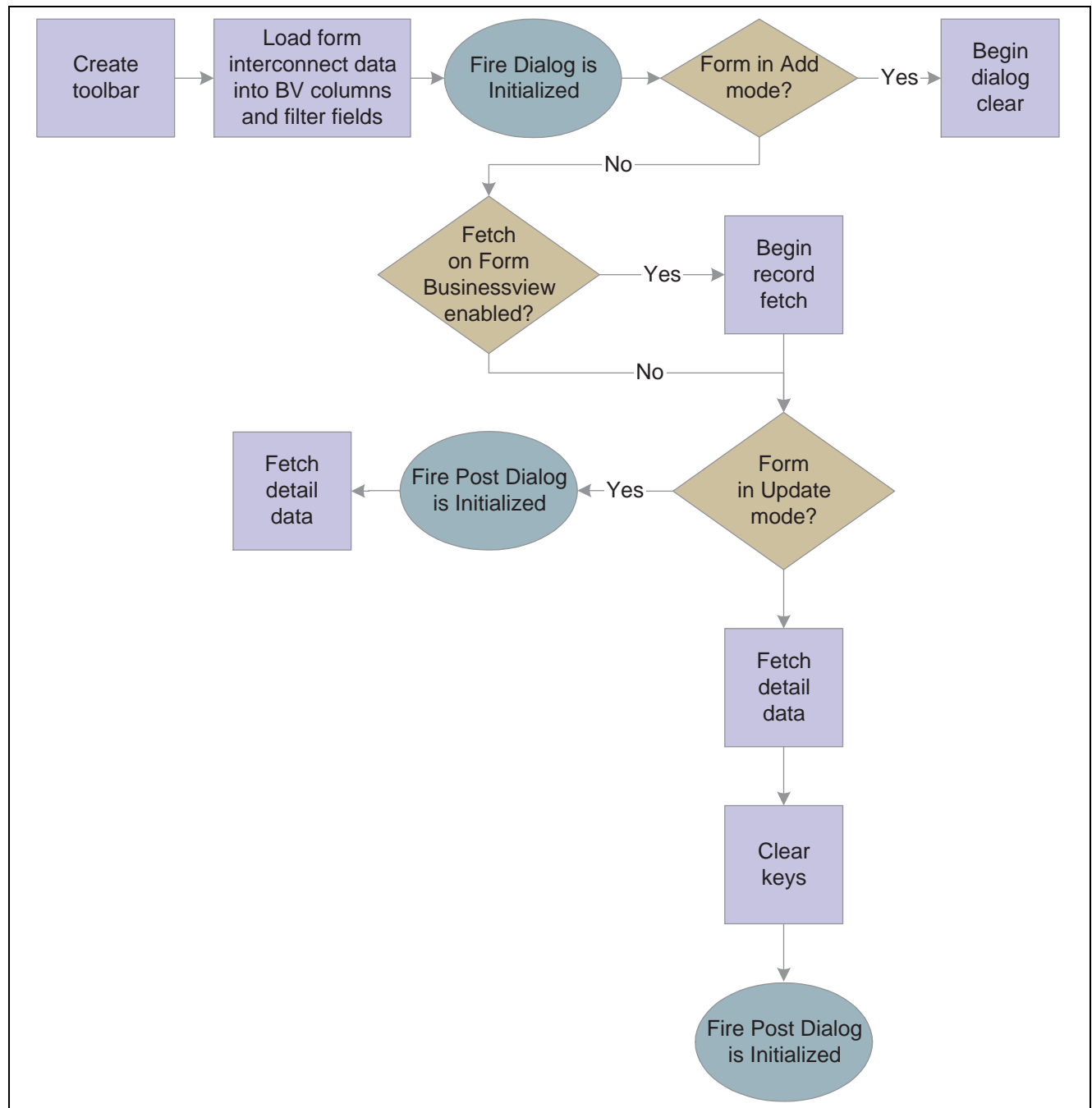
This section discusses how runtime processes power edit forms.

Dialog Initialization

When a power edit form is called, runtime initializes these items in this order:

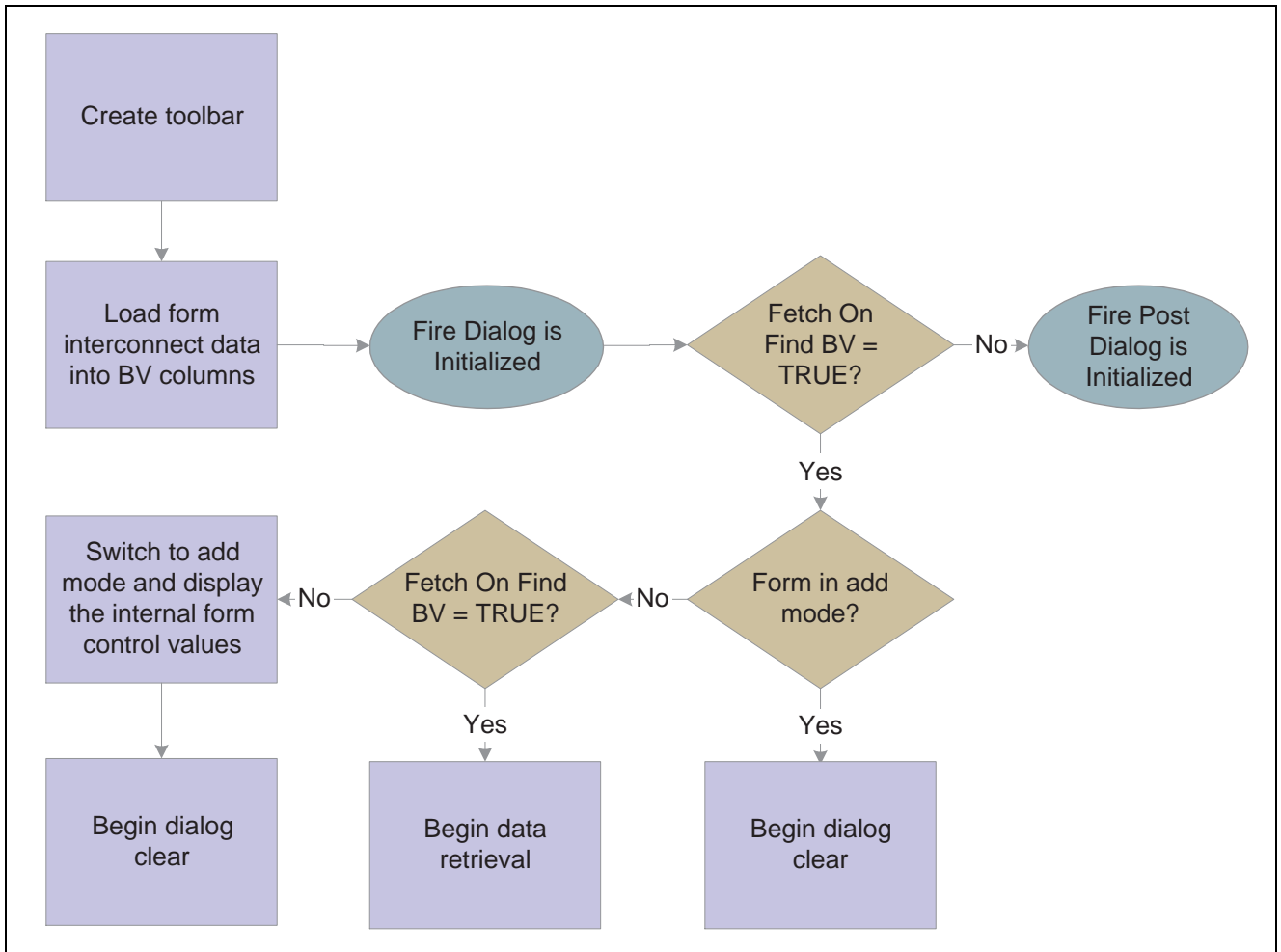
1. Thread handling
2. Error handling process
3. BV columns
4. Form controls (FC)
5. Grid fields
6. Static text
7. Helps
8. Event rules (ER) structures

This flowchart illustrates the tasks that runtime performs after initializing these objects to complete dialog initialization, if the form contains a grid control:



Power edit form with grid control dialog initialization

This flowchart illustrates the tasks that runtime performs after initializing these objects to complete dialog initialization, if the form does not contain a grid control:



Power edit form with no grid control dialog initialization

Dialog Clear

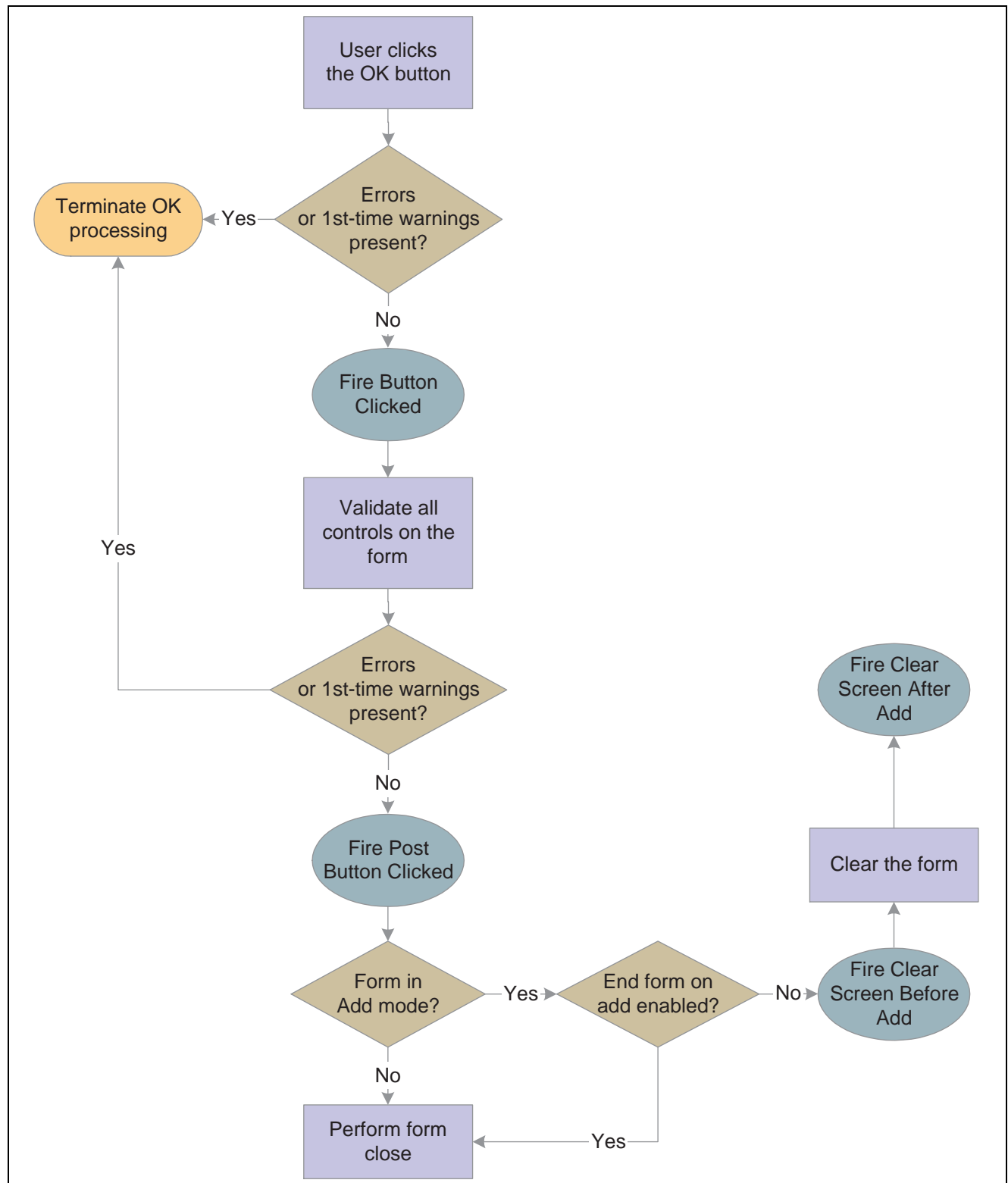
This list describes how runtime clears the form in preparation to display retrieved data:

1. If the form was called in Copy mode, clear the key (primary index) controls for which the Do not clear after add option was selected.
2. If the form was not called in Copy mode, clear all FCs for which the Do not clear after add option has been selected.
3. Fire the **Clear Screen Before Add** event.
4. Fire the **Post Dialog is Initialized** event.

OK Button

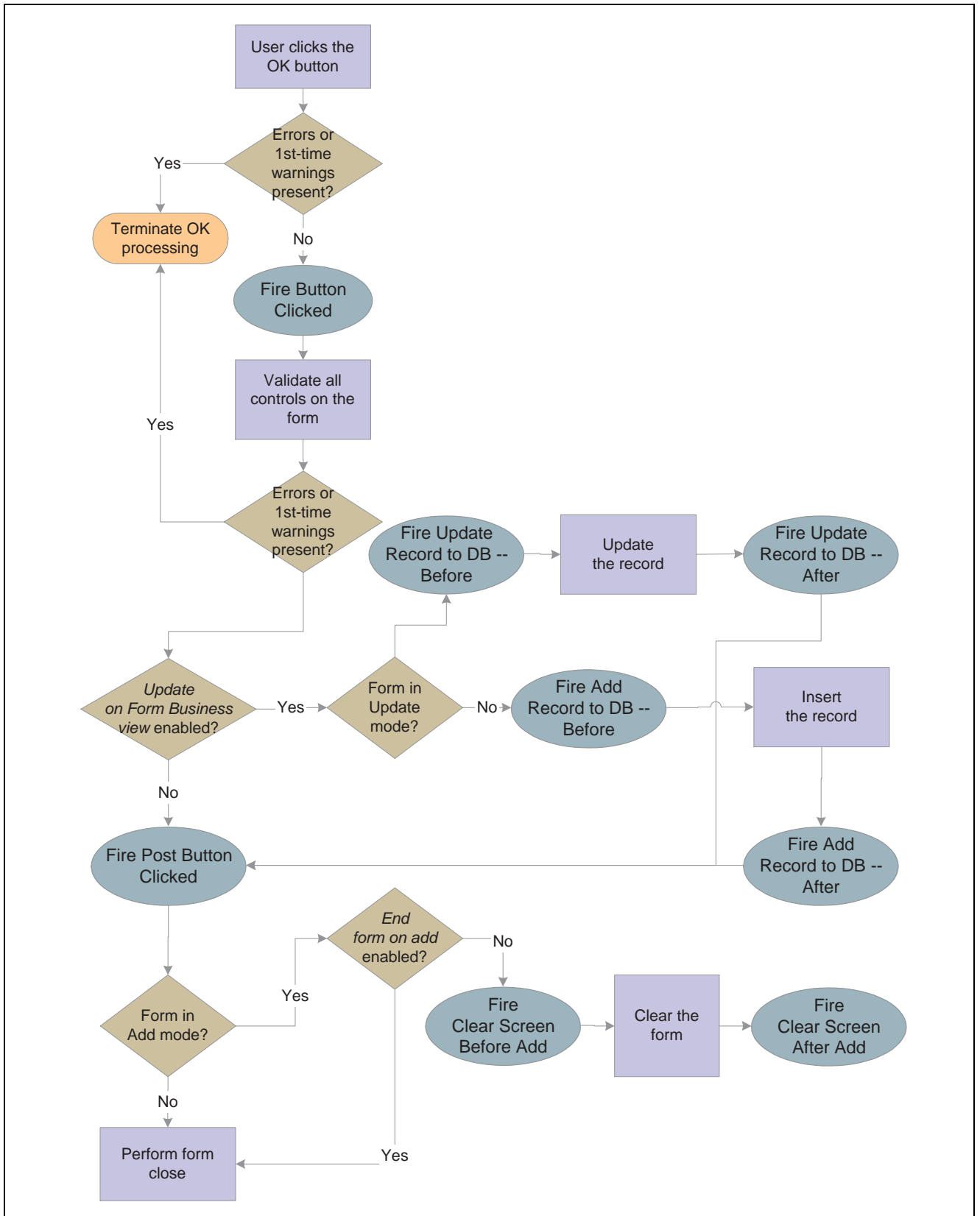
OK is a standard button on power edit forms that appears by default. It causes runtime to validate the information on the form and update or add it to the database through JDEKRNL function calls.

This flowchart illustrates the tasks that constitute runtime processing for the OK button, if the form contains a grid control:



Power edit form with grid control OK button processing

This flowchart illustrates the tasks that constitute runtime processing for the OK button, if the form does not contain a grid control:



Power edit form with no grid control OK button processing

Cancel Button

The Cancel button is a standard button on power browse forms that appears by default. When the user clicks it, runtime fires the **Button Clicked** and **Post Button Clicked** events in immediate succession. If no errors occur, runtime cancels any media objects that might be open. Also, if a manual transaction is in process, runtime attempts to cancel it as well. Then, runtime fires the **End Dialog** event and initiates the dialog close process.

Dialog Close

Power edit can be closed either by the user (typically by clicking the OK or Cancel buttons) or by the system. After performing any control-level close processing that might need to occur, runtime closes the form. If the event has not already occurred, runtime fires the **End Dialog** event. Then it performs these tasks in this order:

1. Load form interconnect data from BV columns for database commit.
2. Terminate error and thread handling.
3. Terminate helps.
4. Free all form structures.
5. Destroy the window.

CHAPTER 15

Understanding Search & Select Forms

This chapter discusses search & select forms.

Search & Select Forms

A search & select form is used to select a single predetermined field from a record in a predetermined file. Search & Select forms return only one value to the calling form, based on the dictionary alias. If no data dictionary (DD) alias matches the value, the first value from the data structure is returned.

Search & Select Events

These events can occur on the headerless detail form during runtime:

- **Dialog is Initialized**
- **Post Dialog is Initialized**
- **Grid Record is Fetched**
- **Last Grid Record Has Been Read**
- **Write Grid Line-Before**
- **Write Grid Line-After**
- **End Dialog**
- **XAPI Subscribe Event**

Search & Select Runtime Processing

This section discusses how runtime processes search & select forms. A search & select form should be attached as a visual assist only to data items that have the same data type as the form data structure element.

This section discusses form-level runtime processing only. Much of the runtime processing for the search & select “form” actually occurs on the level of the grid control, however.

See [Chapter 21, “Understanding Grid Controls,” How Runtime Processes the Grid Control, page 149.](#)

Dialog Initialization

When a search & select form is called, runtime initializes these items in this order:

1. Thread handling
2. Error handling process
3. Business view columns (BC)
4. Form controls (FC)
5. Grid fields
6. Static text
7. Helps
8. Event rules (ER) structures

Next, it performs these actions in this order:

1. Create the tool bar.
2. Load form interconnection data into corresponding BC and filter fields, if any.
3. Fire **Dialog is Initialized**.
4. Fire **Post Dialog is Initialized**.
5. If the Automatically Find On Entry option is selected, begin detail data selection and sequencing in the grid control.

If runtime does not encounter errors, this step leads to the population of the grid control, ultimately.

Find Button

The Find button is a standard button on headerless detail forms that appears by default. When the user clicks it, runtime fires the **Button Clicked** event. If no errors exist in the filter fields, runtime performs data selection and sequencing for the grid control. After reloading the grid with the fetched data, runtime fires the **Post Button Clicked** event.

Select Button

The Select button is a standard button on search & select forms that appears by default. When the user clicks it, runtime fires the **Button Clicked** event. If no errors occur, runtime writes the values from the selected row to the BC and fires the **Post Button Clicked** event. Then it fires the **End Dialog** event and initiates the dialog close process.

Close Button

The Close button is a standard button on search & select forms that appears by default. When the user clicks it, runtime fires the **Button Clicked** and **Post Button Clicked** events in immediate succession. If no errors occur, runtime attempts to close all of the modeless child forms, if any exist. If any of these child forms cannot be closed, the Close button process is terminated. Otherwise, runtime fires the **End Dialog** event and initiates the dialog close process.

Dialog Close

Search & Select can be closed either by the user (typically by clicking the Select or Close buttons) or by the system. After performing any control-level close processing that might need to occur, runtime closes the form. If the event has not already occurred, runtime fires the **End Dialog** event. Then it performs these tasks in this order:

1. Load form interconnect data from BC for database commit.
2. Terminate error handling.
3. Terminate helps.
4. Free all form structures.
5. Destroy the window.

CHAPTER 16

Understanding Wizard Forms

This chapter discusses wizard forms.

Wizard Forms

The wizard form is a simple form with the sole purpose of containing a wizard control. The wizard control is actually the wizard itself, for all intents and purposes. Each wizard control contains multiple pages; each page contains one subform and nothing else. Each subform in the wizard control is parented to this form.

During runtime, the wizard displays the first page and waits for the user to complete the task. Typically, you use the pages to prompt the user to provide input. After completing the task, the user clicks the Next button, and the wizard displays the second page in the list. In a standard scenario, the user continues in this fashion until the last page is reached, at which point the wizard validates and commits the data and then ends.

You cannot attach a business view (BV) to a wizard form. You cannot place any controls except the wizard control on the wizard form. Additionally, the wizard form does not display a tool bar or menus.

The wizard form has no unique property settings. Furthermore, the form does not support event rules (ER); all ER must be placed on the wizard control or its pages instead.

When the wizard runs, the wizard form expands to fill the entire frame. The EnterpriseOne navigation menu is unavailable while the wizard is running. Consequently, the user cannot launch any other EnterpriseOne applications until exiting the wizard.

See Also

[Chapter 34, “Understanding Wizard Controls,” page 269](#)

CHAPTER 17

Understanding Calendar Controls

This chapter discusses calendar controls.

Calendar Controls

With the calendar control, you provide users with a graphical display of a calendar which can show activities (for example, meetings and appointments) and other time sensitive information. It contains three views: day view, week view and month view. It is also fully customizable: you can hide any view, customize work hours and work weeks, and so forth.

The calendar control does not support a business view (BV); hence, it does not fetch or save calendar data. It relies on application logic (BVs and table I/Os) to manage data. However, it does provide a set of event rules (ER) to load, view the details of, and add, delete, or modify calendar activities. An *activity* is a calendaring and scheduling entity that represents a designated amount of time on a calendar. Every calendar event, reminder, and so forth that users or the system schedules is an activity.

Note. In Form Design Aid (FDA), events are points during runtime where you can execute code commands. However, events are also dates that you schedule on a calendar. To avoid confusion, use the term, *activity*, to mean the second kind of event.

The calendar control supports Universal Time (UT) and adjusts for user profile time zone settings. All activities appear based on the user's time zone daylight savings properties in user profile setting. The control itself does not include a function for reminders; therefore, reminder functions must be included as a part of the underlying application itself. The same is true for to-dos and attachments. If the application must display a to-do list, then it is recommended that you use a grid control to do so. The calendar control does not support the ability to attach items to an event directly on the control itself. However, an EnterpriseOne application can associate an attachment with each calendar activity and then display the attachment in the activity details form.

The calendar control does not support mobile devices currently.

The minimum dimensions of the control in FDA are 233 dialogue units high and 428 dialogue units wide. This is to ensure that the control will display correctly on 800 x 600 screen resolution settings. At a 1280 x 960 resolution, you might want a larger calendar control.

The only calendar control-specific design-time property you can set is which views are visible to the user. The calendar control has a variety of specialized system functions and ER.

Calendar Control Design-Time Considerations

You can choose to hide views from the user with a calendar control property. Alternatively, you can use the system function, **Set View Visible**, to hide and show views during runtime. Because the user must have at least one view to be able to interact with the control, FDA will not permit you to hide all of the views.

Calendar Control Events

These events, which are unique to calendar controls, can fire on this control:

- **Load Calendar Activity**
- **Drill Into Calendar Activity**
- **Drill Into Time Span**
- **Add Activity Button Clicked**
- **Post Add Activity Button Clicked**

Load Calendar Activity

When the form is opened, after the **Post Dialog Is Initialized** form-level event, the **Load Calendar Activity** event fires for all calendar controls on the form. Inside this event, the application can use table I/O or business functions to read calendar data and add the activities to the calendar control for display. The system provides two system variables: *SL_StartTime* and *SL_EndTime*. These values define the time range that the calendar control displays. The application logic should fetch all calendar activities that fall between *SL_StartTime* and *SL_EndTime*. Both *SL_StartTime* and *SL_EndTime* are UTIME type variables.

The **Load Calendar Activity** event fires when the calendar control must display a time period so that the application has an opportunity to load the calendar activities in that time period and to display them on the calendar. For example, when a user navigates to a new day, week, or month, the **Load Calendar Activity** is triggered so that the application can load the activities for the new day, week, or month.

The calendar control provides activity caching. When a day, week, or month is revisited for the second time, the calendar control displays the cached activities for the day, week, or month unless the cache is cleared by the application logic. (One way to clear the calendar cache is to call the system function, **Delete Calendar Activity**, to delete all activities.)

Whenever you want to load the calendar with events that exist in a given time period, use this formula:

```
SELECT from where T2>=S and T1<=E
```

where

- *T2* is the activity end time,
- *S* is the system variable, *Slstart*,
- *T1* is the activity start time, and
- *E* is the system variable, *Slend*.

Drill Into Calendar Activity

The calendar control enables a user to click an activity. When that happens, the **Drill Into Calendar Activity** event fires. Consequently, application logic can open an EnterpriseOne form to display the details of the activity for user to modify or delete.

The system outputs a system variable, *SL_CalendarActivityID*, containing the ID of the Activity being clicked.

Drill Into Time Span

The calendar control enables a user to click a time span containing an activity. When that happens, the **Drill Into Time Span** event fires. Consequently, application logic can open an EnterpriseOne form for the user to create a new event. System variables, *SL_StartTime* and *SL_EndTime*, indicate the start and end of the time span the user has clicked.

Double-clicking an empty time span (that is, a time span to which no activity is currently assigned) calls the **Add Calendar Activity** system function.

Add Activity Button Clicked and Post Activity Button Clicked

The calendar control provides an Add button. When this button is clicked, the system triggers the **Add Activity Button Clicked** event and then the **Post Add Activity Button Clicked** event. Applications can use these events to open an EnterpriseOne form for user to create a new activity.

Calendar Control Runtime Processing

This section discusses runtime processing for calendar controls.

No control-specific runtime processing occurs upon form close.

Initialize the Control

When runtime initializes the calendar control, it loads the control and then fires the event, **Load Calendar Activity**. What runtime loads into the control is based on any system functions defined in ER for this event. The date range that defines the initial load is determined by the view (day, week, or month) that has been set. Keep in mind that while up to three views can be visible, the user can interact with only one at a time—the current calendar view.

Calendar view can be set by the system function, **Select Calendar View**. Alternatively, if only one view is visible because the other two have been hidden either by the system function, **Set View Visible**, or by disabling them at design-time, then runtime sets the one remaining view. In all other scenarios, the order of precedence for the view to set is day, week, then month.

Hence, runtime sets the start and stop system variables (*SL_StartTime* and *SL_EndTime*, respectively) based on the current view. Then runtime passes the variables to **Load Calendar Activity**, and the system in turn populates the current view of the control. Only the activities for the current view are loaded. The control populates activities for other dates and views as the user travels to them.

Add a Calendar Activity

This scenario occurs most commonly as a result of control initialization or because a user travels to a date that he has not traveled to before during the course of the session. A user can travel to a new date by clicking the Next or Previous buttons.

Upon the button click, runtime uses ER to call the system function, **Load Calendar Activity**, with the start and end time being passed in as system values. Runtime derives the start and end times from the new date to which the user has just traveled. The system then populates the control with the newly fetched data.

After the initial fetch, data for each date is kept in memory to avoid unnecessary data refreshes. That is why runtime calls **Load Calendar Activity** only when the user travels to a date that has not been loaded for the current session.

Refresh the Control

Sometimes, you might want to force a refresh. To do so, call the system function, **Delete Calendar Activity**, and pass in *Delete All Activities* for the *Activity ID* parameter. Runtime responds by removing all activities from the entire control and flushing the memory where previously traveled-to dates are stored. Runtime also sets the start and stop system variables to match the view that is currently set. In other words, runtime places the control in a condition similar to being newly initialized.

After calling **Delete Calendar Activity**, call **Load Calendar Activity**, passing in the start and stop system variables (*SL_StartTime* and *SL_EndTime*) to populate the current control view.

Calendar Control System Functions

This section discusses the system functions unique to the calendar control.

In general, you can use system functions to perform these types of tasks:

- Add, modify, or delete an activity from the calendar control.
- Classify time ranges as being working, nonworking, or of another special type.
- Modify aspects of the calendar control interface.

When you add an activity through the system function, **Add Calendar Activity**, or modify an activity using the system function, **Modify Calendar Activity**, you can affect these aspects of how the activity appears and acts on the calendar:

- Name of the activity (subject).
- Start and end day and time.

The system requires this value in JDEUTime, even when you designate an activity as an all-day activity. You can also flag an activity for reoccurrence.

If the start and stop times result in a span of time that is greater than or equal to 24 hours, then the system converts the activity to an all day activity. All day activities are displayed in the all day event area in the interface.

- Additional information associated with the activity.

You can set values for the type (meeting, appointment, and so forth), the commitment level (low, medium, or high), the activity location, and the activity organizer.

- Private.

This option causes the system to display a special icon next to the activity. No other logic is applied automatically, however. If you wish to restrict access to an event marked private, then you must do so within the application.

- Font, color, and icons.

You can set the font and related attributes, set a background color, and associate an icon with any activity. The icons from which you have to choose must already reside in the system. To add an icon to the list, save a GIF-type graphic in both of these locations: the B9\<pathcode>\res folder on the machine where FDA is located and the webclient.war/img/res folder on the machine or machines hosting the web server.

When modifying an event, ensure that the ID value for the event exactly matches the ID as it appears in the table. For example, if the table column is left-padded, you must prefix the event ID as it appears in the control with the same number of characters that have been added to the event ID in the table because of the column formatting.

When you delete an activity using the system function, **Delete Calendar Activity**, you remove it from the calendar control; you do not remove it from the database. You can choose to delete a single activity or multiple ones. You can also choose to clear the control of all activities.

In some cases, it might be helpful to assign different classifications to different date ranges. Typical classifications include working and nonworking (for example, holidays). Use **Set Work Day Hours** and **Set Work Week** to designate work time. Currently, **Set Work Day Hours** only affects full hours; it cannot accept half-hour increments, for example.

Furthermore, the work week must not contradict the work week system settings for the current country. For example, in the US, work weeks cannot end on Sunday or start on Saturday. Additionally, a work week cannot span these start and stop days. For example, you cannot start a work week on a Tuesday and end on a Monday.

Use **Set Day Type** to classify days as belonging to any other special categories.

The calendar control can provide users with three views: day, week, and month. Use **Select Calendar View** to display the calendar in a specific view type. You can also prevent users from being able to see a particular view type with **Set View Visible**.

The calendar control also provides an Add button. To change the text on the Add button, use **Set Add Button Text**. To hide the Add button and therefore prevent users from adding their own activities, use **Set Add Button Visible**.

Note. The Add button has no underlying logic. If you choose to display the button, you should assign it some function on the **Add Activity Button Clicked** event.

Add Calendar Activity

Description

Use this system function to add one activity to the calendar control.

Parameters

Parameter	Description
<i>Calendar Control</i>	Input, required. The calendar form control (FC) to affect.
<i>Activity ID</i>	Input, required. The ID of the activity in the database to be added to the calendar. Set the parameter to an applicable object from the object list.
<i>Subject</i>	Input, required. The title of the activity as it should appear on the calendar for the user. Set the parameter to an alphanumeric constant (<Literal>) or an applicable object from the object list.
<i>Start Date and Time</i>	Input, required if <i>All Day Activity</i> = False. The starting time and date, in UTime, for the activity. Literal date value in JDEUTime format.

Parameter	Description
<i>End Date and Time</i>	Input, required if <i>All Day Activity</i> = False. The ending time and date, in UTime, for the activity. Literal date value in JDEUTime format.
<i>All Day Activity</i>	Input, required. The indicator of whether the activity is to be displayed as an all day activity instead of having start and end date and times. Any activity that is equal to or greater than 24 hours automatically becomes an all day activity. All day activities are displayed in the all day activity area in the interface. Set the parameter to <TRUE>, <FALSE>, or an applicable object from the object list.
<i>Recurrence</i>	Input, required. The indicator of whether the activity occurs more than once. This parameter merely sets a flag; the control displays an icon to indicate that the activity has been flagged as reoccurring, but the system performs no other action in connection with recurrence. Set the parameter to <TRUE>, <FALSE>, or an applicable object from the object list.
<i>Type</i>	Input, required. The classification of the activity (meeting, appointment, and so forth). This parameter does not appear on the calendar. Set the parameter to an alphanumeric constant (<Literal>) or an applicable object from the object list.
<i>Commitment Level</i>	Input, required. The relative value of the activity. The value assigned affects the icon displayed and the color of the duration bar for the activity. Set the parameter to <Low Commitment> (1), <Default> (2), or <High> (3).
<i>Private</i>	Input, required. The indicator of whether to display the "private" icon. Set the parameter to <TRUE>, <FALSE>, or an applicable object from the object list.
<i>Location</i>	Input, required. The name of the place where the activity is to occur as it should appear on the calendar for the user. Set the parameter to an alphanumeric constant (<Literal>), <N/A>, or an applicable object from the object list.
<i>Organizer</i>	Input, required. The name of the creator of the activity as it should appear on the calendar for the user. Set the parameter to an alphanumeric constant (<Literal>), <N/A>, or an applicable object from the object list.
<i>Font</i>	Input, required. The font to use to display the activity to the user. Set the parameter to a font and related settings from the Font dialog (<Pick Font>) or the default font and related settings (<Reset Font>).
<i>Bitmap</i>	Input, required. The bitmap of an icon to display to the user on the calendar in association with the activity. To add an icon to the list, save a GIF-type graphic in both of these locations: The B9\<pathcode>\res folder on the machine where FDA is located and the webclient.war/img/res folder on the machine or machines hosting the Web server. Set the parameter to a bitmap from the system-supplied list (<Choose Calendar Bitmap>) or the default icon (<Default Calendar Bitmap>).
<i>Background Color</i>	Input, required. The color to appear behind the information on the calendar. Set the parameter to a color from the color palette (<Pick Color>) or the default color (<Reset Color>).

Additional Notes

When you add an activity through the system function, **Add Calendar Activity**, you can affect these aspects of how the activity appears and acts on the calendar:

- Name of the activity (subject).
- Start and end day and time.

The system requires this value in JDEUTime, even when you designate an activity as an all-day activity. You can also flag an activity for reoccurrence.

If the start and stop times result in a span of time that is greater than or equal to 24 hours, then the system converts the activity to an all day activity. All day activities are displayed in the all day event area in the interface.

- Additional information associated with the activity.

You can set values for the type (meeting, appointment, and so forth), the commitment level (low, medium, or high), the activity location, and the activity organizer.

- Private.

This option causes the system to display a special icon next to the activity. No other logic is applied automatically, however. If you wish to restrict access to an event marked private, then you must do so within the application.

- Font, color, and icons.

You can set the font and related attributes, set a background color, and associate an icon with any activity. The icons from which you have to choose must already reside in the system. To add an icon to the list, save a GIF-type graphic in both of these locations: the B9\<pathcode>\res folder on the machine where FDA is located and the webclient.war/img/res folder on the machine or machines hosting the web server.

Whenever you want to load the calendar with events that exist in a given time period, use this formula:

```
SELECT from where T2>=S and T1<=E
```

where

- *T2* is the activity end time,
- *S* is the system variable, *Slstart*,
- *T1* is the activity start time, and
- *E* is the system variable, *Slend*.

Delete Calendar Activity

Description

Use this system function to delete an existing activity from the calendar control.

Parameters

Parameter	Description
<i>Calendar</i>	Input, required. Control: The calendar FC to affect.
<i>Activity ID</i>	Input, required. The ID of the activity to be removed from the calendar. Set the parameter to a parameter to delete all events currently loaded for the control (< <i>Delete All Activities</i> >) or an applicable object from the object list.

Additional Notes

When you delete an activity using this system function, you remove it from the calendar control; you do not remove it from the database. You can choose to delete a single activity or multiple ones. You can also choose to clear the control of all activities.

Modify Calendar Activity

Description

Use this system function to modify an existing event on the calendar control.

Parameters

Parameter	Description
<i>Calendar</i>	Input, required. The calendar FC to affect.
<i>Activity ID</i>	Input, required. The ID of the activity to be changed on the calendar. Ensure that this value exactly matches the ID (including padding and so forth) as it appears in the table. Set the parameter to an applicable object from the object list.
<i>Subject</i>	Input, required. The title of the activity as it should appear on the calendar for the user. Set the parameter to the current value (<No Change>), an alphanumeric constant (<Literal>) or an applicable object from the object list.
<i>Start Date and Time</i>	Input, required if <i>All Day Activity</i> = False. The starting time and date, in JDEUTime, for the activity. Set the parameter to the current value (<No Change>) or a literal date value in JDEUTime format.
<i>End Date and Time</i>	Input, required if <i>All Day Activity</i> = False. The ending time and date, in JDEUTime, for the activity. Set the parameter to the current value (<No Change>) or a literal date value in JDEUTime format.
<i>All Day Activity</i>	Input, required. The indicator of whether the activity is to be displayed as an all day activity instead of having start and end date and times. Any activity that is equal to or greater than 24 hours automatically becomes an all day activity. All day activities are displayed in the all day activity area in the interface. Set the parameter to the current value (<No Change>), <TRUE>, <FALSE>, or an applicable object from the object list.
<i>Recurrence</i>	Input, required. The indicator of whether the activity occurs more than once. This parameter merely sets a flag; the control displays an icon to indicate that the activity has been flagged as reoccurring, but the system performs no other action in connection with recurrence. Set the parameter to the current value (<No Change>), <TRUE>, <FALSE>, or an applicable object from the object list.
<i>Type</i>	Input, required. The classification of the activity (meeting, appointment, and so forth). This parameter does not appear on the calendar. Set the parameter to the current value (<No Change>), an alphanumeric constant (<Literal>) or an applicable object from the object list.
<i>Commitment Level</i>	Input, required. The relative value of the activity. The value assigned affects the icon displayed and the color of the duration bar for the activity. Set the parameter to the current value (<No Change>), <Low Commitment> (1), <Default> (2), or <High> (3).
<i>Private</i>	Input, required. The indicator of whether to display the "private" icon. Set the parameter to the current value (<No Change>), <TRUE>, <FALSE>, or an applicable object from the object list.
<i>Location</i>	Input, required. The name of the place where the activity is to occur as it should appear on the calendar for the user. Set the parameter to an alphanumeric constant (<Literal>), <N/A>, or an applicable object from the object list.

Parameter	Description
<i>Organizer</i>	Input, required. The name of the creator of the activity as it should appear on the calendar for the user. Set the parameter to the current value (<No Change>), an alphanumeric constant (<Literal>), <N/A>, or an applicable object from the object list.
<i>Font</i>	Input, required. The font to use to display the activity to the user. Set the parameter to the current value (<No Change>), a font and related settings from the Font dialog (<Pick Font>) or the default font and related settings (<Reset Font>).
<i>Bitmap</i>	Input, required. The bitmap of an icon to display to the user on the calendar in association with the activity. To add an icon to the list, save a GIF-type graphic in both of these locations: The B9\<pathcode>\res folder on the machine where FDA is located and the webclient.war/img/res folder on the machine or machines hosting the Web server. Set the parameter to the current value (<No Change>), a bitmap from the system-supplied list (<Choose Calendar Bitmap>) or the default icon (<Default Calendar Bitmap>).
<i>Background Color</i>	Input, required. The color to appear behind the information on the calendar. Set the parameter to the current value (<No Change>), a color from the color palette (<Pick Color>) or the default color (<Reset Color>).

Additional Notes

When you modify an activity using the system function, **Modify Calendar Activity**, you can affect these aspects of how the activity appears and acts on the calendar:

- Name of the activity (subject).
- Start and end day and time.

The system requires this value in JDEUTime, even when you designate an activity as an all-day activity. You can also flag an activity for reoccurrence.

If the start and stop times result in a span of time that is greater than or equal to 24 hours, then the system converts the activity to an all day activity. All day activities are displayed in the all day event area in the interface.

- Additional information associated with the activity.

You can set values for the type (meeting, appointment, and so forth), the commitment level (low, medium, or high), the activity location, and the activity organizer.

- Private.

This option causes the system to display a special icon next to the activity. No other logic is applied automatically, however. If you wish to restrict access to an event marked private, then you must do so within the application.

- Font, color, and icons.

You can set the font and related attributes, set a background color, and associate an icon with any activity. The icons from which you have to choose must already reside in the system. To add an icon to the list, save a GIF-type graphic in both of these locations: the B9\<pathcode>\res folder on the machine where FDA is located and the webclient.war/img/res folder on the machine or machines hosting the web server.

When modifying an event, ensure that the event's ID value exactly matches the ID as it appears in the table. For example, if the table column is left-padded, you must prefix the event ID as it appears in the control with the same number of characters that have been added to the event ID in the table because of the column formatting.

Whenever you want to load the calendar with events that exist in a given time period, use this formula:

SELECT from where $T2 \geq S$ and $T1 \leq E$

where

- $T2$ is the activity end time,
- S is the system variable, $Slstart$,
- $T1$ is the activity start time, and
- E is the system variable, $Slend$.

Select Calendar View

Description

Use this system function to determine which view of the calendar (day, week, or month) to display to the user upon an event trigger.

Parameters

Parameter	Description
<i>Calendar</i>	Input, required. The calendar FC to affect.
<i>Calendar View</i>	Input, required. The calendar view (day, week, or month) to display to the user. Set the parameter to <Day View > (0), <Week View > (1), or <Month View> (2).

CHAPTER 18

Understanding Check Box Controls

This chapter discusses check box controls.

Check Box Controls

Use check boxes to indicate choices. Selecting a check box indicates that the associated function is to be enabled. Check boxes are not mutually exclusive. You must associate a check box with a database or data dictionary (DD) item. You can use a check box to pass a value to an event rule (ER).

Check Box Control Design-Time Considerations

In addition to the standard control property options, check boxes have two unique properties: **Checked Value** and **Unchecked Value**. The control returns one of these values, depending on its state. The default values are 1 for selected (checked) and 0 for cleared (unchecked). You can set the values to any numeric value.

Check Box Events

The only event that can occur on a check box is **Selection Changed**. Runtime fires this event whenever the user selects or clears the check box, so it can occur at any point during runtime processing that the user interacts with the application.

CHAPTER 19

Understanding Combo Box Controls

This chapter discusses the combo box control.

Understanding the Combo Box Control

You can use a combo box (drop-down list) to display a list of items from which the user can make a selection. The combo box includes a type-ahead feature where typing the first character of a matching item description will select that item in the control. It can reside on a form or inside a grid control or in the grid of a parent child control. When inside a grid control, the combo box acts exactly as a normal grid cell except when it has the focus. At that point, it behaves as a combo box; that is, the user can pick an item from the list by clicking and choosing an item from the drop-down, by typing directly in the field, or by using the arrow and enter keys on the keyboard.

You must associate the combo box with a data dictionary (DD) item. If the associated DD item has user-defined code (UDC) values, those values are used to load the combo box list. During runtime, you can load the control with values from a different UDC, if necessary. Alternatively, you can use event rules (ER) in the application to load the values without reference to a UDC, although doing so precludes the ability to directly include translated text in the drop-down list. In either case, using a system function to remove items, you can dynamically filter the list before it is displayed to the user.

No matter what is loaded into the control, upon initialization, the control always displays -- Select One -- as the currently selected item to indicate that the user should select an item in the control. This is true of combo boxes both on a form and in a grid. Although it is an item in its own right, you cannot remove it from the list. --Select One-- always appears at the top of the list; you cannot insert a list item above it.

Note. -- Select One -- is merely a prompt—an indication to the user to make a selection. --Select One -- is not an actual value. Therefore, it will never be written to the database with the intent of saving it as actual data.

Loading the Combo Box Control

You can load the combo box control from any of these data sources:

- Its associated DD UDCs.
- Cache (using the **Load from Cache** system function).
- Any source available to the **Add Item** system function.

In all cases, runtime checks each item for valid *Key* and *Item* parameter pairs. If the *Key* is null, or its matching description is null or an empty string, runtime throws an error and does not load that item. Also, to avoid duplicate items from appearing in the list, runtime does not load an item if it finds the item's *Key* parameter value in the combo box's current item list. This check avoids duplicate list items; however, it does not prevent two (or more) different list items with the same name (Description parameter) from appearing in the list.

No matter what method you choose to load the control, ensure that the user has at least one option from which to choose. As part of the initialization process, if runtime cannot find at least one value to put in the drop-down list, it disables the combo box control until one or more values are loaded into the control.

In some cases, the values to load into the combo box might include an item where the *Key* is " " (single blank) and the description is "(None)." In fact in EnterpriseOne applications, this value has been used as a blank default value, historically. Although this is a valid item to load into the combo box, it is recommended that you ascribe a meaningful text string to such an item that makes sense in the context of the application as opposed to descriptions such as (None), or " ."

Note. As of release 8.94, the practice of the runtime engine automatically loading a combo box that is configured in Form Design Aid (FDA) as being not required with a default combo box item of key = " " (single blank) and description = "(None)" is discontinued. If such an item must be loaded in the context of the application in question, that task is now the responsibility of the application developer to define and insert into the control.

Loading a Combo Box from a UDC

The combo box control supports string and numeric data types for the combo box's keys and the matching descriptions are always treated as strings. Dynamic changes to a combo box based on a UDC can be made with the system functions **Set Data Dictionary Item** or **Set Data Dictionary Overrides** to load a new set of key (value)/description pairs.

If the application must support multiple languages, loading from a UDC is probably an ideal choice because of the way EnterpriseOne applies alternate UDC lists automatically based on the user's language code. An alternative to dynamically changing the values displayed in a combo box at runtime is to use the **Set Data Dictionary Item** or **Set Data Dictionary Overrides** system functions to load the correct set of values as needed. You can suppress inappropriate list items within an existing combo box item list using the **Remove Item** system function. If you must set up the combo box based on a non-UDC DD item but translated descriptions still need to be displayed to the user, you have some options.

You can use the ER system value, *SL LanguagePreference*, to acquire user's language preference. Use that value to fetch and load items into the combo box manually from the appropriate UDC table (F0005 standard UDC table, or F0005D translated UDC table).

Alternatively, you can define the necessary description items as text variables in the application and then use these text variables in conjunction with the **Add Item** system function to add the items to the combo box.

Runtime always loads from the UDC if one is associated with the combo box's data dictionary item. If the combo box is loaded from a UDC and you want to load from a non-UDC source instead, use **Set Data Dictionary Item** to associated a data dictionary item that does not have a UDC with the combo box. Then you can load from cache or use **Add Item** instead.

Loading a Combo Box from Cache

The **Load From Cache** system function enables you to load a combo box from an EnterpriseOne cache data source. Runtime will load values from cache only if the target combo box control is based on a valid DD item that does not have a UDC list associated with it.

When working from cache, you must have a prepared data structure to govern loading data from source into the cache and then transferring it from the cache to the combo box control. When you choose (or create) a data structure to use for this purpose, ensure that the *Key*, *Item*, and optional *Filter Column* parameter data types are compatible with the corresponding data types demanded by the system functions you will call.

Note. All of the values are retrieved from cache prior to filtering. Therefore, you should avoid loading large quantities of data into cache to reduce the possibility of poor performance.

The EnterpriseOne cache from which the combo box item data will be read must have defined a distinct column containing one or more rows of data for each of these combo box data items:

- E1 Cache Column - Keys (values)
- E1 Cache Column - Items (descriptions)
- E1 Cache Column - Filter Column Values (optional filter value)

If the optional filter cache column is not designated or does not contain valid data, runtime will retrieve and load into the combo box all of the key/description pairs currently loaded in the specified cache. After the combo box has been loaded from the cache, you can dynamically remove items from its item list using the **Remove Item** system function.

Loading a Combo Box with the Add Item System Function

To load a small number of items into a combo box, you can use the **Add Item** system function. Note that this system function can be used only when the combo box does not have UDC values loaded into it. If translated description text is needed, you can use text variables for the item descriptions or fetch the data manually from the appropriate UDC table (F0005, or F0005D) based on the user's language preference that is accessible through the system value, *SL LanguagePreference*.

See Also

Chapter 19, "Understanding Combo Box Controls," Combo Box Control System Functions, page 124

Understanding Combo Box Control Design-Time Considerations

The combo box control has no unique property settings. Among the more standard property settings, the Required flag deserves mention in relationship to the combo box control. Recall that when initialized, the control simply displays the user prompt, -- Select One --. If the Required flag is enabled and this prompt is displayed during validation, the engine throws an error and prevents the user from continuing until he or she chooses an item from the list. On the other hand, if the flag is disabled under the same circumstance, the engine will actually write a blank (that is, key =) to the database if the "-- Select One --" prompt is still selected at the time the OK or Save button is clicked. The blank is written to the database in this case because the system cannot save an empty string () as a value for a key.

Understanding Combo Box Control Events

These events can fire on the combo box control during runtime:

- **Selection Changed**
- **Control is Entered**

- **Control is Exited**

Runtime fires the **Selection Changed** event when the user or application chooses an item in the combo box list that is different from the item currently selected. In Microsoft Internet Explorer (IE), this selection can be made using any of these methods:

- User chooses an item with the mouse from the combo box's drop-down list.

With the drop-down list displayed, the user can also type the first character of a matching item, or use the up and down arrow keys.

- User places the focus on the combo box without activating its drop-down list and types the first character matching that of an item description in the combo box's list of items (type-ahead feature).

If more than one item that matches this character exists, the user can cycle through all of the matching items by pressing the same character key multiple times. Each time a new item is selected, the **Selection Changed** event fires.

- User places focus on the combo box without activating its drop-down list and then uses the up and down arrow keys to cycle through the combo box's item list until the desired item is found.

Each time a new item is selected, the **Selection Changed** event fires.

- Combo box is the target of an ER assignment statement and the value assigned to it is a valid key value matching one of the current item key values loaded in the control.
- Combo box and a valid key value are passed into the combo box system function, **Select Item**.

In some explorers other than IE (such as Mozilla), the user might need to tab out of the combo box to trigger the **Selection Changed** event, particularly if the selection was made using the keyboard.

Of course, runtime fires **Control is Entered** when the user changes the focus to the check box by any of these means, and fires **Control is Exited** when the control loses focus.

Understanding Combo Box Control Runtime Processing

This section describes how the runtime engine processes the combo box.

Control Initialization

When runtime loads the control, items with a blank key are valid, but display no description if the description parameter is also blank. If the item has a key which is a string of one or more continuous blank characters, runtime concatenates the multiple blank characters to a single blank character. Similarly, the engine removes blank characters that follow an otherwise non-blank item key value prior to inserting it into the control.

Note. The previous behavior of the HTML runtime engine automatically loading a combo box item having:

key = or (depending on SP level)

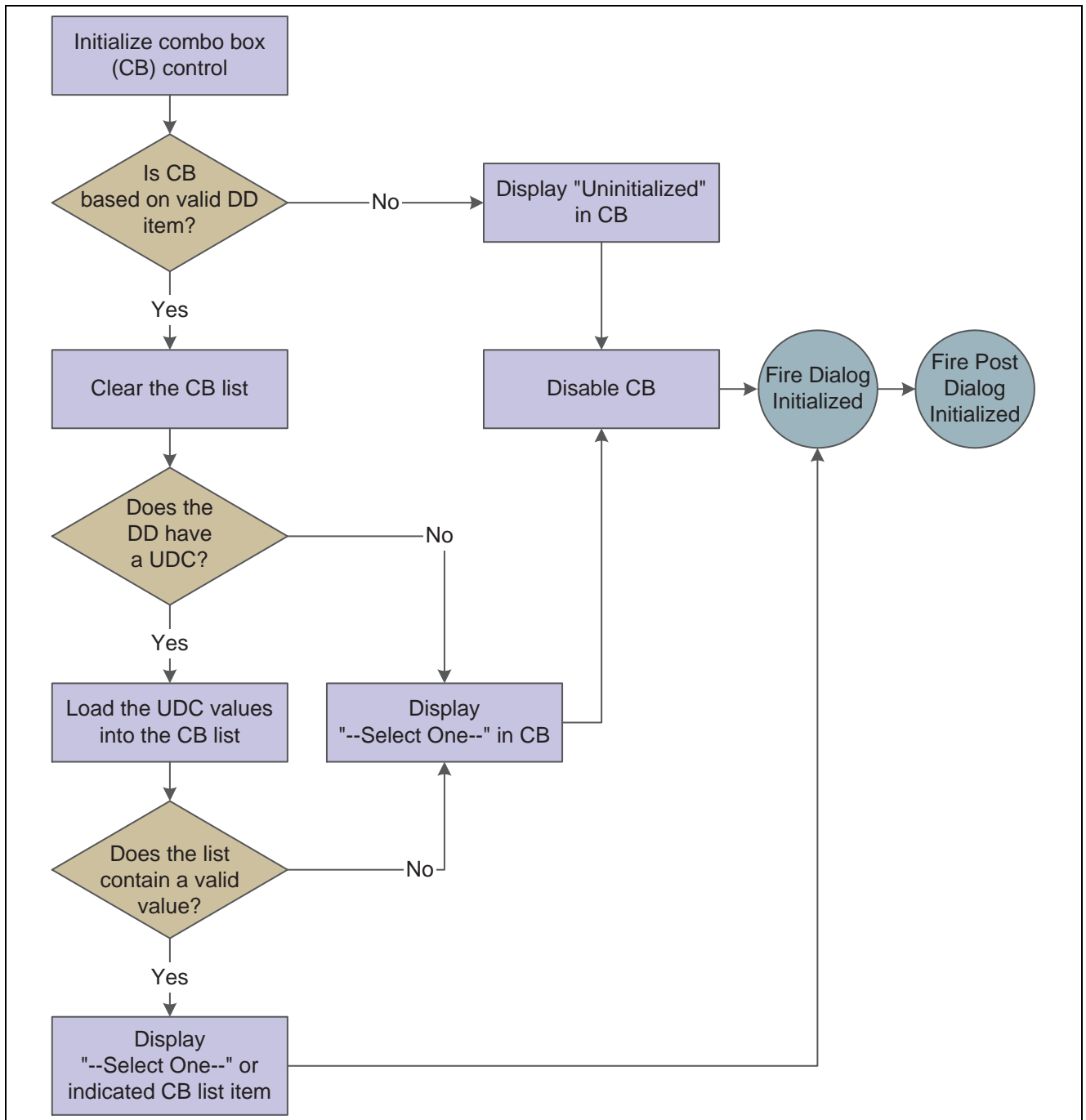
and

item description = (None)

when the control is marked in FDA as being not required is discontinued as of EnterpriseOne 8.94.

Additionally, to be functional, a combo box must have a valid DD item associated with it. Combo boxes that do not have such an associated item are rendered displaying the text string, "Uninitialized," and are disabled automatically.

The control initialization process is illustrated in this flowchart:



Combo box control initialization process

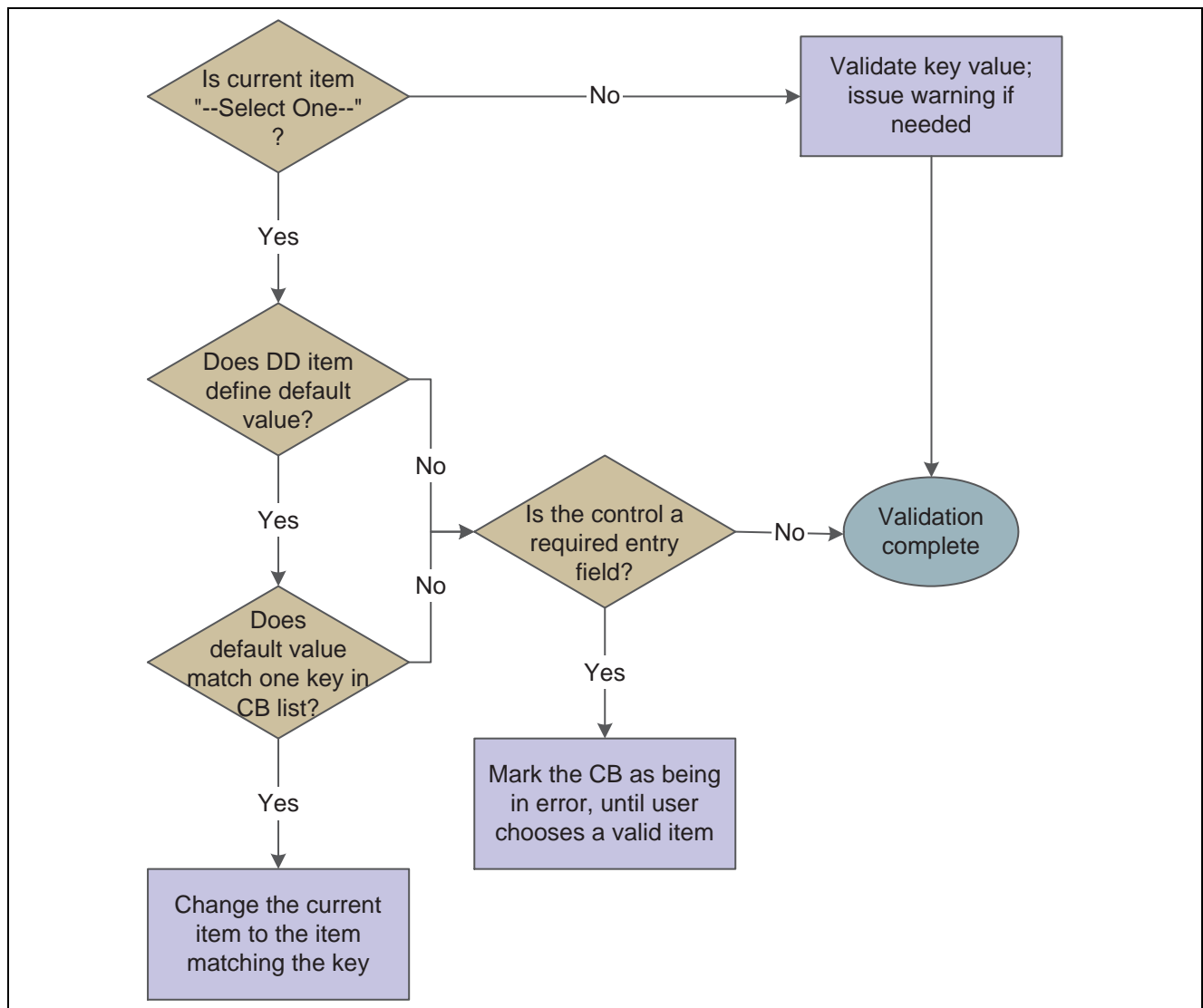
Generally two cases exist where runtime might show an item description other than the "-- Select One --" prompt as the currently selected item in a combo box when a form appears initially:

- When a given combo box's associated DD item has a default value defined and you come into the form in Add mode, runtime attempts to match the default value to the keys of the items currently loaded in the combo box and sets that item as the selected item.
- Dependent on form type, regardless of whether the form was started using form interconnect, the current form mode (Add/Copy/Update), and whether the control is mapped to a business view column (Key or otherwise), runtime might have retrieved a key value also and will try to set this as the current selected item in the combo box.

If you want to load the combo box from cache or using the **Add Item** system function, the event most typically used for such loads is **Post Dialog Initialized**.

Control Validation

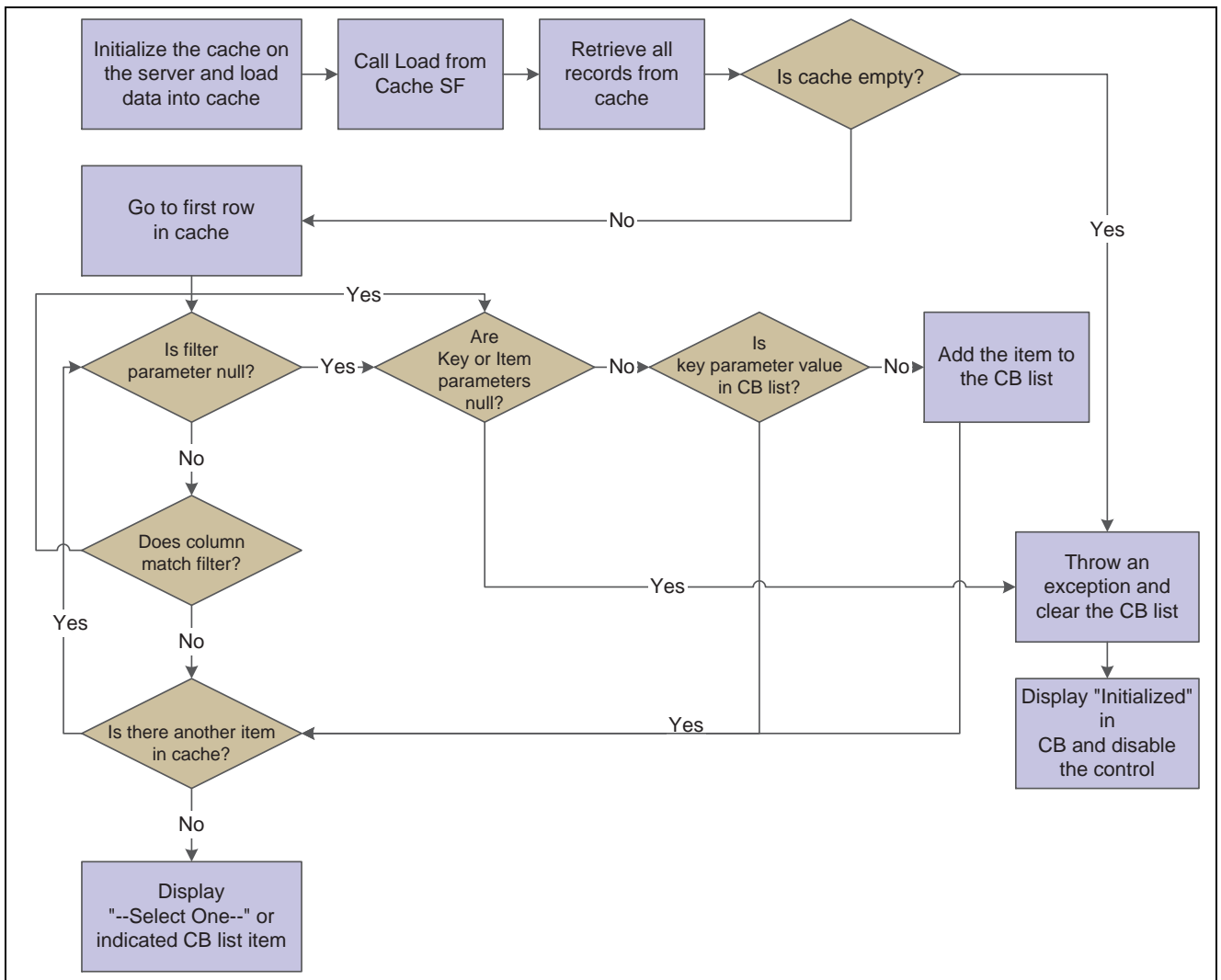
Runtime performs the validation process illustrated in this flowchart when the control is validated:



Combo box control validation process

Load from Cache

This flowchart illustrates how runtime loads the combo box from cache when the **Load from Cache** system function is processed. If an error occurs at any point during the processing, any data that might have been loaded into the combo box is cleared and the control is disabled:



Combo box control load from cache process

Database Commit

When an application signals a database commit, runtime writes the key value (instead of the displayed description) of the currently-selected combo box item to the database. If the field is flagged as not required and the currently-selected value is -- Select One --, runtime saves a string of continuous blank characters matching the size of the associated table column. The system cannot save the actual matching empty string ("") key value associated with the "-- Select One --" prompt to the database, which is why it saves a string of blank characters instead.

System Functions

When you call, **Set Data Dictionary Item**, **Set Data Dictionary Override**, or **Load From Cache**, runtime clears the control of its current items prior to loading any new data specified by the system function call. This is not only true of combo boxes on a form, but in a grid as well. If the new data dictionary item associated with the system function call or associated UDC override results in a change from the current DD Item and or UDC specified for the grid control, then runtime removes all list items from all embedded combo box controls for all current rows of the column, except for the default -- Select One -- prompt. If this action results in a new set of UDC values, then runtime loads those values into every current embedded combo box control for all of the rows in the grid or Parent Child control.

Import into Grid

HTML users can import data into a grid containing embedded combo boxes just as then can into a grid without the EnterpriseOne grid import/export feature. When a user does so, the value being imported into the embedded combo box must be the Key parameter value that corresponds to a valid combo box key/item (description) pair.

If the target cell resides in a grid column that has a data dictionary item with an associated UDC, then runtime creates an embedded combo box in the target cell and loads the UDC-defined key/item pairs into it. Next, runtime attempts to match the imported key to one of the currently loaded keys. If it finds a match, runtime sets it as the currently-selected item in the cell. Otherwise, it ignores the item and sets the cell to the -- Select One -- value.

If the target cell resides in a grid column that does not have a data dictionary item with an associated UDC, then runtime creates an embedded combo box with one item set as selected having a Key parameter of (blank) and an Item (description) parameter of Uninitialized. Then runtime disables the cell.

Combo Box Control System Functions

This chapter describes the system functions unique to the combo box control when it resides on a form (that is, it is not embedded in a grid).

Add Item

Description

Use this system function to add an item to the combo box if it is not displaying a list that is based on a UDC.

Parameters

Parameter	Description
<i>ComboBox Control</i>	Input, required. The combo box FC to affect.
<i>Key</i>	Input, required. The key of the item to add (string or math-numeric). Set the parameter to an alphanumeric constant (<Literal>) or applicable object from the object list.
<i>Item</i>	Input, required. The name of the item to add (string). Set the parameter to an alphanumeric constant (<Literal>) or applicable object from the object list.
<i>Index</i>	Input, optional. The zero-based index position where the item is to added (integer). If you do not specify this parameter, runtime appends the item to the bottom of the list. The default "-- Select One --" prompt always occupies index position zero. If an invalid index is specified, the addition of the new item fails and an error message is logged. Set the parameter to an alphanumeric constant (<Literal>) or applicable object from the object list.

Additional Notes

Use this system function to add an item to the combo box if it is not displaying a list that is based on a UDC (however, the control itself must be based on a valid DD item). Runtime verifies that the Key and Item parameter values are both specified. If either is missing, the addition of the new item fails and an error is logged. If you try to insert an item having a key value that already exists in the list, runtime will not add the item and will log an error.

The combo box control must reside on the form (as opposed to being embedded in a grid). The control must have a valid data dictionary item associated with it which does not have a related UDC.

Get Description

Description

Use this system function to get the displayed description of the current selected item of a specified combo box.

Parameters

Parameter	Description
<i>ComboBox</i>	Input, required. The combo box FC to affect.
<i>Item</i>	Input, required. The object to which to assign the return value that designates the description text of the current selected item (string). Set the parameter to an applicable object from the object list.

Returns

This system function returns the name of a combo box item to the object identified by *Item*.

Get Index of Key

Description

Use this system function to get the zero-based index position of an item in a combo box list by specifying its key. If the key value specified does not exist in the combo box's current set of items, no index value is returned and an error message is logged.

Parameters

Parameter	Description
<i>ComboBox Control</i>	Input, required. The combo box FC to affect.
<i>Key</i>	Input, required. The key of the target item (string or math-numeric). Set the parameter to an alphanumeric constant (<Literal>) or applicable object from the object list.
<i>Index</i>	Input, required. The object to which to assign the return value that designates the zero-based index position of the target item (integer). Set the parameter to an alphanumeric constant (<Literal>) or applicable object from the object list.

Returns

This system function returns the zero-based index position of a combo box item to the object identified by *Index*.

Get Item at Index

Description

Use this system function to get the displayed description of an item in a combo box list by specifying its index position in the list. If the index value specified is invalid, no description is returned and an error message is logged.

Parameters

Parameter	Description
<i>ComboBox Control</i>	Input, required. The combo box FC to affect.
<i>Index</i>	Input, required. The zero-based index position of the target item (integer). Set the parameter to an alphanumeric constant (<Literal>) or applicable object from the object list.
<i>Item Description</i>	Input, required. The object to which to assign the return value that designates the displayed description text of the target item (string). Set the parameter to an applicable object from the object list.

Returns

This system function returns the name of a combo box item to the object defined by *Item Description*.

Get Item Count

Description

Use this system function to determine the total number of items in the combo box list including the -- Select One -- prompt.

Parameters

Parameter	Description
<i>ComboBox Control</i>	Input, required. The combo box FC to affect.
<i>Number</i>	Input, required. The object to which to assign the return value that designates the number of list items. Set the parameter to an applicable object from the object list.

Returns

This system function returns the number of items in the combo box list to the object identified by *Number*.

Get Key at Index

Description

Use this system function to get the key of an item in a combo box list by specifying its index position in the list. If the specified index value is invalid, no key value is returned and an error message is logged.

Parameters

Parameter	Description
<i>ComboBox Control</i>	Input, required. The combo box FC to affect.
<i>Index</i>	Input, required. The zero-based index position of the target item (integer). Set the parameter to an alphanumeric constant (<Literal>) or applicable object from the object list.
<i>Key</i>	Input, required. The object to which to assign the return value that designates the key of the target item (string or math-numeric). Set the parameter to an alphanumeric constant (<Literal>) or applicable object from the object list.

Returns

This system function returns the key of a combo box item to the object defined by *Key*.

Load from Cache

Description

Use this system function to load the combo box control from an EnterpriseOne cache. The control must be based on a DD item which does not have an associated UDC defined for it.

Parameters

Parameter	Description
<i>ComboBox Control</i>	Input, required. The combo box FC to affect.
<i>Cache Name</i>	Input, required. The name of the prepopulated EnterpriseOne cache from which to load combo box items (string). Set the parameter to an applicable object from the object list.
<i>Data Structure</i>	Input, required. The name of the predefined EnterpriseOne data structure to use to load the combo box items from cache (string). This must be the same data structure as the one used to initialize and load the cache. Set the parameter to an applicable object from the object list.
<i>Key Column</i>	Input, required. The cache column containing the key values for the items to be loaded into the combo box (type defined by data structure). Set the parameter to an applicable object from the object list (the list is empty until you select a data structure).
<i>Item Description Column</i>	Input, required. The cache column containing the text descriptions for the items to be loaded into the combo box (type defined by data structure). Set the parameter to an applicable object from the object list (the list is empty until you select a data structure).
<i>Filter Column</i>	Input, optional. The cache column containing the values against which to filter items to be loaded into the combo box (type defined by data structure). The data type must match that of the Key Column parameter. Set the parameter to an applicable object from the object list (the list is empty until you select a data structure).
<i>Filter Value</i>	Input, optional. A value used by runtime to compare against values specified in the <i>Filter Column</i> parameter (string or numeric). The data type must match that of the <i>Filter Column</i> parameter itself. After it retrieves all of the cache records, runtime loads a key/description pair into the combo box only if its <i>Filter Value</i> matches the <i>Filter Column</i> value for that record. Otherwise the retrieved record is ignored and runtime proceeds to process the next retrieved record. Set the parameter to <N/A> or an applicable object from the object list.

Additional Notes

The **Load From Cache** mechanism always retrieves all of the records currently loaded in the cache from the enterprise server to the Web server when the call is made regardless of whether the optional *Filter Column* and *Filter Value* parameters are specified. If you include several combo boxes on a form, you might decide to create a single data structure that loads the cache for all of them. However, you must use discretion in balancing how many records are loaded simultaneously in the cache versus the performance penalties that might result when loading a single combo box with a relatively small subset of the cache.

Remove Item by Index

Description

Use this system function to remove an item from a combo box list by specifying its index position.

Parameters

Parameter	Description
<i>ComboBox Control</i>	Input, required. The combo box FC to affect.
<i>Index</i>	Input, required. The zero-based index position of the item to remove (integer). Set the parameter to an alphanumeric constant (<Literal>) or applicable object from the object list.
<i>Optional</i>	Input, required. A special value indicating whether to remove all items. Set the parameter to <Remove All> or <N/A>.

Additional Notes

You can also clear all items from the list. If the index value specified is invalid and the *Optional* parameter equals <N/A>, no item is removed from the control and an error message is logged.

Remove Item by Key

Description

Use this system function to remove an item from a combo box list by specifying the item's key.

Parameters

Parameter	Description
<i>ComboBox Control</i>	Input, required. The combo box FC to affect.
<i>Key</i>	Input, required. The key of the item to remove (string or math-numeric). Set the parameter to an alphanumeric constant (<Literal>) or applicable object from the object list.
<i>Optional</i>	Input, required. A flag indicating whether to remove all items. Set the parameter to <Remove All> or <N/A>.

Additional Notes

You can also clear all items from the list. If the key value specified is invalid and the *Optional* parameter equals <N/A>, no item is removed from the control and an error message is logged.

Select Item

Description

Use this system function to set one of the items in the combo box as the currently selected item programmatically. The item's description appears in the control just as if the user had chosen it.

Parameters

Parameter	Description
<i>ComboBox Control</i>	Input, required. The combo box FC to affect.
<i>Key</i>	Input, required. The key of the item to set (string or numeric). Set the parameter to an alphanumeric constant (<Literal>), <Null>, or an applicable object from the object list.

Additional Notes

If the key value specified is invalid, the combo box's selection remains unchanged.

Embedded Combo Box System Functions

This chapter describes the system functions unique to the combo box control when it is embedded in a grid.

Add Item

Description

Use this system function to add an item to the combo box list embedded in a grid that is not displaying a list based on a UDC.

Parameters

Parameter	Description
<i>Grid Control</i>	Input, required. The grid FC to affect.
<i>Grid Column</i>	Input, required. The grid column to affect. Set the parameter to an applicable object from the object list.
<i>Grid Row</i>	Input, required. The grid row to affect (integer). Set the parameter to an alphanumeric constant (<Literal>), <Currently Selected Row>, or an applicable object from the object list.
<i>Key</i>	Input, required. The key of the item to add (string or numeric). Set the parameter to an alphanumeric constant (<Literal>) or applicable object from the object list.
<i>Item</i>	Input, required. The name of the item to add (string). Set the parameter to an alphanumeric constant (<Literal>) or applicable object from the object list.
<i>Index</i>	Input, optional. The zero-based index position where the item is to added (integer). If you do not specify this parameter, runtime appends the item to the bottom of the list. The default "-- Select One --" prompt always occupies index position zero. If an invalid index is specified, the addition of the new item fails and an error message is logged. Set the parameter to an alphanumeric constant (<Literal>), <Null>, or an applicable object from the object list.

Additional Notes

Use this system function to add an item to the combo box if it is not displaying a list that is based on a UDC (however, the control itself must be based on a valid DD item). Runtime verifies that the Key and Item parameter values are both specified. If either is missing, the addition of the new item fails and an error is logged. If you try to insert an item having a key value that already exists in the list, runtime will not add the item and will log an error.

The combo box control must reside on the form (as opposed to being embedded in a grid). The control must have a valid data dictionary item associated with it which does not have a related UDC.

Get Description

Description

Use this system function to get the name of an item in a combo box list by specifying its row number.

Parameters

Parameter	Description
<i>Grid Control</i>	Input, required. The grid FC to affect.
<i>Grid Column</i>	Input, required. The grid column to affect. Set the parameter to an applicable object from the object list.
<i>Grid Row</i>	Input, required. The grid row to affect (integer). Set the parameter to an alphanumeric constant (<Literal>), <Currently Selected Row>, or an applicable object from the object list.
<i>Item</i>	Input, required. The object to which to assign the return value that designates the description text of the current selected item (string). Set the parameter to an applicable object from the object list.

Returns

This system function returns the name of a combo box item to the object identified by *Item*.

Get Index of Key

Description

Use this system function to get the zero-based index position of an item in a combo box list by specifying its key. If the key value specified does not exist in the combo box's current set of items, no index value is returned and an error message is logged.

Parameters

Parameter	Description
<i>Grid Control</i>	Input, required. The grid FC to affect.
<i>Grid Column</i>	Input, required. The grid column to affect. Set the parameter to an applicable object from the object list.
<i>Grid Row</i>	Input, required. The grid row to affect (integer). Set the parameter to an alphanumeric constant (<Literal>), <Currently Selected Row>, or an applicable object from the object list.
<i>Key</i>	Input, required. The key of the target item (string or math-numeric). Set the parameter to an alphanumeric constant (<Literal>) or applicable object from the object list.
<i>Index</i>	Input, required. The object to which to assign the return value that designates the zero-based index position of the target item (integer). Set the parameter to an alphanumeric constant (<Literal>) or applicable object from the object list.

Returns

This system function returns the zero-based index position of a combo box item to the object identified by *Index*.

Get Item at Index

Description

Use this system function to get the displayed description of an item in a combo box list by specifying its index position in the list. If the index value specified is invalid, no description is returned and an error message is logged.

Parameters

Parameter	Description
<i>Grid Control</i>	Input, required. The grid FC to affect.
<i>Grid Column</i>	Input, required. The grid column to affect. Set the parameter to an applicable object from the object list.
<i>Grid Row</i>	Input, required. The grid row to affect (integer). Set the parameter to an alphanumeric constant (<Literal>), <Currently Selected Row>, or an applicable object from the object list.
<i>Index</i>	Input, required. The zero-based index position of the target item (integer). Set the parameter to an alphanumeric constant (<Literal>) or applicable object from the object list.
<i>Item Description</i>	Input, required. The object to which to assign the return value that designates the displayed description text of the target item (string). Set the parameter to an applicable object from the object list.

Returns

This system function returns the name of a combo box item to the object defined by *Item Description*.

Get Item Count

Description

Use this system function to determine the total number of items in the combo box list including the “--Select One --” prompt.

Parameters

Parameter	Description
<i>Grid Control</i>	Input, required. The grid FC to affect.
<i>Grid Column</i>	Input, required. The grid column to affect. Set the parameter to an applicable object from the object list.
<i>Grid Row</i>	Input, required. The grid row to affect (integer). Set the parameter to an alphanumeric constant (<Literal>), <Currently Selected Row>, or an applicable object from the object list.
<i>Number</i>	Input, required. The object to which to assign the return value that designates the number of list items. Set the parameter to an applicable object from the object list.

Returns

This system function returns the number of items in the combo box list to the object identified by *Number*.

Get Key at Index

Description

Use this system function to get the key of an item in a combo box list by specifying its index position in the list. If the specified index value is invalid, no key value is returned and an error message is logged.

Parameters

Parameter	Description
<i>Grid Control</i>	Input, required. The grid FC to affect.
<i>Grid Column</i>	Input, required. The grid column to affect. Set the parameter to an applicable object from the object list.
<i>Grid Row</i>	Input, required. The grid row to affect (integer). Set the parameter to an alphanumeric constant (<Literal>), <Currently Selected Row>, or an applicable object from the object list.
<i>Index</i>	Input, required. The zero-based index position of the target item (integer). Set the parameter to an alphanumeric constant (<Literal>) or applicable object from the object list.
<i>Key</i>	Input, required. The object to which to assign the return value that designates the key of the target item (string or math-numeric). Set the parameter to an alphanumeric constant (<Literal>) or applicable object from the object list.

Returns

This system function returns the key of a combo box item to the object defined by *Key*.

Load from Cache

Description

Use this system function to load the combo box control from an EnterpriseOne cache. The control must be based on a DD item which does not have an associated UDC defined for it.

Parameters

Parameter	Description
<i>Grid Control</i>	Input, required. The grid FC to affect.
<i>Grid Column</i>	Input, required. The grid column to affect. Set the parameter to an applicable object from the object list.
<i>Grid Row</i>	Input, required. The grid row to affect (integer). Set the parameter to an alphanumeric constant (<Literal>), <Currently Selected Row>, or an applicable object from the object list.
<i>Cache Name</i>	Input, required. The name of the prepopulated EnterpriseOne cache from which to load combo box items (string). Set the parameter to an applicable object from the object list.
<i>Data Structure</i>	Input, required. The name of the predefined EnterpriseOne data structure to use to load the combo box items from cache (string). This must be the same data structure as the one used to initialize and load the cache. Set the parameter to an applicable object from the object list.
<i>Key Column</i>	Input, required. The cache column containing the key values for the items to be loaded into the combo box (type defined by data structure). Set the parameter to an applicable object from the object list (the list is empty until you select a data structure).
<i>Item Description Column</i>	Input, required. The cache column containing the text descriptions for the items to be loaded into the combo box (type defined by data structure). Set the parameter to an applicable object from the object list (the list is empty until you select a data structure).
<i>Filter Column</i>	Input, optional. The cache column containing the values against which to filter items to be loaded into the combo box (type defined by data structure). The data type must match that of the Key Column parameter. Set the parameter to an applicable object from the object list (the list is empty until you select a data structure).
<i>Filter Value</i>	Input, optional. A value used by runtime to compare against values specified in the <i>Filter Column</i> parameter (string or numeric). The data type must match that of the <i>Filter Column</i> parameter itself. After it retrieves all of the cache records, runtime loads a key/description pair into the combo box only if its <i>Filter Value</i> matches the <i>Filter Column</i> value for that record. Otherwise the retrieved record is ignored and runtime proceeds to process the next retrieved record. Set the parameter to <N/A> or an applicable object from the object list.

Additional Notes

The **Load From Cache** mechanism always retrieves all of the records currently loaded in the cache from the enterprise server to the Web server when the call is made regardless of whether the optional *Filter Column* and *Filter Value* parameters are specified. If you include several combo boxes on a form, you might decide to create a single data structure that loads the cache for all of them. However, you must use discretion in balancing how many records are loaded simultaneously in the cache versus the performance penalties that might result when loading a single combo box with a relatively small subset of the cache.

Remove Item by Index

Description

Use this system function to remove an item from a combo box list by specifying its index position.

Parameters

Parameter	Description
<i>Grid Control</i>	Input, required. The grid FC to affect.
<i>Grid Column</i>	Input, required. The grid column to affect. Set the parameter to an applicable object from the object list.
<i>Grid Row</i>	Input, required. The grid row to affect (integer). Set the parameter to an alphanumeric constant (<Literal>), <Currently Selected Row>, or an applicable object from the object list.
<i>Index</i>	Input, required. The zero-based index position of the item to remove (integer). Set the parameter to an alphanumeric constant (<Literal>) or applicable object from the object list.
<i>Optional</i>	Input, required. A special value indicating whether to remove all items. Set the parameter to <Remove All> or <N/A>.

Additional Notes

You can also clear all items from the list. If the index value specified is invalid and the *Optional* parameter equals <N/A>, no item is removed from the control and an error message is logged.

Remove Item by Key

Description

Use this system function to remove an item from a combo box list by specifying the item's key.

Parameters

Parameter	Description
<i>Grid Control</i>	Input, required. The grid FC to affect.
<i>Grid Column</i>	Input, required. The grid column to affect. Set the parameter to an applicable object from the object list.
<i>Grid Row</i>	Input, required. The grid row to affect (integer). Set the parameter to an alphanumeric constant (<Literal>), <Currently Selected Row>, or an applicable object from the object list.
<i>Key</i>	Input, required. The key of the item to remove (string or math-numeric). Set the parameter to an alphanumeric constant (<Literal>) or applicable object from the object list.
<i>Optional</i>	Input, required. A flag indicating whether to remove all items. Set the parameter to <Remove All> or <N/A>.

Additional Notes

You can also clear all items from the list. If the key value specified is invalid and the *Optional* parameter equals <N/A>, no item is removed from the control and an error message is logged.

Select Item

Description

Use this system function to set one of the items in the combo box as the currently selected item programmatically. The item's description appears in the control just as if the user had chosen it.

Parameters

Parameter	Description
<i>Grid Control</i>	Input, required. The grid FC to affect.
<i>Grid Column</i>	Input, required. The grid column to affect. Set the parameter to an applicable object from the object list.
<i>Grid Row</i>	Input, required. The grid row to affect (integer). Set the parameter to an alphanumeric constant (<Literal>), <Currently Selected Row>, or an applicable object from the object list.
<i>Key</i>	Input, required. The key of the item to set (string or numeric). Set the parameter to an alphanumeric constant (<Literal>), <Null>, or an applicable object from the object list.

Additional Notes

If the key value specified is invalid, the combo box's selection remains unchanged.

CHAPTER 20

Understanding Edit Controls

This chapter describes edit controls.

Edit Controls

An edit control is a text field on a form. All form types can contain edit controls except message forms. Two types of edit controls are available. The first type is commonly referred to as a database field. It is associated with an item in the business view (BV) and through that connection to a specific data dictionary (DD) item. Database fields represent a field in a database record. The second type of edit control is commonly referred to as a DD field, and it also has a connection to a specific DD item.

Within the realm of database fields an additional distinction between filter fields and nonfilter fields exists. Database fields are nonfilter fields by default. Filter fields are used to alter the selection criteria of a database fetch. A filter can have a comparison type of equal, not equal, less than, less than or equal to, greater than, or greater than or equal to. A filter can be marked such that a wildcard (*) displays when the filter is not included in the selection.

The storage for the value of the edit control is based on the type of its associated DD item (such as numeric, string, character). An edit control is affected by the properties of the associated DD item. For example, if an edit control is associated with a DD item of type string with a length of thirty, the edit control will not permit more than thirty characters to be typed into the field.

Edit controls are generic input fields and have no associated text. You must associate edit controls with data items.

If you associate an edit control with a data item from the BV associated with the form, then the value entered by a user at runtime updates the table. If you associate an edit control with a data item not associated with the BV, then the value entered at runtime is for display only. However, you can make a field read-only for BV data items by enabling the Read Only property. You can also associate an edit control with a DD item with a user-defined code (UDC).

Edit controls have a Type Ahead feature. When a user enters a character in the field, the system searches a history list for a match. If a match exists, it appears in the field and is highlighted. This feature is particularly useful for data entry work because it can reduce the amount of typing required. A user can enable or disable Type Ahead editing in EnterpriseOne. Additionally, the Type Ahead capabilities of the browser (such as Internet Explorer) must be enabled as well. Type Ahead is disabled for double-byte languages and multiline edit controls.

As you add controls to the form, you can indicate how the runtime engine filters the incoming records from the database. For example, if the find/browse form has two controls on which you want to filter, the resulting SQL statement that you generate will have 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 \geq and the Search Type control should be $=$. If a user types *D* and puts a *V* in the Search Type field, this is the resulting SQL statement:

```
SELECT * FROM F0101 WHERE (ABAT1 = V AND ABDC LIKE D%) ORDER BY ABAN8 ASC
```

You might also want to apply filters to edit controls 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 filter records that are greater than or equal to the present status and less than or equal to the next status. You drop one value, filter the value, and then drop the next value.

Edit Control Events

These events can fire on an edit control:

- **Control is Entered**
- **Visual Assist Button Clicked**
- **Post Visual Assistant Clicked**
- **Control is Exited**
- **Control is Exited/Changed-Inline**
- **Control is Exited/Changed-Asynch**

Edit Control Runtime Processing

This section describes how the runtime engine processes the edit control.

Control is Entered

The control can acquire focus either through a user action, such as by pressing the TAB key or clicking it with the mouse, or through a system command. When the control acquires focus, runtime fires the **Control is Entered** event.

Control is Exited

When the user leaves the control (such as by pressing the TAB key), the system validates the control and saves the value to memory. A number of events are fired during this process.

Data validation consists of ensuring that the value meets any established criteria for the DD item upon which it is based, including size, type, and so forth. Additionally, runtime compares the current value of the control with the value of the control before it started the exit process. If the value has changed, then it fires two additional events.

This flowchart illustrates the runtime processing that occurs when the control is exited:

Edit Control System Functions

This section discusses system functions for edit controls. The system functions reside in the Control folder

Set Edit Control Color

Description

Use this system function to set the background color for the edit control.

Parameters

Parameter	Description
<i>Control</i>	Input, required. The FC to affect.
<i>Color</i>	Input, required. The color to which to set the control. Double-click <Pick Color> to choose a specific color. Otherwise, set the parameter to <Pick Color> or the default color (<Reset Color>).

Set Edit Control Font

Description

Use this system function to set the font and font characteristics (for example, typeface and color) for an edit control.

Parameters

Parameter	Description
<i>Control</i>	Input, required. The FC to affect.
<i>Font</i>	Input, required. The font and font characteristics to apply. Double-click <Pick Font> to choose specific settings. Otherwise, set the parameter to <Pick Font> or the system default settings (<Reset Font>).

CHAPTER 21

Understanding Grid Controls

This chapter provides overviews of the grid control and its system functions, event rules, and runtime processing and discusses how to configure the grid control at design time.

Grid Controls

A grid control is similar to a spreadsheet. Use grids to display data and to enable users to enter information. Unlike an edit control, grid controls can show multiple data items and multiple table rows at once. You also can use grid controls to enable users to edit table records. In end-user documentation, grid controls are referred to as detail areas.

A grid can either be a browse grid or an update grid. You can use a browse grid for viewing only, and you cannot select individual cells. The find/browse, search & select, power browse, and portlet browse forms have browse grids, as do browse subforms.

You can use an update grid to add, delete, or update records. Cells in an update grid can be selected individually. The header detail, headerless detail, power edit, and portlet edit forms have update grids, as do update subforms.

Grid controls of both types contain columns. The columns are specified at design time and are one of these types:

- Database column or
- Data dictionary (DD) column.

A database column is associated with an item in the business view (BV) and through that connection to a DD item. Database columns represent a field in a database record.

Although only one type of column is referred to as a DD column, both types have a connection to a specific DD item. The difference is that a database column has the additional connection to a BV field. A grid column is affected by the properties of the associated dictionary item. For example, if a grid column is associated with a dictionary item of type string with a length of 30, that grid column will not permit more than 30 characters to be typed into the cell.

Grid controls can also have a query-by-example (QBE) line. The QBE columns have a one-to-one correspondence with the grid columns. You use a QBE value to change the selection criteria of a database fetch. Only database columns enable entry in the QBE columns because the purpose of the QBE is to affect the selection and only database columns are in the BV. A QBE column can have one of these comparison types:

- Equal
- Not equal
- Less than

- Less than or equal to
- Greater than
- Greater than or equal to.

The comparison type is equal unless you specify otherwise. You can specify the comparison type in the QBE column or by using system functions. You can use wildcards (* or %) for an inexact search on a string field.

Grid values can be retrieved on a cell-by-cell basis or on a row-by-row basis. Internal storage for the grid columns exists in the interactive engine. The way the grid column value is stored is based on the type of the associated dictionary item (for example, math numeric, string, character), and it is distinct from the screen representation of the value. Only one row of data can be acted upon at any given time. Each event executes in the context of a specific row.

Grid Control Design-Time Considerations

Grid columns can have a heading row and might also have a QBE row to facilitate user searches. Additionally, you can specify rows or columns to display information that you acquire or calculate during runtime. Typically, the rest of the rows contain data. Generally, this data comes from database tables.

Usually, when you make the grid control, you place data items in the grid. The data items can come from the BV attached to the form, or they can be based on DD items. Either way, each data item becomes a column. Just as the grid control itself has a variety of property settings, you can set property values for each column in the grid as well.

You have a great deal of control, both at design time and at runtime, over how the grid appears, what it displays, and how it functions.

This section discusses how to:

- Design the grid.
- Add columns to the grid control.
- Set property values for the grid control.
- Show multiple currencies per column.

Designing the Grid

This task outlines the overall process for creating a grid.

To design the grid:

1. Create a form.
Find/Browse, header detail, headerless detail, and search & select forms have a grid control on them automatically.
Power edit, power browse, reusable and embedded edit subforms, reusable and embedded browse subforms, edit portlet, and browse portlet forms will accept a grid control.
2. On the form level, attach a BV for the engine to use to populate the grid or use values assigned in ER and system functions to insert grid rows.
3. Drop a grid control onto the form.

4. Configure the control.

If you are working with a header detail form, you can attach a different BV to the control.

5. Add columns to the control.
6. Configure each column.
7. Add logic.

Adding Columns to the Grid Control

To add columns to a grid control:

1. Perform one of these tasks:
 - To add a data item to the grid from the BV associated with the form, double-click a data item in the Business View Columns Browser.
 - To add a data item to the grid that is not in the BV associated with the form, search for a data item in the Data Dictionary Browser, and then double-click it.

Each time you double-click an item, Form Design Aid (FDA) adds it as a column to the grid control.

2. Repeat step one until you have added all of the columns you need.
3. Double-click the grid control.
4. On Grid Properties, click the Columns tab and arrange the column order by choosing a column and then using the Up and Down buttons to shift its position in the list.
5. Click the Sort Order tab and set the order in which the system will sort returned records by performing these steps:
6. Choose a data item in the Unsorted Columns list and click the right arrow. Repeat for each data item on which you want to sort.

The data item moves to the Sorted Columns list.

7. To toggle between displaying by ascending and descending order, click the data item in the Sorted Columns list.

The letter A next to the data item indicates that the system will display records in ascending order. The letter D next to the data item indicates that the system will display records in descending order.

8. Arrange the column sort order by choosing a column in the Sorted Columns list and then using the Up and Down buttons to shift its position in the list.

Setting Property Values for the Grid Control

Use design-time settings in FDA to affect grid display (including showing multiple currencies per column), loading and processing, and how a user can enter data.

In the next subsections, it is assumed that you access all property values from the Property Browser in FDA.

Grid Control Display

You control grid appearance through a combination of property values that you set at design time and system function values you program to occur during runtime. Consider carefully the design-time settings you make because you may not be able to change them during runtime.

Desired Result	To Implement	Notes
enable users to click a column value.	Select the column and set Clickable to <i>Yes</i> .	Runtime fires the Grid Column Clicked event when the user selects the column value.
Change the appearance of the grid based on HTML code.	Select the grid control, set Use Alternate Grid Format to <i>Yes</i> , and paste the HTML formatting commands to use the render the grid in Alternate Grid Row Format String.	
Disable the QBE row.	Select a column and set Disable QBE to <i>Yes</i> .	You can only disable the QBE on a column-by-column basis. Alternatively, you can hide the entire row with the grid property, Hide Query By Example.
Display multiple currency types in a single column.	Select a column and set Support Multiple Currencies to <i>Yes</i> .	
Hide a column if currency is disabled.	Select a column and set No display if currency is OFF to <i>Yes</i> .	
Hide the row selector cell at the beginning of each row (check box or radio button) with which shows which rows have been selected.	Select the grid control and set Hide HTML Row Selector to <i>Yes</i> .	
Hide the Query-By-Example (QBE) row.	Select the grid control and set Hide Query By Example to <i>Yes</i> .	If you hide the QBE row, not only do you prevent user interaction, but you also prevent program interaction. You cannot use system functions to work with a hidden QBE row. Alternatively, you can disable QBE cells on a column-by-column basis with the column property, Disable QBE.
Hide the row headers.	Select the grid control and set Hide Row Numbers to <i>Yes</i> .	
Open the form in update mode.	Select the grid control and set Update Mode to <i>Yes</i> .	
Override the grid column header.	Select the column and enter the text in Column Header One and Column Header Two that you want to appear.	Column Header One is the first line and Column Header Two is the second.
Prevent users from customizing the grid.	Select the grid control and set Display Customize Grid to <i>No</i> .	When disabled, the system will display neither the "Customize Grid" link on the grid header nor the "Grid Formats" menu item.

Desired Result	To Implement	Notes
Reorder the columns in the grid.	Select the grid control, select Column Order, and launch the Grid Properties form. Choose the column to move and then click the Up or Down buttons to change its location relative to the other columns.	
Set the number of rows to display at a time.	Select the grid control and set Grid Row Count to the number of rows you want to display at a time.	The JDE.ini property, GlobalPageSize also controls the number of rows to display at a time. Runtime uses whichever value is greater.
Wrap text in the column	Select the column and set Wrap Text to <i>Yes</i> .	When disabled, the system truncates text that exceeds the column width.

In addition to these property settings, use these system functions to affect grid appearance:

- **Set Grid Color**
- **Set Grid Font**
- **Set Grid Row Format**

Loading and Processing Behavior

FDA provides some options to automate fetches. You can choose to load the grid based on the fetch logic when the user launches an application if the grid resides on a find/browse or headerless detail form. Use automatic loading options with care; in some cases, enabling these features can have a significantly negative impact on performance.

Especially in HTML environments, the page-at-a-time processing feature can enhance performance. This option enables you to populate the grid at the rate of one page at a time (the runtime engine only loads enough data to fill the grid, and then loads enough data to fill the grid again if the user clicks Next). Page-at-a-time processing is enabled by default, and the PeopleSoft standard is to leave it enabled for all form types.

Desired Result	Navigation	Notes
enable users to select multiple rows.	Select the grid control and set Multiple Select to <i>Yes</i> .	
Change the sort order.	Select the grid control, select Column Sort Order, and launch the Grid Properties form. Move columns to the right that you do want to sort on.	
Load a different data item.	Select a column, select Data Item Information, and launch the Grid Column Properties form. Select the <i>Data Items</i> tab and select the data item to use.	

Desired Result	Navigation	Notes
Load data at the rate of one grid page at a time (page-at-a-time).	Select the grid control and set Disable Page-at-a-Time Process to <i>No</i> .	
Load data automatically upon changes in a related (child) form.	Select the grid control and set Auto Find On Changes to <i>Yes</i> .	Use this option on forms that have no modeless form interconnections. only.
Prevent users from importing or exporting files.	Select the grid control and set one or more of these properties to <i>No</i> : <ul style="list-style-type: none"> • Display Import from Excel • Display Export to Excel • Display Export to Word 	If a property is disabled, the system will not display the related option icon or menu item.
Process all the grid rows on a database commit, not just the ones that changed.	Select the grid control and set Process All Rows in Grid to <i>Yes</i> .	When enabled, this property causes runtime to apply row changed and row exited logic to all rows, no matter their state.

Data Entry Behavior

These properties affect how the user interacts with the grid.

Desired Result	Navigation	Notes
Post rows silently to improve application performance if you are running in low interactivity mode.	Select the grid control and set Multi-Line edit to <i>Yes</i> .	When enabled, runtime posts rows in groups of three to five to the database in the background. When disabled, in low interactivity mode, runtime posts each time the user tabs out of the row, forcing the user to wait for a refresh before continuing.
enable users to sort on a column.	Select the column and set Sortable by end user to <i>Yes</i> .	When enabled, users can choose to sort grid rows in ascending or descending order based on this column.
Make a column a check box column.	Choose the column and set Display Style-Default or Checkbox to <i>Checkbox</i> .	The Column Selection Changed event fires when the user selects or clears a check box in a check box column.
Prevent user from adding lines to a grid.	Choose the grid control and set No Adds on Update Grid to <i>Yes</i> .	Although they cannot add new records, users can still edit existing ones.
Prevent users from entering data.	Choose the column and set Disabled to <i>Yes</i> .	
Require the user to input a value into a column.	Choose the column and set Required entry field to <i>Yes</i> .	

Showing Multiple Currencies per Column

When a column has the Support Multiple Currencies option enabled, the runtime engine assumes that each cell contains its own currency setting, and it formats each cell based on that cell's currency decimal setting. The runtime engine will not apply the currency settings to other grid rows, however. Therefore, the application needs to apply currency to each grid row individually. For example, the Amount column in row 1 might have a JPY currency type and be formatted with no decimals, while the Amount column in row 2 might have a USD currency type and be formatted with two decimals.

When a column has the Support Multiple Currencies option disabled, the runtime engine assumes that all of the cells in that column share the same currency setting, and so it applies that currency setting to other grid rows. Therefore, if you specify the currency setting in one row, the system overwrites the currency setting for all the other rows in the grid to match. This feature offers a performance benefit for those grids that contain only one currency because the application needs to specify a currency setting to one grid row only to affect the entire grid.

These currency rules apply:

- When assigning values using conventions such as target = source, if the source object does not have any currency information (currency code = null or empty string), then the target object keeps its own currency.
- When a GB object is cleared, the currency code and currency decimal information for that column is not cleared.

Grid Control Events

This section discusses the events that runtime fires while processing a grid control.

Depending on the type and mode of the form, runtime fires a variety of events in response to events regarding the grid control.

If you set up a custom fetch routine, you will want to make use of **Get Custom Grid Row**. To perform a custom fetch, attach logic to this event to fetch a single row. To indicate that there are more records to fetch, use the system function, **Continue Custom Data Fetch** (set it to True).

When runtime loads the grid control, it fires **Add Last Entry Row to Grid** when (and if) it adds an entry row as the last line in the grid.

The pattern for adding and updating records is to fire an event immediately before and after commit. Then, runtime fires another event after all the records have been processed. This is a list of applicable events:

- **Add Grid Rec to DB - Before**
- **Add Grid Rec to DB - After**
- **All Grid Recs Added to DB**
- **Update Grid Rec to DB - Before**
- **Update Grid Rec to DB - After**
- **All Grid Recs Updated to DB**

The pattern for deleting a row is similar except that runtime notifies you before and after the user action as well as the commits:

- **Delete Grid Rec Verify - Before**

- **Delete Grid Rec Verify - After**
- **Delete Grid Rec from DB - Before**
- **Delete Grid Rec from DB - After**
- **All Grid Recs Deleted from DB**

Runtime notifies you of these user actions:

- **Column Selection Changed**

This is an event applies to check box grid columns only. **Column Selection Changed** fires if the user selects or clears an editable check box grid cell. It does not fire if the grid cell is modified by event rules.

- **Grid Column Clicked**

This event fires only if the grid column was configured as being clickable during design time.

- **Visual Assist Button Clicked and Post Visual Assist Clicked**

If you want to override the default UDC form that appears automatically in response to clicking the Visual Assist, these are the events to which to add the logic.

- **Double Click on Row Header**

Finally, runtime signals as the user works through the grid, starting with when the grid gets focus and ending when the grid loses focus.

- **Set Focus on Grid**

This event fires whether the user or runtime changes the focus.

- **Column Selection Changed**

This event fires only if the column is a check box column (that is, the column Display Style-Default or Checkbox property has been set to *Checkbox*).

- **Row is Entered**

- **Row is Selected (Web Only)**

This event fires on power forms and subforms only.

- **Row is Exited**

This event fires whenever a row is exited.

- **Row Exit & Changed - Inline**

This event fires after **Row is Exited** if the grid is an update grid and the row has been changed since the last time the row was exited.

- **Row is Exited & Changed - Asynch**

This event fires after **Row Exit & Changed - Inline** if the grid is losing focus, or if another row is entered. **Row is Exited & Changed - Asynch** is equivalent to validating the contents of the row. It is often used for the Edit Line master business function.

- **Kill Focus on Grid**

This event fires whether the user or runtime changes the focus.

Grid Control Runtime Processing

This section discusses runtime processing for the grid control.

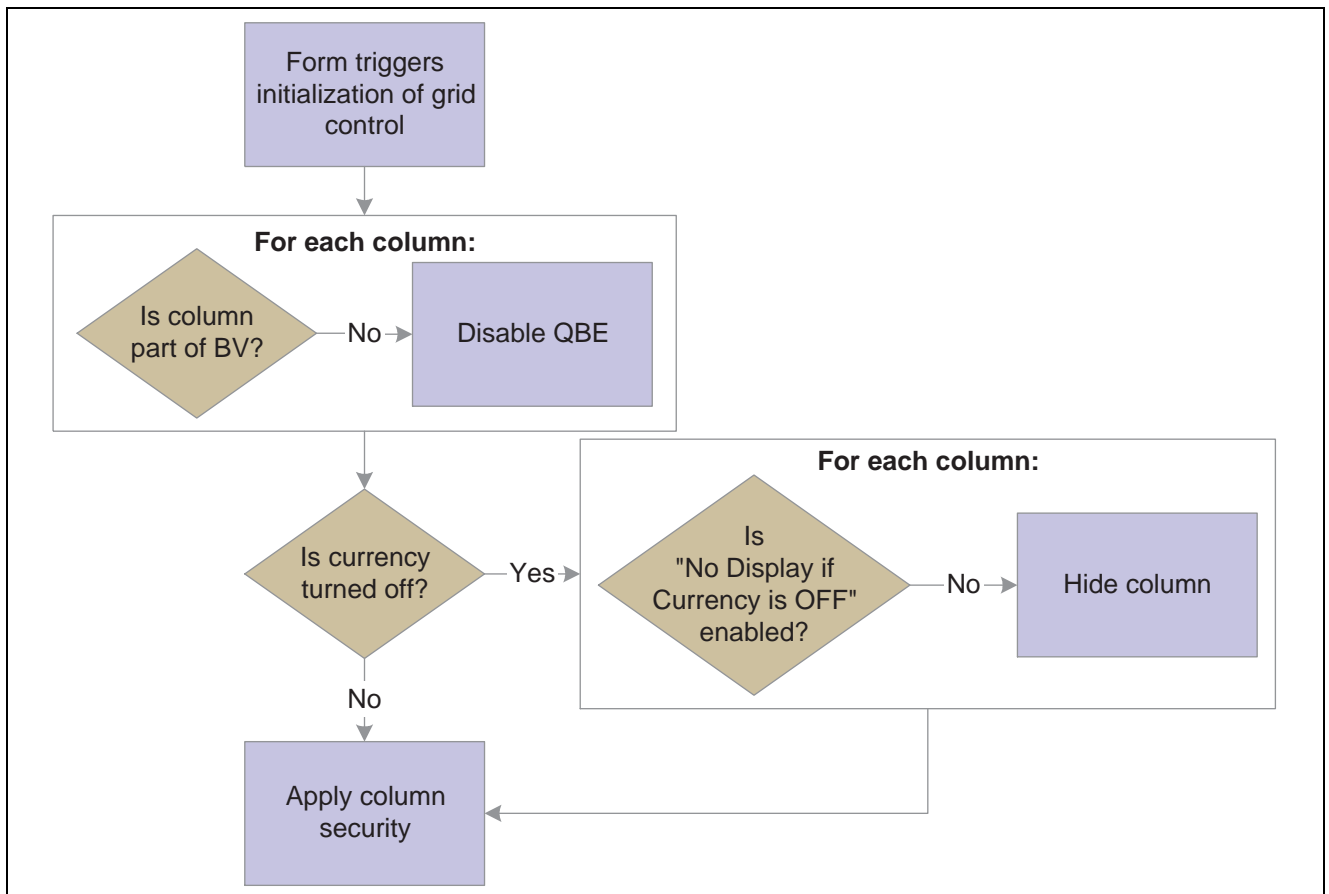
How Runtime Processes the Grid Control

This subsection discusses how runtime processes the grid control.

Typically, runtime processes the grid control at these points in a standard workflow:

- On control initialization (triggered by form initialization).
- On grid population request (triggered conditionally by control initialization, Next, and Previous, and always by Find).
- On row or control exit.

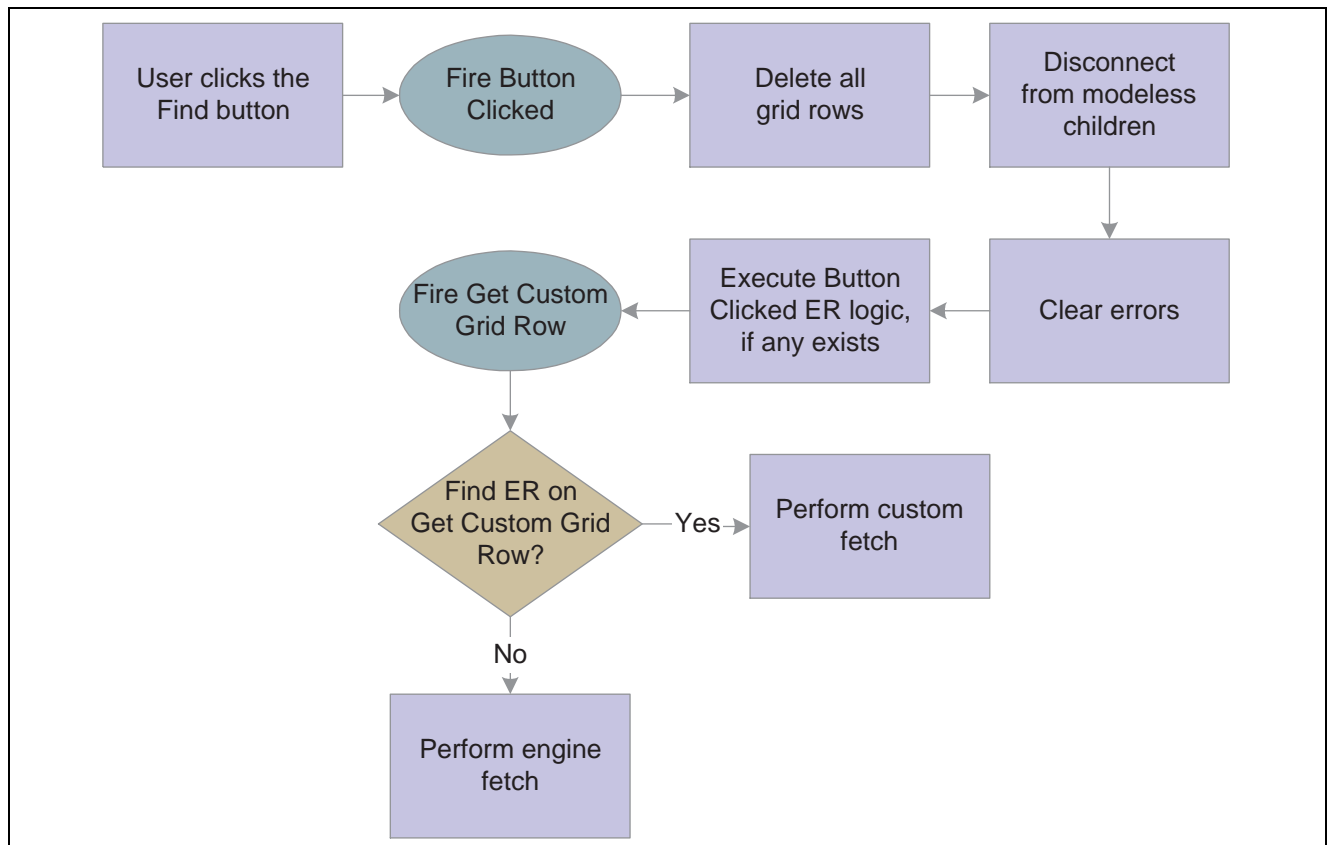
This flowchart illustrates the initialization process for a grid control:



Grid control initialization

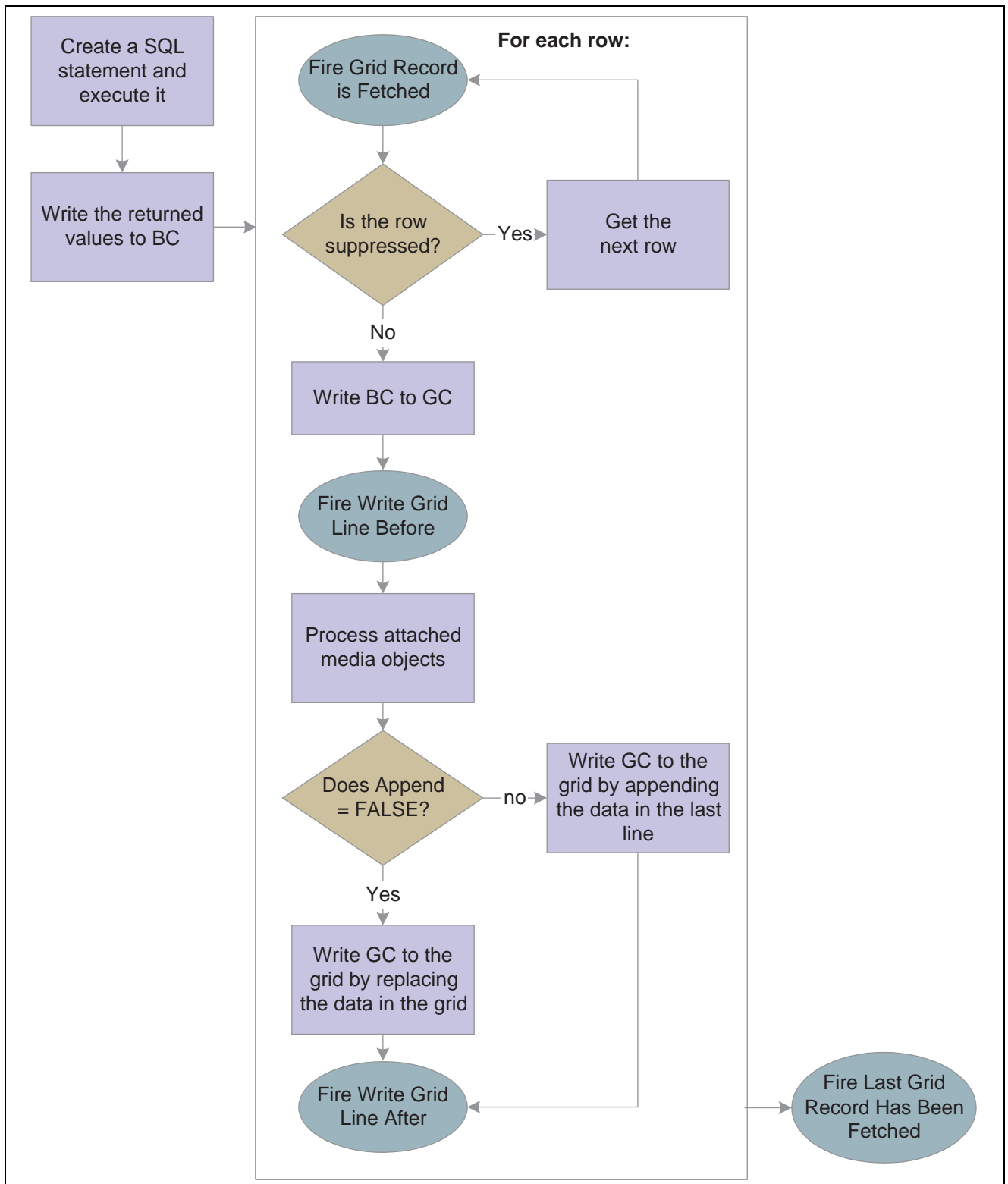
Grid population occurs immediately after the initialization process if the option, Automatically Find On Entry is selected. The grid population process is also triggered when the user clicks the Find button or if you apply logic on an event to do so. If the fetch requires access to the form's BV only, then you can probably enable the engine to perform the fetch on its own. However, if you want to access other tables, you must set up and execute a custom fetch programmatically.

This flowchart illustrates the steps that occur when find is initiated:



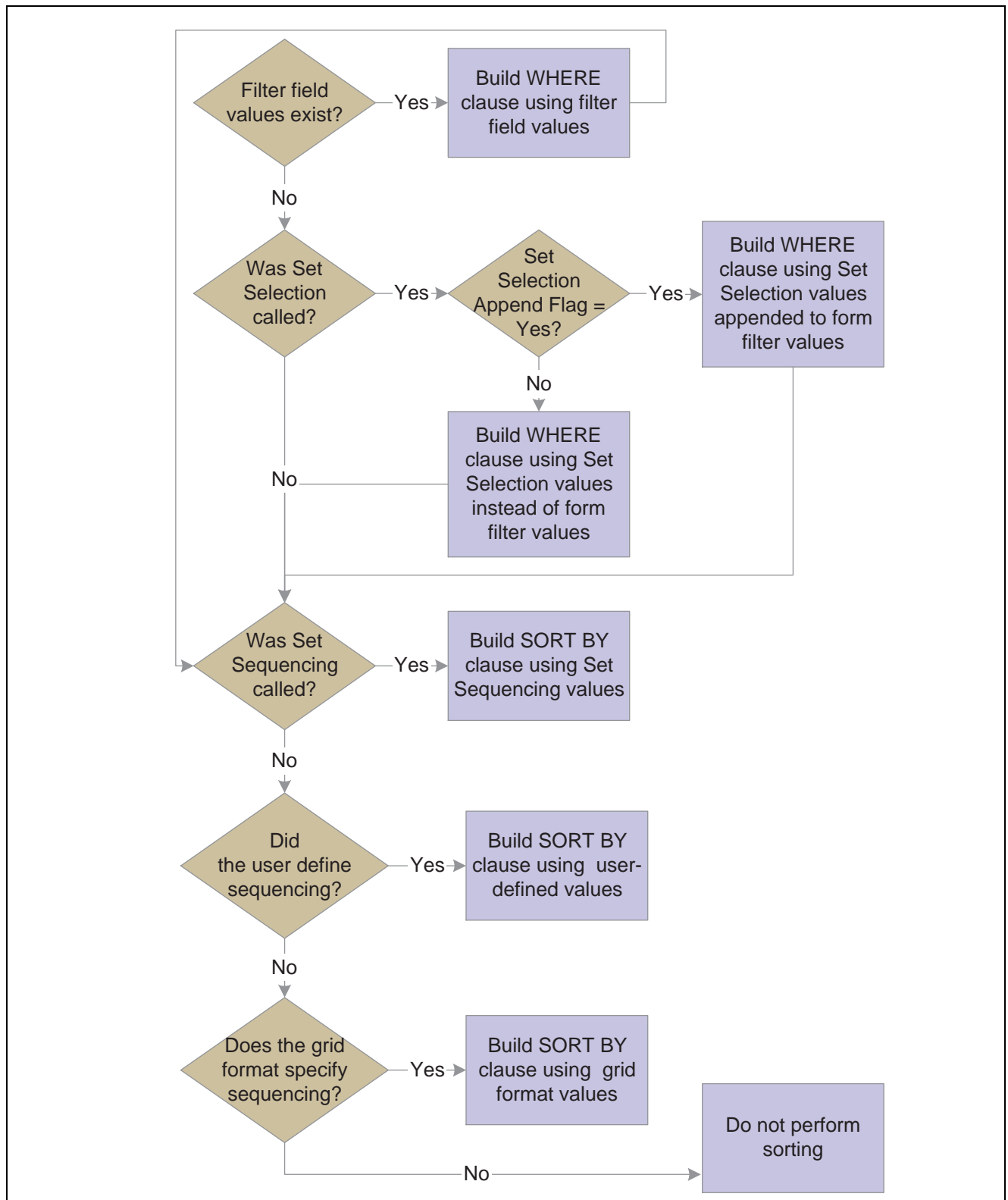
Grid control find button processing

This flowchart illustrates how the system performs an engine fetch:



Grid control fetch processing

This flowchart illustrates how the system creates the SQL statement:



Grid control SQL processing

Custom fetches are executed based exclusively on the ER associated with the **Get Custom Grid Row** event. The ER provides instructions for how to load a row, and runtime repeats the ER until you tell it to stop or it reaches the end of a page if page-at-a-time processing is enabled. To tell the engine to continue looping or not, include a call to **Continue Custom Fetch** in the ER. Set **Continue Custom Fetch** to True to run another loop; set it to False to exit the custom fetch loop. (Not calling **Continue Custom Fetch** is equivalent to setting it to false.)

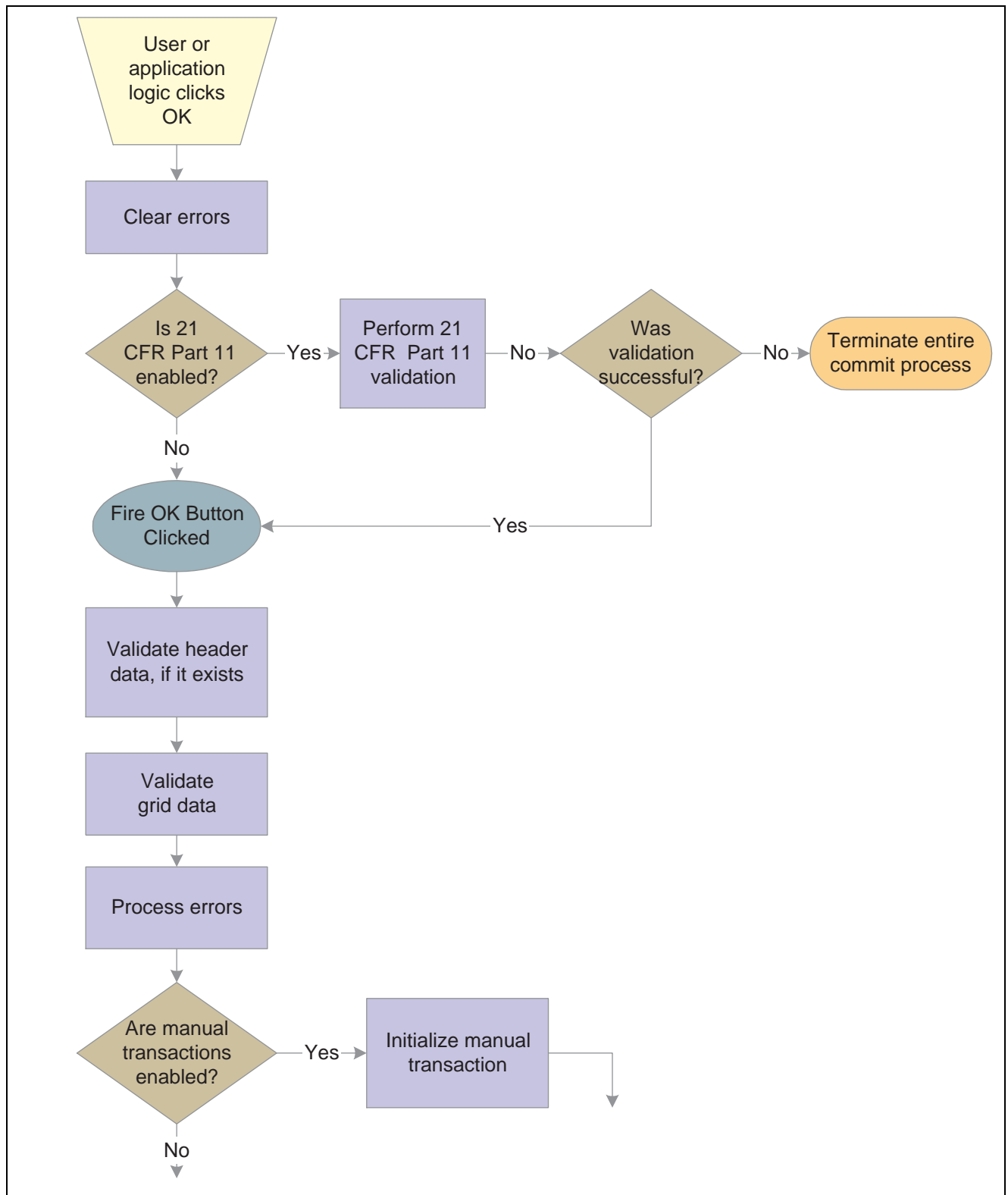
Grid controls are used to display information. In some instances, information display is all the grid does, in which case, no more grid control processing occurs.

Grids can also be used to enable users to delete, edit, and add data as well. To enable any of these functions, the grid (and therefore the form) must be in Add or Update mode (Update mode is the default mode; add mode occurs either when the user clicks Add or a fetch fails to retrieve data). Furthermore for add and update capabilities, the grid must not be disabled; that is, the Disable Input option was not set at design time. Finally, to add data, the user must have an entry row (that is, a blank row) as well. Runtime automatically adds an entry row to update grids in the add and update modes as long as the No Adds on Update Grid option was not set at design time.

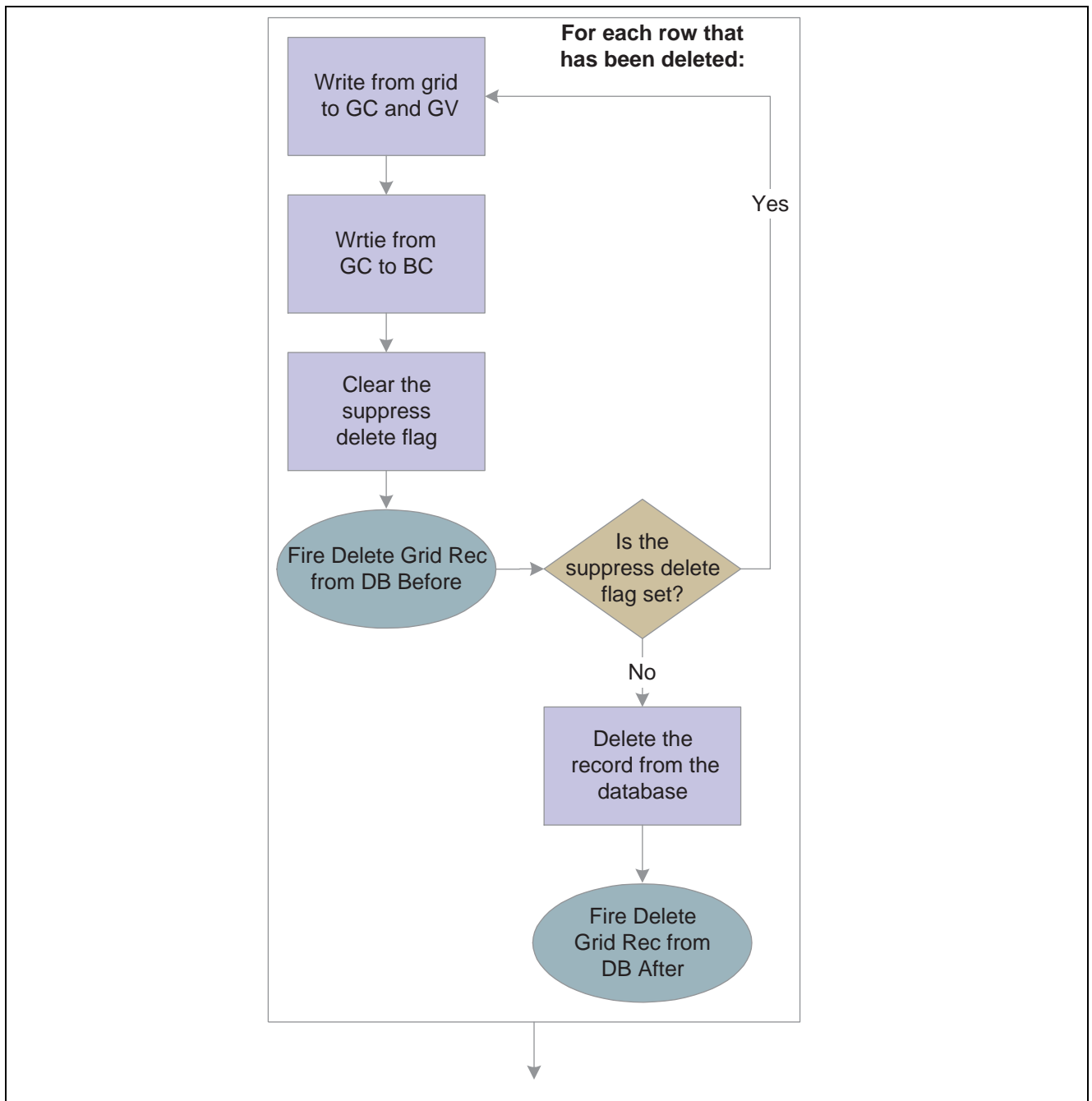
When the user tabs out of an entry row, runtime performs these functions in this order:

1. Clear the GC hash table.
2. Fire **Last Entry Row to Grid**.
3. Get the value from the GC and copy the contents into a new entry row.

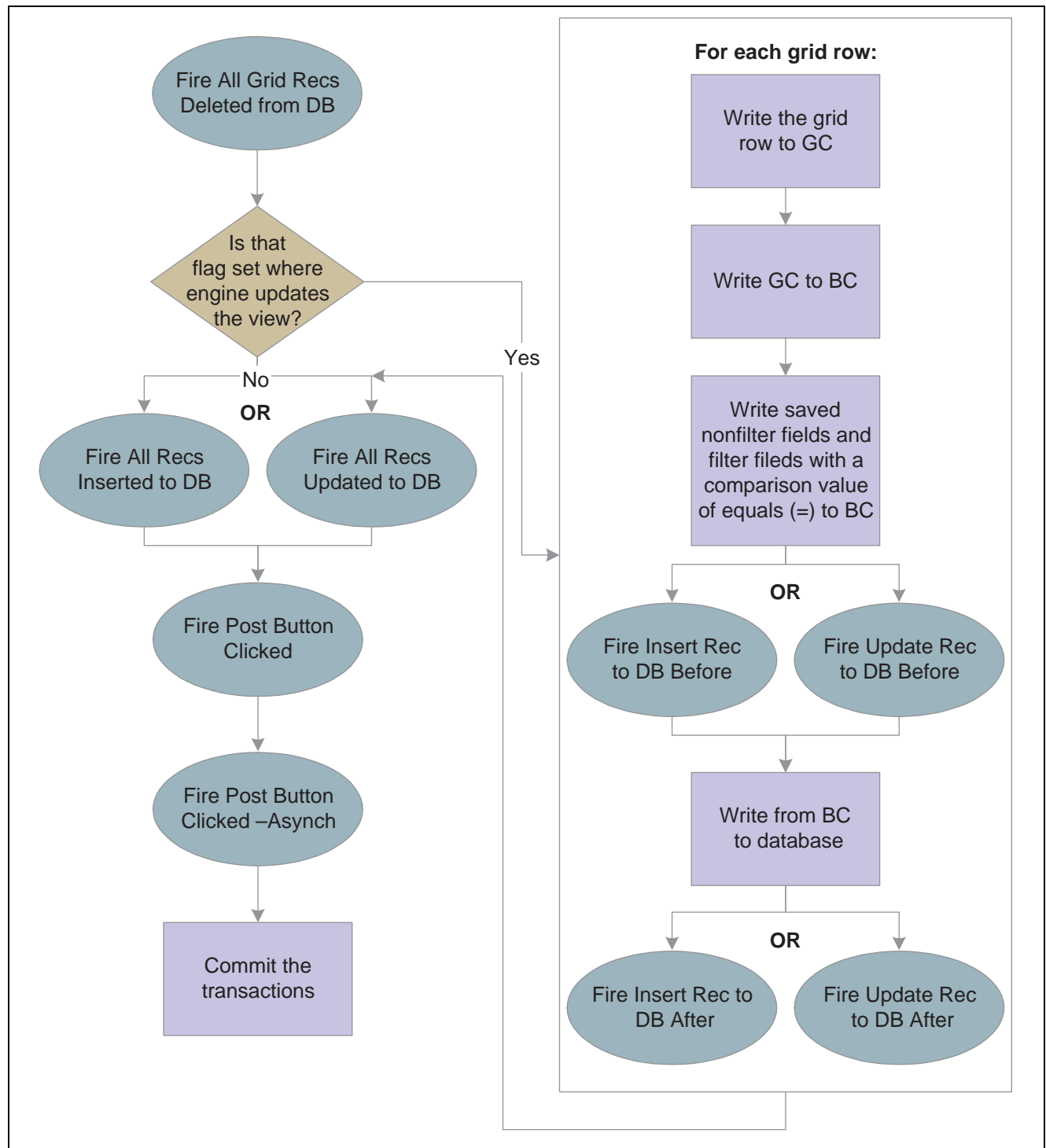
This three-part flowchart illustrates how runtime processes the act of writing data from the grid and committing the change to the database:



Grid control database commit, part 1 of 3

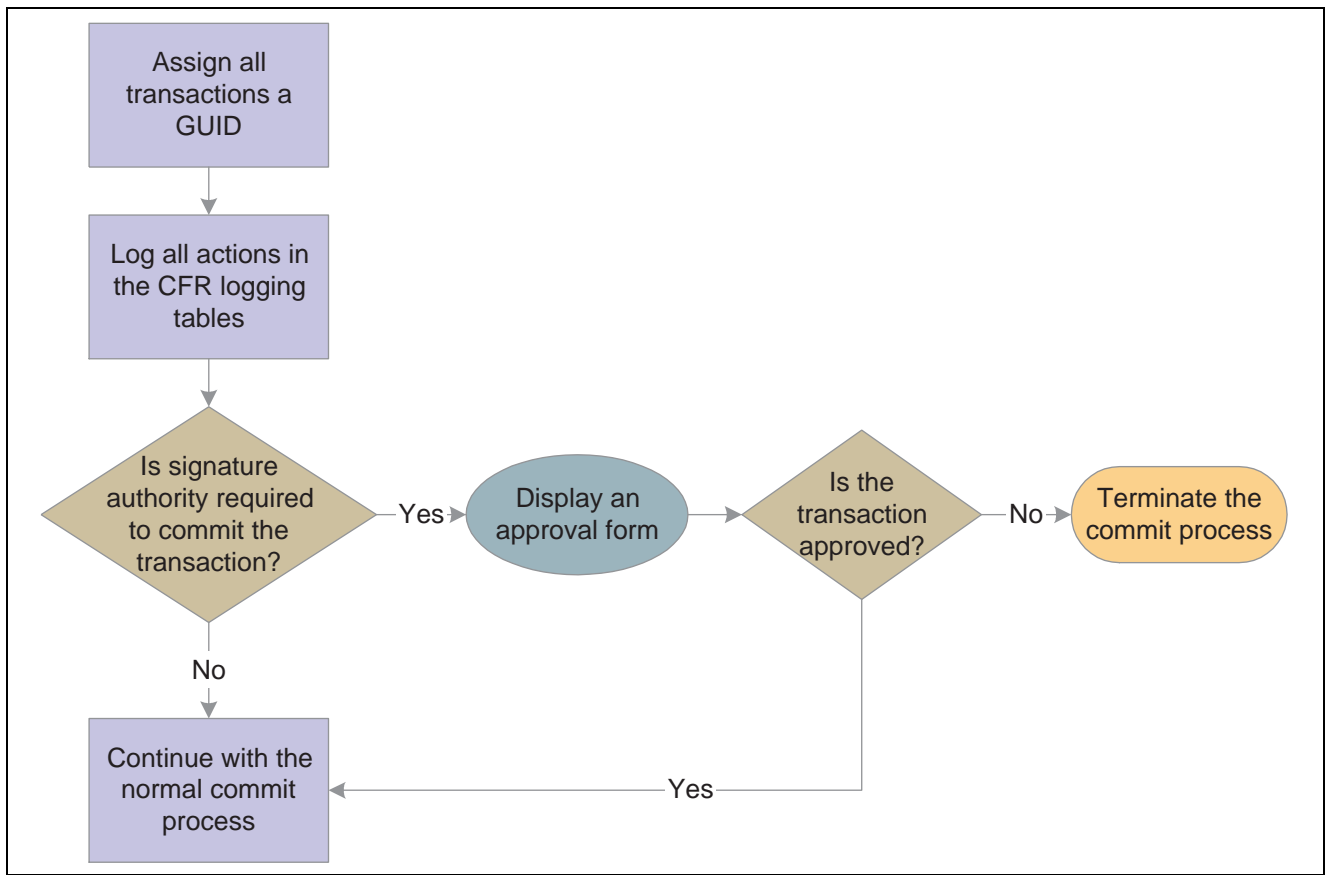


Grid control database commit, part 2 of 3



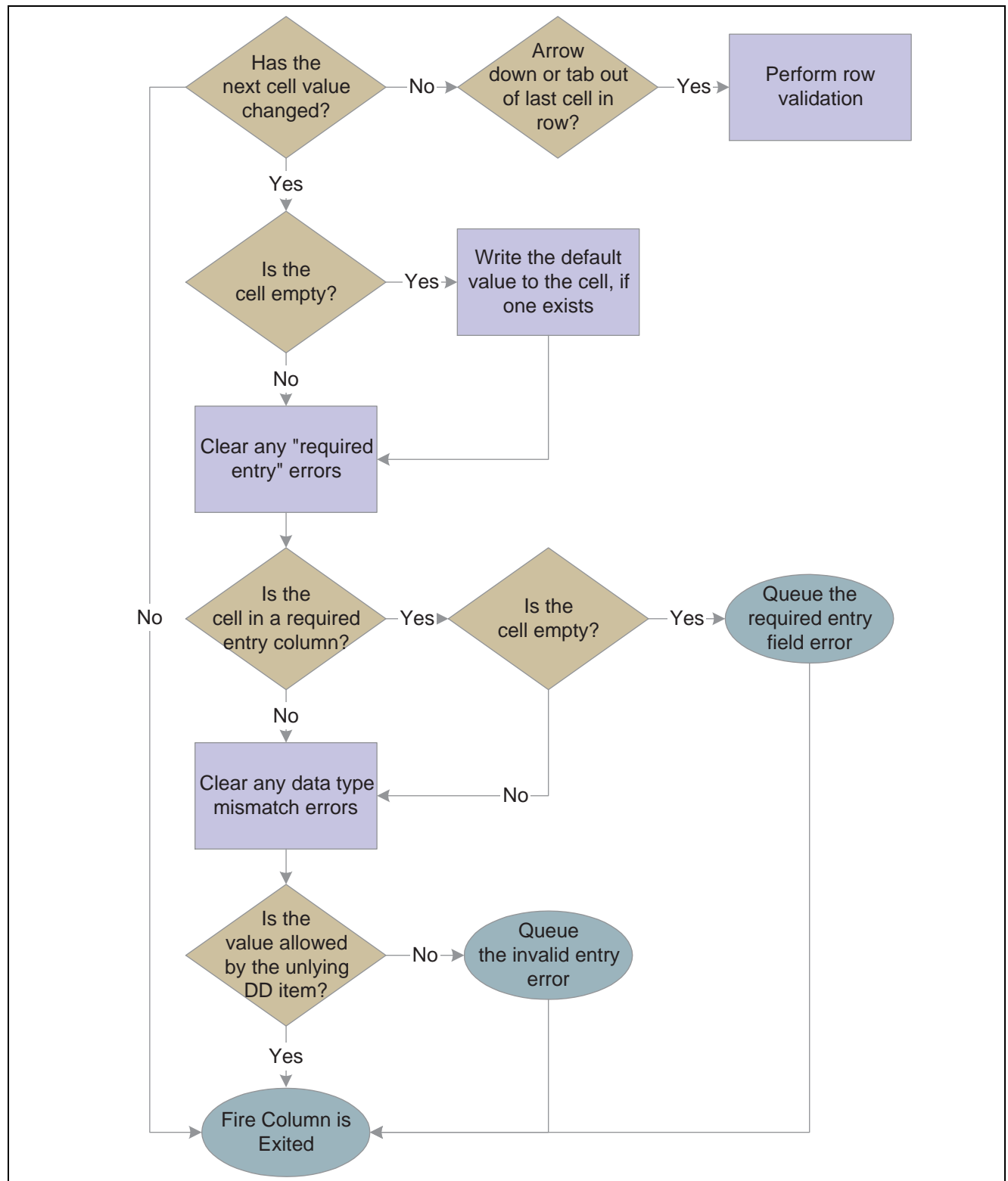
Grid control database commit, part 3 of 3

This flowchart illustrates 21 CFR 11 validation:

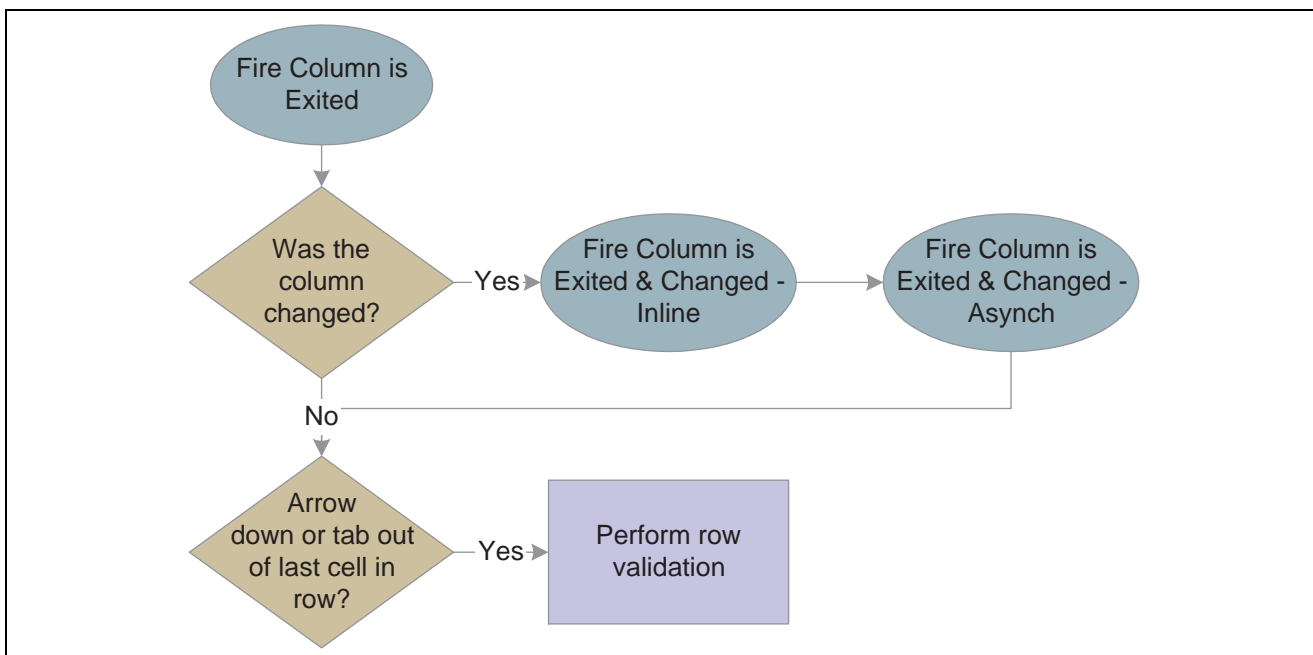


Grid control 21 CFR validation

This two-part flowchart illustrates grid data validation:



Grid control data validation, part 1 of 2



Grid control data validation, part 2 of 2

Impact of Interactivity Levels

This subsection discusses the effect of different runtime processing environments and modes: HTML (low or high interactivity) or Windows.

This table contrasts the differences:

Grid Type	Interactivity	Effect
Non-editable	Low	Refresh the entire form (including grid control) on Find, Next, and Previous. Display only one page of data in the grid.
Non-editable	High	Refresh only the grid (not the entire form) on Find. Append additional rows and expand the page size on Next. The interface provides a vertical scroll bar.
Non-editable	Windows	Refresh only the grid (not the entire form) on Find.

Grid Type	Interactivity	Effect
Editable	Low	Display only one page of data in the grid. Post and refresh behavior depends on whether the Multi-Line Edit option is enabled: <ul style="list-style-type: none"> Multi-Line Edit option disabled: Post in-line on row exit and refresh the whole form. Multi-Line Edit option enabled: Post asynchronously and refresh the grid rows every 3-5 row exits.
Editable	High	Expand the page and add a vertical scroll bar to display all rows. Post asynchronously and refresh the grid control at every user action (row exit, column exit, and so forth). Add a new entry row upon user keying into and exiting any cell in the entry row.
Editable	Windows	Expand the page and add a vertical scroll bar to display all rows. Process every user action immediately. Add a new entry row upon user keying into any cell in the entry row.

For high interactive editable grid, post asynchronously for each cell change.

Grid Control System Functions

This section discusses the system functions you can use to affect how the grid control works during runtime.

Grid system functions enable you to manipulate the appearance of the grid and what goes into or comes out of it during runtime. In the latter case, the system applies the command either to the model version of the grid or to the display version. The model version of the grid contains all rows and columns, whereas the display version might have hidden rows or columns. This is an important distinction to make when you are performing actions based on row or column number. For example, assume a grid contains three rows, but the second one is hidden. If you insert a row beneath row number two in the model grid, the insert row becomes row three, and row three becomes row four. However, if you insert a row beneath row number two in the display grid, the row is inserted at the bottom of the grid, because the actual row number two is hidden.

Another distinction to make is whether the system function expects the first grid row number to be zero or one. All system function descriptions where these factors might be an issue include whether they affect the model or display version of the grid and whether they are zero- or one-based.

Some grid-related system functions enable you to alter the appearance of the grid. Use **Set Grid Color** to change the background color of a cell, row, column, or the entire control. Use **Set Grid Font** to change the type, style, size, effects, and color of the font in a cell, row, column, or the entire control. Alternatively you can use **Set Grid Row Format** to use HTML formatting commands to control the appearance of the grid, not only for color and font, but for other factors as well.

To enable or prevent a user from customizing the grid, use **Display Customize Grid Option**.

Use **Hide Grid Column** and **Show Grid Column** to hide and display columns in the grid. Although a user cannot affect a hidden column, you can still work with and change values in it programmatically.

If the column is based on a DD item, use **Set Data Dictionary Overrides** to change the DD attributes of the column, or **Set Data Dictionary Item** to substitute another DD item altogether. To change the heading for the column, use **Set Grid Column Heading**.

Use **Hide Grid Row** and **Show Grid Row** to hide and display rows in the grid. As with columns, you can still affect hidden grid rows programmatically.

You can also control the bitmap icon that appears next to a given row with **Set Grid Row Bitmap**.

You can enable or prevent users from changing an existing cell, row, column, or the entire control. Use **Enable Grid** to enable user edits (assuming that the current mode permits such actions) and **Disable Grid** to prevent them. Although users cannot alter a disabled grid object, you can change it programmatically.

You can enable or prevent users from importing from Excel or from importing or exporting to Excel or Word with **Display Import from Excel Option**, **Display Export to Excel Option**, and **Display Export to Word Option**.

You can determine if a grid cell has been updated since the last check with **Was Grid Cell Value Entered**.

Use **Get Max Grid Rows** to count the number of rows currently loaded into the grid control. Use **Get Selected Grid Row Count** to count the number of rows in the current selection. The selected area need not consist of contiguous rows.

Use **Set QBE Column Compare Style** to set the comparison value (=, >, <=, and so forth) for a QBE column. **Clear QBE Column** clears an entry in a QBE column.

Runtime always fetches all records for all columns defined by the BV. However, you can impact how runtime builds the WHERE and ORDER BY clauses of the SQL SELECT statement it executes to perform a fetch. (Use the various Hide system functions to filter what the user sees even more, if necessary.) Use **Set Selection** to input the specific values you want to use in the WHERE clause. For example, say you wanted a list of accounts that are greater than 1000 USD. The SQL WHERE statement might look like this (literal values are used to simplify the examples):

```
WHERE F0902.GBAN01 > 1000
```

The **Set Selection** statement that builds this code would look like this:

```
Set Selection(FC Grid, F0902, GBAN01, <Greater Than>, '1000', <None>)
```

That is: (grid control, table, alias, comparison type, comparison value, and/or).

The values you input with **Set Selection** are persisted until form close. Consequently, subsequent calls to **Set Selection** enable you to build a complex WHERE clause. For example, say you wanted a list of document types for Foundation. In the F0005 table, system code and object type are in two different columns, so you must call **Set Selection** twice. First:

```
Set Selection(FC Grid, F0005, DRSY, <Equal To>, '00', <None>)
```

Then:

```
Set Selection(FC Grid, F0005, DRRT, <Equal To>, 'DT', <AND>)
```

By setting the and/or parameter to **<AND>**, the system knows to append the WHERE clause with the values from a subsequent call to **Set Selection** with an AND. Therefore, runtime will build this WHERE clause:

```
WHERE F0005.DRSY = '00' AND F0005.DRRT = 'DT'
```

Use **Set Selection Append Flag** to indicate whether the criteria from the **Set Selection** system function should replace the criteria from existing filter fields and QBE or be appended to it.

You can alter the sort order of the returned values by using **Set Sequencing**. **Set Sequencing** builds the ORDER BY clause of the SQL statement. The **Set Sequencing** parameters are: Set Sequencing (grid control, table, alias, sort order), so if we wanted to sort the preceding example by the document type code (DRKY) in ascending order, you would code:

```
Set Selection(FC Grid, F0005, DRKY, <ASCENDING>)
```

If you need to build a WHERE clause and do not want to use values that may already exist because of previous uses of **Set Selection**, use **Clear Selection** to delete all existing selection (but not sequencing) values from memory. To remove sequencing values, use **Clear Sequencing**.

When you fetch data, you can populate GB and then use system functions to write GB values to the grid control. You can also write from the grid control to the GB and manipulate the GB and grid data in other ways.

To write from the GB to the grid, use either **Insert Grid Buffer Row** or **Update Grid Buffer Row**, depending on whether you want to add a new row to the grid (insert) or overwrite an existing row (update). In either case, when you write a new row, you have a variety of configuration options. You can make the new row selectable, editable, updatable, and deletable, assuming that the current conditions permit such actions. You can also clear the GB automatically after insert so that you do not need to make repeated calls to **Clear Grid Buffer**, which is the system function that removes all values from the GB.

Insert Grid Buffer Row and **Update Grid Buffer Row** both require a row number that is based on the model grid as input so that runtime knows where to place the new row. One way to determine a grid row number is with the system function, **Get Selected Grid Row Number**.

To activate a specific grid row during an event, use **Get Grid Row**. After identifying a row with **Get Grid Row**, all subsequent ER that normally references GCs will reference the identified row instead. The alternate reference lasts for the duration of the event.

To prevent a row from being written to the grid during the runtime fetch, use **Suppress Grid Line**.

To copy or populate a grid row to the GB, use **Copy Grid Row to Grid Buffer**. To remove a grid row from the grid control, use **Delete Grid Row**.

You can set or clear an error on a cell, row, column, or the entire control. Use **Set Grid Cell Error** to set errors and **Clear Grid Cell Error** to clear them.

Change Row Selection

Description

Use this system function to select or deselect grid rows programmatically.

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid form control (FC) to affect.
<i>Row</i>	Input, required. The row to affect. Set the parameter to <i><All Rows></i> , <i><Currently Selected Row></i> , or an applicable object from the object list.
<i>Select State</i>	Input, required. The state (selected or deselected) in which to put the indicated row or rows. Set the parameter to <i><Selected (1)></i> , <i><Unselected (0)></i> , or an applicable object from the object list.

Clear Grid Buffer

Description

Use this system function to clear the grid buffer (GB) manually.

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid FC to affect.

Clear Grid Cell Error

Description

Use this system function to clear an error on a cell, a row, a column, or the entire control. Also, this system function does not take hidden rows into account; in other words, it affects the display grid instead of the model grid.

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid FC to affect.
<i>Row</i>	Input, required. The row in which to clear the error. Set the parameter to an alphanumeric constant (<i><Literal></i>), <i><All Rows></i> , or <i><Currently Selected Row></i> .
<i>Column</i>	Input, required. The column in which to clear the error. Set the parameter to an applicable object (GC grid column) from the object list or <i><All Columns></i> .

See Also

EnterpriseOne Tools 8.94 PeopleBook: System Administration, “Setting Up EnterpriseOne Printing”

Clear QBE Column

Description

Use this system function to clear text from a single QBE column or from all columns.

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid FC to affect.
<i>Column</i>	Input, required. The QBE column to clear. Set the parameter to an applicable object (GC grid column) from the object list or <i><All Columns></i> .

Additional Notes

You might use this function when the QBE column had a value forced into it, such as the current date on a purchase order find/browse form. Then, after the user performs a Find, you could clear the QBE column in case you want to use another date.

Clear Selection

Description

Use this system function to clear all system-function-defined selection information that was used to build previous SQL statements during the current session.

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid FC to affect.

See Also

Chapter 21, “Understanding Grid Controls,” Set Selection, page 177

Clear Sequencing

Description

Use this system function to clear all system-function-defined sequencing information that was used to build previous SQL statements during the current session.

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid FC to affect.

Copy Grid Row to Grid Buffer

Description

Use this system function to write all of the GC fields to GB. Depending on the conditions when this function is initiated, the GC can be either in memory or in the grid. This system function affects the model version of the grid and is one-based.

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid FC to affect.
<i>Row</i>	Input, required. The row to copy to GC. Set the parameter to an alphanumeric constant (<Literal>), <All Rows>, or <Currently Selected Row>, or applicable object from the object list.

Delete Grid Row

Description

Use this function to delete a grid row from the model grid. That is, you can use this system function to affect hidden rows. This system function affects the model version of the grid and is one-based.

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid FC to affect.
<i>Row</i>	Input, required. The row to delete. Set the parameter to an alphanumeric constant (<Literal>), <All Rows>, or <Currently Selected Row>, or applicable object from the object list.

Disable Grid

Description

Use this system function to protect a cell, a row, a column, or the entire control from being edited.

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid FC to affect.
<i>Row</i>	Input, required. The relative row to disable. The first row is 0. Set the parameter to an alphanumeric constant (<Literal>), <All Rows>, or <Currently Selected Row>.
<i>Column</i>	Input, required. The column to disable. Set the parameter to an applicable object (GC grid column) from the object list or <All Columns>.

Additional Notes

This function is useful when you need to complete more information in the header before you can add individual rows or when you want to use information in the header as default information in the grid. Also, this system function does not take hidden rows into account when deleting a row based on an absolute row number value.

See Also

Chapter 21, “Understanding Grid Controls,” Enable Grid, page 167

Display Customized Grid Option

Description

Use this function to hide or display the option that enables users to customize the grid. This includes the customize grid link on the grid header and the *Grid Formats* menu item under Tools menu.

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid FC to affect.
<i>Enable</i>	Input, required. A flag indicating whether to hide (<i>Enable</i> =<FALSE>) or display (<i>Enable</i> =<TRUE>) the option. Set the parameter to <TRUE> or <FALSE>.

Display Export to Excel Option

Description

Use this function to hide or display the option that enables users to export the contents of the grid control to Excel. This includes the “Export to Excel” icon on the grid header and the “Export To Excel” menu item under Tools menu.

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid FC to affect.
<i>Enable</i>	Input, required. A flag indicating whether to hide (<i>Enable</i> =<FALSE>) or display (<i>Enable</i> =<TRUE>) the option. Set the parameter to <TRUE> or <FALSE>.

See Also

[Chapter 21, “Understanding Grid Controls,” Display Export to Word Option, page 166](#)

[Chapter 21, “Understanding Grid Controls,” Display Import from Excel Option, page 167](#)

Display Export to Word Option

Description

Use this function to hide or display the option that enables users to export the contents of the grid control to Word. This includes the *Export to Word* icon on the grid header and the *Export To Word* menu item under Tools menu.

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid FC to affect.
<i>Enable</i>	Input, required. A flag indicating whether to hide (<i>Enable</i> =<FALSE>) or display (<i>Enable</i> =<TRUE>) the option. Set the parameter to <TRUE> or <FALSE>.

See Also

[Chapter 21, “Understanding Grid Controls,” Display Export to Excel Option, page 166](#)

[Chapter 21, “Understanding Grid Controls,” Display Import from Excel Option, page 167](#)

Display Import from Excel Option

Description

Use this function to hide or display the option that enables users to import the contents of an Excel file into a grid. This includes the “Import From Excel” icon on the grid header and the “Import From Excel” menu item under Tools menu.

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid FC to affect.
<i>Enable</i>	Input, required. A flag indicating whether to hide (<i>Enable</i> =<FALSE>) or display (<i>Enable</i> =<TRUE>) the option. Set the parameter to <TRUE> or <FALSE>.

See Also

[Chapter 21, “Understanding Grid Controls,” Display Export to Excel Option, page 166](#)

[Chapter 21, “Understanding Grid Controls,” Display Export to Word Option, page 166](#)

Enable Grid

Description

Use this system function to enable a user to edit a cell, a row, a column, or the entire grid. Also, this system function does not take hidden rows into account when enabling a row based on an absolute row number value.

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid FC to affect.
<i>Row</i>	Input, required. The relative row to enable. The first row is 0. Set the parameter to an alphanumeric constant (<Literal>), <All Rows>, or <Currently Selected Row>.
<i>Column</i>	Input, required. The column to enable. Set the parameter to an applicable object (GC grid column) from the object list or <All Columns>.

See Also

[Chapter 21, “Understanding Grid Controls,” Disable Grid, page 165](#)

Get Grid Row

Description

Use this function to target a grid row for use in an upcoming function in place of the grid row that would be used ordinarily. This system function affects the model version of the grid and is one-based.

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid FC to affect.
<i>Row</i>	Input, required. The relative row to acquire. Set the parameter to an alphanumeric constant (<Literal>), <Blank>, <Zero>, or applicable object from the object list.

Additional Notes

If the row specified is greater than the total number of rows, the last row is used. If the row specified is invalid, the active row becomes zero.

Get Max Grid Rows

Description

Use this system function to count the number of rows in the model grid; that is, it counts both hidden and visible rows.

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid FC to affect.
<i>Row</i>	Input, required. The object to which to assign the return value. Set the parameter to an applicable object from the object list.

Returns

This system function returns the number of rows in the model grid to the object identified by *Row*.

See Also

[Chapter 21, “Understanding Grid Controls,” Disable Grid, page 165](#)

Get Next Selected Row

Description

Use this system function to determine which row in a grid after a given row has been selected.

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid FC to affect.
<i>Start Row</i>	Input, required. The row from which to start searching for a selected row. In other words, runtime will return the first row it finds, after this row, that has been selected. To include the first row of the grid in the search, select the special parameter, <i><Before First Row></i> . Set the parameter to <i><Before First Row></i> or an applicable object from the object list.
<i>Row</i>	Output, required. The object to which to assign the return value. Set the parameter to an applicable object from the object list.

Returns

This system function returns the index of the first selected row the system encounters beneath the indicated row. Runtime writes the index to the object specified by *Row*.

Get Selected Grid Row Count

Description

Use this function to count the number of grid rows in the current selection. The rows to be counted need not be contiguous.

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid FC to affect.
<i>Row</i>	Output, required. The object to which to assign the return value. Set the parameter to an applicable object from the object list.

Returns

This system function returns the number of rows in the current selection to the object identified by *Row*.

See Also

[Chapter 21, “Understanding Grid Controls,” Get Max Grid Rows, page 168](#)

Get Selected Grid Row Number

Description

Use this function to get the row number for a selected row. This system function affects the model version of the grid and is one-based.

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid FC to affect.
<i>Row</i>	Output, required. The object to which to assign the return value. Set the parameter to an applicable object from the object list.

Additional Notes

If multiple rows are selected, the function returns the index value for the first row. Typically, you use this function only when you need to save the row as a variable for future processing.

Returns

This system function returns the index position of the selected row to the object identified by *Row*.

Hide Grid Column

Description

Use this function to prevent an entire column from appearing on the form.

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid FC to affect.
<i>Column</i>	Input, required. The column to hide. Set the parameter to an applicable object (GC grid column) from the object list or <i><All Columns></i> .

See Also

[Chapter 21, “Understanding Grid Controls,” Show Grid Column, page 179](#)

Hide Grid Row

Description

Use this system function to prevent an entire row from appearing on the form. This system function affects the model version of the grid and is one-based.

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid FC to affect.
<i>Row</i>	Input, required. The relative row to hide. Set the parameter to an alphanumeric constant (<i><Literal></i>), <i><All Rows></i> , <i><Currently Selected Row></i> , an applicable object from the object list.

Additional Notes

This is an example of how to use this system function. If you create a form with a grid and you want to summarize rows, add together several rows to determine a total and show the total row. If you use this system function, you might include a check box for the detail information and then display the row instead of repopulating the grid.

If no row is selected when it is invoked, then the system function hides the last row.

See Also

Chapter 21, “Understanding Grid Controls,” Show Grid Row, page 180

Insert Grid Buffer Row

Description

Use this system function to insert a row from the GB into the grid control. After the grid row is inserted, the row will receive focus on update grid. This system function affects the model version of the grid and is one-based.

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid FC to affect.
<i>Row</i>	Input, required. The relative row to insert the row. Set the parameter to an alphanumeric constant (<Literal>), <After Current Row>, or <After Last Row>.
<i>Selectable?</i>	Input, required. An indicator of whether the Select button processes for the inserted row. Set the parameter to <Yes> or <No>.
<i>Protected?</i>	Input, required. An indicator of whether the user can edit the inserted row. Set the parameter to <Yes> or <No>.
<i>Updateable?</i>	Input, required. An indicator of whether runtime will attempt to update or insert the underlying table if the user edits the data and clicks the OK button. Set the parameter to <Yes> or <No>.
<i>Deleteable?</i>	Input, required. An indicator of whether the user can delete the inserted row. Set the parameter to <Yes> or <No>.
<i>Clear After?</i>	Input, required. An indicator of whether runtime should clear the GB automatically immediately after the insert. Set the parameter to <Yes> or <No>.

See Also

Chapter 21, “Understanding Grid Controls,” Update Grid Buffer Row, page 180

Insert Grid Row

Description

This system function is deprecated. Use **Insert Grid Buffer Row** instead.

See Chapter 21, “Understanding Grid Controls,” Insert Grid Buffer Row, page 171.

Set Data Dictionary Item

Description

Use this system function to override form controls and grid columns that are DD items. This system function is affects the model version of the grid, and it is one-based.

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid FC to affect.
<i>Column</i>	Input, required. The column to change. Set the parameter to an applicable object (GC grid column) from the object list or <i><All Columns></i> .
<i>DD Alias</i>	Input, required. The alias of the DD item that you want to use. Set the parameter to a specific DD alias (<i><Pick DD Item></i>) or an applicable object from the object list.
<i>System Code</i>	This parameter is reserved for future functionality.

Additional Notes

This system function makes a complete change; you substitute a different DD item for the existing one. You can also create a new data item that is not the same type as the old item. The new DD item must be the same data type as the previous DD item or the system function call does not make any changes.

Set Data Dictionary Item Overrides

Description

Use this system function to change a specific DD item property. This system function affects the model version of the grid, and it is one-based.

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid FC to affect.
<i>Column</i>	Input, required. The column to override. Set the parameter to an applicable object (GC grid column) from the object list or <i><All Columns></i> .
<i>Overrides</i>	Input, required. The override to apply. Set the parameter to the specific type of override to apply.

Set Grid Cell Error

Description

Use this system function to set an error on a cell, a row, a column, or the entire control. Also, this system function does not take hidden rows into account; in other words, it affects the display grid instead of the model grid.

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid FC to affect.
<i>Row</i>	Input, required. The relative row on which to set the error. Set the parameter to an alphanumeric constant (<Literal>), <All Rows>, <Currently Selected Row>, or applicable object from the object list.
<i>Column</i>	Input, required. The column on which to set the error. Set the parameter to an applicable object (GC grid column) from the object list or <All Columns>.
<i>Error Code</i>	Input, required. The alias of the error to be set on the row and column. Set the parameter to an alphanumeric constant (<Literal>), <Blank>, <Zero>, or applicable object from the object list.

See Also

Chapter 21, “Understanding Grid Controls,” Clear Grid Cell Error, page 163

Set Grid Color

Description

Use this system function to set a color on a cell, a row, a column, or the entire control. You can choose a specific color to set, or you can reset the color which returns the color of the object to its default value. The system function affects the model version of the grid and it is one-based.

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid FC to affect.
<i>Row</i>	Input, required. The relative row on which to set the color. Set the parameter to an alphanumeric constant (<Literal>), <All Rows>, <Currently Selected Row>, or applicable object from the object list.
<i>Column</i>	Input, required. The column on which to set the color. Set the parameter to an applicable object (GC grid column) from the object list or <All Columns>.
<i>Color</i>	Input, required. The color to be set on the row and column. A color from the color palette (<Pick Color>) or the default color value (<Reset Color>).

Set Grid Column Heading

Description

Use this system function to change the text in a grid column heading. This system function affects the model version of the grid, and it is one-based.

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid FC to affect.
<i>Column</i>	Input, required. The column for which to change the header. Set the parameter to an applicable object (GC grid column) from the object list or <i><All Columns></i> .
<i>Text</i>	Input, required. The text to use for the column header. Set the parameter to an alphanumeric constant (<i><Literal></i>), <i><Blank></i> , <i><Zero></i> , or applicable object from the object list

See Also

Chapter 21, “Understanding Grid Controls,” Set Data Dictionary Item Overrides, page 172

Set Grid Font

Description

Use this system function to set a font for the text in a cell, a row, a column, or the entire control. Font in this context includes font type (such as Arial or Times New Roman), style (such as italic or bold), size, effects (strikeout or underline), and color. You can choose a specific font to set, or you can reset the font which returns the font of the object to its default value. This system function affects the model version of the grid, and it is one-based.

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid FC to affect.
<i>Row</i>	Input, required. The relative row for which to set the font. Set the parameter to an alphanumeric constant (<i><Literal></i>), <i><All Rows></i> , <i><Currently Selected Row></i> , or applicable object from the object list.
<i>Column</i>	Input, required. The column on which to set the font. Set the parameter to an applicable object (GC grid column) from the object list or <i><All Columns></i> .
<i>Font</i>	Input, required. The font to be set on the row and column. Set the parameter to a font and related settings from the Font dialog (<i><Pick Font></i>) or the default font settings (<i><Reset Font></i>).

Set Grid Row Bitmap

Description

Use this system function to manipulate the bitmap icon that appears next to the row. This function sets a specific system bitmap, such as a check mark or a trash can, to use as an icon on a specified row header. You cannot modify the icon list. This system function affects the model version of the grid, and it is one-based.

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid FC to affect.
<i>Row</i>	Input, required. The relative row for which to set the bitmap. Set the parameter to an alphanumeric constant (<Literal>), <All Rows>, <Currently Selected Row>, or applicable object from the object list.
<i>Bitmap</i>	Input, required. The bitmap to be set on the row. Set the parameter to the specific bitmap that you want to apply (<Checkbox>, <X Mark>, and so forth).

Set Grid Row Format

Description

This system function applies a system-defined HTML row format or returns to the default value.

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid FC to affect.
<i>Format</i>	Input, required. The format to apply. You can apply the default format, or you can use the alternate HTML formatting string you entered into the Alternate Grid Row Format String parameter for the grid at design-time. Set the parameter to <Default> or <Alternate>.

Set Lower Limit

Description

Use this system function to create the WHERE clause of a SQL search statement for use in a succeeding fetch when you are searching against a table that employs a ragged hierarchy.

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid FC to affect.
<i>Table</i>	Input, required. The source from which to acquire comparison data. Set the parameter to an alphanumeric constant (<Literal>), <Null>, or an applicable object from the object list.
<i>Alias</i>	Input, required. The column of the table from which to acquire comparison data. Set the parameter to an alphanumeric constant (<Literal>), <Null>, or an applicable object from the object list.
<i>Comparison Type</i>	Input, required. The comparison operator to use. Set the parameter to the comparison operator to use (<Equal To>, <Not Equal To>, and so forth).
<i>Comparison Value</i>	Input, required. The comparison value to use. Set the parameter to an alphanumeric constant (<Literal>), <Blank>, <Zero>, or applicable object from the object list.
<i>And/Or</i>	Input, required. Indicates whether to append another WHERE clause to the SQL statement and how the new statement relates to the one before it (AND or OR). Set the parameter to <And>, <Or> or <None>.

Additional Notes

In some cases, referencing an item just by table and column (as you do with **Set Selection**) is insufficient. For example, in EnterpriseOne, some tables, especially in Financials, are structured using ragged hierarchies. A ragged hierarchy is one in which the parent attribute of one or more child attributes is not at the level immediately above the child. In short, some attributes in the hierarchy effectively have an empty parent-level attribute and descend from a grandparent attribute instead.

This table illustrates how information might appear in a ragged hierarchy:

Segment	Family	Class	Commodity
Mineral & Textile			
Mineral & Textile	Mineral, Ores, & Metals		
Mineral & Textile	Mineral, Ores, & Metals	Base Metals	
Mineral & Textile	Mineral, Ores, & Metals	Base Metals	Titanium
Mineral & Textile	Mineral, Ores, & Metals	Precious Metals	
Mineral & Textile	Mineral, Ores, & Metals	Precious Metals	Gold
Time, Jewelry, & Gem			
Time, Jewelry, & Gem	Gemstone		
Time, Jewelry, & Gem	Gemstone	Pearls	

Segment	Family	Class	Commodity
Time, Jewelry, & Gem	Timepiece		
Time, Jewelry, & Gem	Timepiece	Watches	

If you searched at the commodity level, the system would return only titanium and gold because at that level, watches and pearls are technically null values. To compensate, use the system function, **Set Lower Limit**, instead of **Set Selection** to build the WHERE clause of the SQL statement in this situation. With **Set Lower Limit**, you can have the grid display all records starting from the Titanium line down. With conventional filter fields it is impossible to display all the subsequent records because none of the other "class" fields are greater than or equal to Titanium. **Set Lower Limit** not only includes the most restrictive field (in this case Class - Titanium), but also includes each step in the hierarchy to determine if a record is actually "greater than" the lower limit record.

In general, this system function should not be used in cases where the user can affect the select. For example, QBE and filter fields are ways the user can affect the select. If the user can affect the select, then the user's select information should always be taken into consideration. This system function can be used in conjunction with **Set Selection Append Flag** to either append or replace the QBE or filter selection with the system function selection.

See Also

Chapter 21, "Understanding Grid Controls," Set Data Dictionary Item Overrides, page 172

Chapter 21, "Understanding Grid Controls," Set Selection, page 177

Set QBE Column Compare Style

Description

When the application performs a QBE search, this system function enables the application to set which comparison operator to use. For example, if you want to set the QBE for a date column to be > January 1, 2005, you would use QC WorkDate = "010105" and Set QBE Column Compare Style(FC Grid, GC WorkDate, <Greater Than>).

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid FC to affect.
<i>Column</i>	Input, required. The column on which to set the QBE comparison operator. Set the parameter to an applicable object (GC grid column) from the object list or <All Columns>.
<i>Compare Style</i>	Input, required. The comparison operator to use. Set the parameter to <Equal To>, <Not Equal To>, <Greater Than>, <Less Than>, <Greater Than or Equal To>, or <Less Than or Equal To>.

Set Selection

Description

Use this system function to create the WHERE clause of a SQL search statement for use in a succeeding fetch.

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid FC to affect.
<i>Table</i>	Input, required. The source from which to acquire comparison data. Set the parameter to an alphanumeric constant (<Literal>), <Null>, or an applicable object from the object list.
<i>Alias</i>	Input, required. The column of the table from which to acquire comparison data. Set the parameter to an alphanumeric constant (<Literal>), <Null>, or an applicable object from the object list.
<i>Comparison Type</i>	Input, required. The comparison operator to use. Set the parameter to the comparison operator to use (<Equal To>, <Not Equal To>, and so forth).
<i>Comparison Value</i>	Input, required. The comparison value to use. Set the parameter to an alphanumeric constant (<Literal>), <Blank>, <Zero>, or applicable object from the object list.
<i>And/Or</i>	Input, required. Indicates whether to append another WHERE clause to the SQL statement and how the new statement relates to the one before it (AND or OR). Set the parameter to <And>, <Or> or <None>.

Additional Notes

In general, this system function should not be used in cases where the user can affect the select. For example, QBE and filter fields are ways the user can affect the select. If the user can affect the select, then the user's select information should always be taken into consideration. This system function can be used in conjunction with **Set Selection Append Flag** to either append or replace the QBE or filter selection with the system function selection.

See Also

[Chapter 21, "Understanding Grid Controls," Clear Selection, page 164](#)

[Chapter 21, "Understanding Grid Controls," Set Lower Limit, page 175](#)

[Chapter 21, "Understanding Grid Controls," Set Selection Append Flag, page 178](#)

[Chapter 21, "Understanding Grid Controls," Set Sequencing, page 179](#)

Set Selection Append Flag

Description

When building a SQL statement programmatically, use this system function to indicate whether the returned value should replace the existing data or be appended to it.

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid FC to affect.
<i>Append?</i>	Input, required. Indicates whether to append or replace. Set the parameter to append (<Yes>) or replace (<No>).

See Also

[Chapter 21, "Understanding Grid Controls," Set Selection, page 177](#)

Set Sequencing

Description

Use this system function to create the ORDER BY clause of a SQL search statement for use in a succeeding fetch.

Note. Use of this system function implies that grid formats that include sequencing will *not* be used in the SQL.

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid FC to affect.
<i>Table</i>	Input, required. The source from which to acquire comparison data. Set the parameter to an alphanumeric constant (<Literal>), <Null>, or an applicable object from the object list.
<i>Alias</i>	Input, required. The column of the table from which to acquire comparison data. Set the parameter to an alphanumeric constant (<Literal>), <Null>, or an applicable object from the object list.
<i>Sort Order</i>	Input, required. Whether to sort in ascending or descending order. Set the parameter to <Ascending> or <Descending>.

Additional Notes

In general, this system function should not be used in cases where the user can affect the select. For example, QBE and filter fields are ways the user can affect the select. If the user can affect the select, then the user's select information should always be taken into consideration.

See Also

[Chapter 21, "Understanding Grid Controls," Clear Sequencing, page 164](#)

[Chapter 21, "Understanding Grid Controls," Clear Sequencing, page 164](#)

[Chapter 21, "Understanding Grid Controls," Set Selection, page 177](#)

[Chapter 21, "Understanding Grid Controls," Set Selection Append Flag, page 178](#)

Show Grid Column

Description

Use this function to make a hidden column visible. This system function affects the model version of the grid, and it is one-based.

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid FC to affect.
<i>Column</i>	Input, required. The column to show. Set the parameter to an applicable object (GC grid column) from the object list or <All Columns>.

See Also

[Chapter 21, “Understanding Grid Controls,” Hide Grid Column, page 170](#)

Show Grid Row

Description

Use this function to reveal and display a hidden row. This system function affects the model version of the grid, and it is one-based.

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid FC to affect.
<i>Row</i>	Input, required. The relative row to show. Set the parameter to an alphanumeric constant (<Literal>), <All Rows>, or applicable object from the object list.

See Also

[Chapter 21, “Understanding Grid Controls,” Hide Grid Row, page 170](#)

Suppress Grid Line

Description

Use this function to prevent a row from becoming part of the grid. This system function has an effect only when called from the **Grid Record is Fetched** event.

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid FC to affect.

Update Grid Buffer Row

Description

Use this system function to update a row from the grid buffer (GB) into the grid control. This system function affects the model version of the grid and is one-based.

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid FC to affect.
<i>Row</i>	Input, required. The relative row to affect. Set the parameter to an alphanumeric constant (<Literal>), <Currently Selected Row>, or applicable object from the object list.
<i>Selectable?</i>	Input, required. An indicator of whether the Select button processes for the inserted row. Set the parameter to <Yes> or <No>.
<i>Protected?</i>	Input, required. An indicator of whether the user can edit the inserted row. Set the parameter to <Yes> or <No>.
<i>Updateable?</i>	Input, required. An indicator of whether runtime will attempt to update or insert the underlying table if the user edits the data and clicks the OK button. Set the parameter to <Yes> or <No>.
<i>Deleteable?</i>	Input, required. An indicator of whether the user can delete the inserted row. Set the parameter to <Yes> or <No>.
<i>Clear After?</i>	Input, required. An indicator of whether runtime should clear the GB automatically immediately after the insert. Set the parameter to <Yes> or <No>.

Was Grid Cell Value Entered

Description

This system function returns a nonzero value if a specific grid cell has been changed since last time this system function is called. Row is taken from the current context.

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid FC to affect.
<i>Column</i>	Input, required. The column containing the grid cell to check. Set the parameter to an applicable object (GC grid column) from the object list or <All Columns>.
<i>Return To</i>	Input, required. The object to which to assign the return value that designates whether the cell or node has changed. Set the parameter to an applicable object from the object list.

Returns

This system function returns one of these values to the object identified by *Row*:

Value	Description
0	The cell has not changed since the last check.
1	The cell has changed since the last check.

CHAPTER 22

Understanding Image Controls

This chapter discusses image controls.

Image Controls

You can use a bitmap control to create a control that looks like a picture or other artwork. 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 automatically links to a different form. **Button Clicked** is the only event that can fire on an image control.

You also can use the bitmap control for an animated gif instead of a bitmap. An animated gif is particularly useful for Java and HTML applications. The animated gif is animated in Java and HTML applications; in Windows applications, however, it does not appear animated and only the first image of the animated gif file appears.

Image Control Design-Time Considerations

This list describes how to use the design-time properties that are particularly useful for image controls:

- Clickable

If you want to make the image control act as a button (that is, fire the **Button Clicked** event when the user clicks it), enable this option. The option does not automate the image, however. Hence, the user receives no feedback when he or she clicks the control unless you add logic on the **Button Clicked** event to that end.

- Maintain Aspect Ratio

Select this option to maintain the height-to-width ratio of the image when you resize it.

- Prevent Resizing

Select this option to prevent your accidentally resizing the image during design-time.

- Tool Tip

To display text when the user hovers over the control, enter the text you want to show in this property field.

CHAPTER 23

Understanding Media Object Controls

This chapter discusses media object controls.

Media Object Controls

The *media object control* is a specialized control. Use this control to enable the user to enter text or attach objects. You can place this control on any form type except the message box.

A single media object control can contain multiple items. Furthermore, depending on how it is configured, a media object can also contain a variety of different types of items. The user can only work with one at a time, however.

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 also can add generic files or URLs. If you want to display a file that is available on the internet, you can attach the media object control to the form and create a link to the internet.

After you define the media object queue for the internet and include a valid HTTP address, you can use the **Start Web Browser** system function to open the control and display the internet file. For example, you might use this control when you need to verify the Web page for a shipping vendor so that you can track the status of shipments. You can look up the shipment directly within the media object control.

You also can use the media object control to display the employee queues to which messages are sent. Place a media object control next to the tree control on a parent/child form. Next, use event rules (ER) 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 appears in the control.

Media Object Control Design-Time Considerations

You can control the type of media objects users can attach by selecting one or more of these options:

- Allow Image Items
- Allow OLE Items
- Allow RTF Text
- Allow Text Items

Additionally, you can opt to cause a new text item to be created every time the control is opened by selecting the New Text Item on Open option.

Media Object System Functions

This section discusses the system functions unique to media objects.

Access Media Object

Description

Use this system function access a media object. Most of the media object system functions require you to use this system function to access a media object before you can manipulate it in another way.

Parameters

Parameter	Description
<i>Media Object Control</i>	Input, required. The media object FC to affect.
<i>GTName</i>	Input, required. The media object data structure to use. Set the parameter to an alphanumeric constant (<Literal>), a user prompt (<Choose GTName>), or an applicable object from the object list.
<i>Key String</i>	Input, required. The key string of the media object to access. Set the parameter to an alphanumeric constant (<Literal>), <Blank>, <Zero>, or an applicable object from the object list.
<i>Action</i>	Input, required. The display mode for the media object. You can show the media object as a read-only object, or you can enable users to edit the media object. Set the parameter to <Edit> or <Display>.
<i>Status</i>	Output, required. This parameter is reserved for future use.
<i>Active Item</i>	Input, optional. The item to activate (that is, bring to the forefront of the control). A media object can have more than one item associated with it. If you want an item of a certain type to be displayed initially, specify it with this parameter. You can also choose to enable the user to select the initial display item type as well. Set the parameter to <First Image Item>, <First OLE Item>, <First Text Item>, or <Specify Item#>.

Activate Item

Description

Use this system function to activate (that is, bring forward) a specific item in a given media object. You must first access the media object itself before activating its items.

Parameters

Parameter	Description
<i>Media Object Control</i>	Input, required. The media object FC to affect.
<i>Item</i>	Input, required. The ID of the item to activate. Set the parameter to <First Image Item>, <First OLE Item>, <First Text Item>, or <Specify Item#>.

See Also

[Chapter 23, “Understanding Media Object Controls,” Access Media Object, page 186](#)

Clear Characterization Cache

Description

Use this system function to clear all values in the characterization cache that were set previously by the **Set Characterization Cache** system function. After this system function is called, adding media objects will not automatically generate a record in the F00166 table, which is where metadata about a media object is saved.

Parameters

Parameter	Description
<i>Media Object Control</i>	Input, required. The media object FC to affect.
<i>GTName</i>	Input, required. The media object data structure to clear. Set the parameter to an alphanumeric constant (< <i>Literal</i> >), a user prompt (< <i>Choose GTName</i> >), or an applicable object from the object list.

See Also

[Chapter 23, “Understanding Media Object Controls,” Disable Characterization Cache, page 187](#)

[Chapter 23, “Understanding Media Object Controls,” Set Characterization Cache, page 191](#)

Delete Item

Description

Use this system function to delete a specified item from a given media object. You must first access the media object itself before deleting its items.

Parameters

Parameter	Description
<i>Media Object Control</i>	Input, required. The media object FC to affect.
<i>Item</i>	Input, required. The ID of the item to delete. Set the parameter to < <i>First Image Item</i> >, < <i>First OLE Item</i> >, < <i>First Text Item</i> >, or < <i>Specify Item#</i> >.

See Also

[Chapter 23, “Understanding Media Object Controls,” Access Media Object, page 186](#)

[Chapter 23, “Understanding Media Object Controls,” Activate Item, page 186](#)

Disable Characterization Cache

Description

Use this system function to prevent runtime from changing the media object icon to represent the characterization of the media object.

Parameters

Parameter	Description
<i>Media Object Control</i>	Input, required. The media object FC to affect.

Get OLE Item

Description

This system function accepts a media object item sequence number and an OLE automation server interface as parameters and returns a handle to the OLE object with that sequence number in the media object record.

Parameters

Parameter	Description
<i>Media Object Control</i>	Input, required. The media object FC to affect.
<i>Item</i>	Input, required. The ID of the item containing the OLE object to be fetched. Set the parameter to <First Image Item>, <First OLE Item>, <First Text Item>, or <Specify Item#>.
<i>ObjectRef</i>	Output, required. The object to which to return the ActiveX/COM interface variable. Set the parameter to an applicable object from the object list.

Returns

This system function returns the handle of the referenced OLE item to the object specified by *ObjectRef*. It also returns one of these two values:

Value	Description
NOERROR	Indicates that the system function succeeded.
E_FAIL	Indicates that the system function failed.

See Also

[Chapter 23, “Understanding Media Object Controls,” Insert OLE Object, page 188](#)

[Chapter 23, “Understanding Media Object Controls,” Insert URL, page 190](#)

Insert OLE Object

Description

This system function adds an OLE object to a given media object.

Parameters

Parameter	Description
<i>Media Object Control</i>	Input, required. The media object FC to affect.
<i>Item Name</i>	Output, required. The object to which to return the name of the item. Set the parameter to an alphanumeric constant (<Literal>), <Blank>, <Zero>, or an applicable object from the object list.
<i>Item</i>	Input, required. The object to which to return the next available ID. Set the parameter to an alphanumeric constant (<Literal>) or <Null>.
<i>ObjectRef</i>	Output, optional. Reserved for future functionality.

Returns

This system function returns the name of the inserted object, as well as the next available ID value to the objects indicated by *Item Name* and *Item*, respectively. It also returns one of these two values:

Value	Description
NOERROR	Indicates that the system function succeeded.
E_FAIL	Indicates that the system function failed.

See Also

Chapter 23, “Understanding Media Object Controls,” Get OLE Item, page 188

Insert Text

Description

Use this system function to add text object to a given media object.

Parameters

Parameter	Description
<i>Media Object Control</i>	Input, required. The media object FC to affect.
<i>Text ID</i>	Input, required. The ID to assign to the item being inserted. Set the parameter to <Default Text Object> or an alphanumeric constant (<Literal>).
<i>Text</i>	Input (EVDT_STRING), required. The actual text to be inserted. Set the parameter to an alphanumeric constant (<Literal>), <Null>, or an applicable object from the object list.

Insert URL

Description

Use this system function to place an HTML page in a media object. You must pass in the media object queue name where the HTML file resides and the name of the file.

Parameters

Parameter	Description
<i>Media Object Control</i>	Input, required. The media object FC to affect.
<i>MO Queue Name</i>	Input, required. The media object queue in which HTML files reside. Set the parameter to an alphanumeric constant (<Literal>), <Blank>, <Zero>, or an applicable object from the object list.
<i>HTML File Name</i>	Input, required. The name of the HTML file to be attached. Set the parameter to an alphanumeric constant (<Literal>), <Blank>, <Zero>, or an applicable object from the object list.

Returns

This system function returns one of these two values:

Value	Description
NOERROR	Indicates that the system function succeeded.
E_FAIL	Indicates that the system function failed.

See Also

[Chapter 23, “Understanding Media Object Controls,” Get OLE Item, page 188](#)

[Chapter 23, “Understanding Media Object Controls,” Insert OLE Object, page 188](#)

Hide the Viewer Icon Panel

Description

This system function hides the viewer icon panel on a given media object control. When the panel is hidden, users cannot choose particular media objects to view, although they can interact with any media object displayed in the main portion of the control. You can use the **Activate Item** system function to display different items in the media object programmatically.

Parameters

Parameter	Description
<i>Media Object Control</i>	Input, required. The media object FC to affect.
<i>Hide Icon Panel?</i>	Input, required. Indicates whether to hide the icon panel. Set the parameter to <Yes> or <No>.

Lock the Viewer Splitter Bar

Description

This system function exists for backwards compatibility. Do not use it.

Set Characterization Cache

Description

Use this system function to set data in the characterization cache. This cache is saved in the F00166 table and is used as metadata.

Parameters

Parameter	Description
<i>Media Object Control</i>	Input, required. The media object FC to affect.
<i>GTName</i>	Input, required. The media object data structure to use. Set the parameter to an alphanumeric constant (<Literal>), a user prompt (<Choose GTName>), or an applicable object from the object list.
Categories	Input, required. The data values to use as metadata. Double-click <Define Characterization> to set the values.

Set Cursor Position

Description

This system function sets the cursor in a text type media object item either at the beginning of the text (Home position).

Note. This system function has been marked for deprecation in a future release; therefore, avoid its use.

Parameters

Parameter	Description
<i>Media Object Control</i>	Input, required. The media object FC to affect.
<i>Text ID</i>	Input, required. The ID of the text item you want to affect. You can select a specific item, or you can choose to affect the default text item if you have configured the control at runtime so that a default text item launches each time the user opens the control. Set the parameter to <i><Default Text Object></i> or an alphanumeric constant (<i><Literal></i>).
<i>Position</i>	Input, required. The position at which to place the cursor. Set the parameter to <i><HOME></i> .

Set Grid Text Indicator

Description

Use this system function to determine whether a media object is associated with a given row record. If so, runtime can display a paperclip icon in its row header as a signal to the user that the media object exists.

Parameters

Parameter	Description
<i>Grid</i>	Input, required. The grid FC to affect.
<i>Row</i>	Input, required. The row to check. Set the parameter to an alphanumeric constant (<i><Literal></i>), <i><All Rows></i> , <i><Currently Selected Row></i> , or an applicable object from the object list.
<i>Set Indicator?</i>	Input, required. Indicates whether runtime should display the paperclip icon for the user if a media object is associated with the row. Set the parameter to <i><Yes></i> or <i><No></i> .

Set Text Color

Description

This system function enables you to set the text color for a text-type item in a given media object control.

Parameters

Parameter	Description
<i>Media Object Control</i>	Input, required. The media object FC to affect.
<i>Text ID</i>	Input, required. The ID of the text item you want to affect. You can select a specific item, or you can choose to affect the default text item if you have configured the control at runtime so that a default text item launches each time the user opens the control. Set the parameter to <i><Default Text Object></i> or an alphanumeric constant (<i><Literal></i>).
<i>Color</i>	Input, required. The color to be set. Set the parameter to a color from the color palette (<i><Pick Color></i>) or the default color value (<i><Reset Color></i>).

CHAPTER 24

Understanding Parent Child Controls

This chapter discusses the parent child control.

Parent Child Controls

The parent child control is a composite control with a tree on the left and a grid on the right. The control is used to provide a hierarchical view of the business data. Users can resize the tree part of the control during runtime. The tree and grid portions use the same business view, if one is attached. If an editable parent child control links to a business view and user has edited the content, the change is saved by runtime when the form is saved.

At most, a form can contain a single parent child control. Additionally, a form cannot contain both a parent child control and a grid control. You cannot place a parent child control on a tab page; although you can nest a reusable subform containing a parent child control on a tab page.

You can add data dictionary (DD) items to the grid. Because of its similarity to a grid control, many of the properties for parent child controls are the same.

After you configure the control, you set up either a parent/child relationship or use event rules (ER) to load child nodes to the tree. The method that you use depends on whether the table has an inherent parent/child relationship.

You can use ER and system functions to customize the way in which the parent child control functions. For example, you can change the top node of a tree or change the node that appears as the first child on a tree.

To increase performance during runtime, parent child controls can make use of page-at-a-time processing. Page-at-a-time processing ensures that each fetch fetches only 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 the number of nodes that can fit in the current view. When the Find process begins, only one page of 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 the expand-all style of parent/child forms is similar to that for standard mode. When the Find process starts, one page of first-level nodes is 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 deep tree might affect performance.

See Also

Chapter 21, “Understanding Grid Controls,” page 141

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. In this case, the runtime engine does not automatically fetch the child database records because it does not know the table from which to retrieve them.

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 table for the uppermost node. You can use a parent filter in the control, and the runtime engine loads the first level nodes to the tree. If you cannot do this, you must insert the first-level nodes yourself. To do this, you typically use table I/O on the **Button is Clicked** event of the Find button. You use the same methods that you use to insert child nodes. Use a **Suppress Find** system function to stop the runtime engine from attempting to load any nodes.

Whenever a node is expanded, the system function, **Suppress Fetch on Node Expand**, is called from the event, **Tree Node Is Expanded**. This function tells the runtime engine not to do any fetches because ER will handle the loading of the child nodes. **Tree Node Is Expanding** is the main event of the application. This event occurs when the tree node is expanding (such as when the plus next to a child node is expanded for the first time). You place ER 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 come from different tables, based on the type of parent node that is expanded. If possible, you should perform a `SELECT` and then use the `FETCH NEXT` command to retrieve records in a `DO WHILE` loop. The grid buffer (GB) runtime data structure is populated with data from the records read in the loop, and then an **Insert Grid Buffer Row** system function is called. This parent/child system function is different from the **Insert Grid Buffer Row** in the normal grid section. At this point, you also can set custom tree bitmaps using the **Set Tree Node Bitmap** system function.

Lean Manufacturing

Lean manufacturing (also known as *Psynch* or *Product synch*) is a special mode of the parent child control that was created to be used in applications designed to support lean manufacturing processes. It displays a parent child grid in a graphical format specific to lean manufacturing. In many respects, including runtime processing, the lean manufacturing control functions in the same way as the standard parent child control.

Parent Child Control Design-Time Considerations

This section discusses parent child control properties and power forms.

Parent Child Control Properties

Parent child controls are a specialized form of the grid control. Therefore, many of the design options that apply to grid controls also apply to parent child controls. This table details some key design-time settings specifically for the parent child control and their impact. All locations are relative to the parent child control Properties form:

Setting	Impact
Multiple Select	Select this option to permit the user to choose multiple lines to affect with a single operation. If cleared (the default), users cannot affect multiple nodes for operations such as cut-and-paste.
Disable Drag & Drop (Move), Disable Copy, Disable Drag & Drop (Cut/Copy/Paste), Move Up and Down, Indent and Outdent	Select these options to enable the user to affect the contents and organization of the tree.
Location Indicator Feature	Select this option to provide a button that permits the user to toggle showing and hiding location indicators in the tree. The location indicator is a numerical representation of the position of a node in the tree.
Expand All/Collapse All	Select this option to provide a button for the user to expand or collapse the entire tree.

Parent Child Control and Power Forms

If you place the parent child control on a power edit form, an embedded subform on a power edit form, or an editable reusable subform, then the parent/child will be editable within these parameters:

- Both tree columns and the grid columns (except for parent child relationship columns) are editable unless they have been rendered read-only by application logic.
- Columns that define the parent/child key relationship are not editable.
- Runtime provides format and validation for grid cells and tree nodes, similar to the one provided for regular grid.
- The column events, **Column Is Exited**, **Column Is Exited and Changed In-Line**, and **Column Is Exited and Changed - Asynchronous**, are supported for the tree column and all grid columns.
- The row events, **Row Is Exited**, **Row Is Exited and Changed In-Line**, **Row Is Exited and Changed - Asynchronous**, are supported for parent child controls.
- The event sequence is the same as the grid event sequence.
- Form default buttons such as OK and Delete work for parent child controls as they do for grid controls.

At most, a power form or a subform can contain a single parent child control directly. However, a power form can contain multiple subforms, and each subform can contain a parent child control. Similarly, while you cannot place a parent child control on a tab page, you can create a subform as a tab page and that subform can contain a parent child control.

Lean Manufacturing Properties

Place the parent child control in PSYNC mode. This enables the Product Sync Mapping property for grid columns in the control. For every item in the property, you must place a corresponding column in the grid and map it to the property.

Grid columns in the parent child control in PSYNC mode have the Product Synch Mapping property. This list summarizes its parameters:

- *Attach Path to Segment*

Parameter 2 no longer permits the *<Currently Selected Row>* special value.

- *Change Row Selection*

Parameter 3 special value is *<Unselected>* (as opposed to *<Deselected>*).

- *Set Action*

Parameter 2 has no special values. Parameter 3 special values are *<Enabled (1)>* and *<Disabled (0)>*.

Parent Child Control Events

These events can fire on the parent child control during runtime:

- **Delete Grid Rec Verify - Before**
- **Delete Grid Rec Verify - After**
- **Delete Grid Rec from DB-Before**
- **Delete Grid Rec from DB-After**
- **Double Click on Row Header**
- **Set Focus on Grid**
- **Kill Focus on Grid**
- **Row is Entered**
- **Get Custom Tree Node**
- **Tree Node Bitmap is Clicked**
- **Node Outdent Verify Before**
- **Tree Node is Outdented**
- **Node Indent Verify Before**
- **Tree Node is Indented**
- **Node Move Down Verify Before**
- **Node is Moved Down**
- **Node Move Up Verify Before**
- **Node is Moved Up**
- **Tree-Node Level Changed**
- **Tree - Begin Drag/Drop/Copy**
- **Tree - Drag Over Node**
- **Tree - Cancel Drag Drop/Paste**
- **Tree - End Drag Drop/Paste**
- **Kill Focus on Control**
- **Set Focus on Control**
- **Tree Node Is Collapsing**

- **Tree Node Is Expanding**
- **Tree Node Selection Change**
- **Tree Node Is Deleted**
- **Row is Exited**

Many of these events are unique to parent child controls and can be categorized for discussion:

- Selecting tree nodes.
- Performing drag-and-drop or copy/cut/paste.
- Expanding and collapsing nodes.
- Clicking bitmaps.

Selecting Tree Nodes

The **Tree Node Selection Changed** event runs every time a user selects a tree node, either by clicking it once with the mouse or by moving an arrow up and down the nodes. You can place ER that need to run when a node is selected on the **Tree Node Selection Changed** event. The Work Center application uses this feature to load the media object that appears next to the tree with the message information for each node. You can also use this event when you need to protect controls or exits, based on the kind of node that the user selects.

Performing Drag-and-Drop or Copy/Cut/Paste

If you enable the functionality, users can affect the tree structure by adding, moving, and deleting nodes. In Windows environments, users accomplish these tasks with drag-and-drop functions. In Web environments, users accomplish these tasks with copy/cut/paste.

When you place the control on a form initially, all three options are enabled:

- Cut (drag-and-drop move).
- Copy.
- Cut-copy-paste (drag-and-drop).

You can disable these options at design time. When an option is enabled and a user attempts the operation, the cursor indicates that the function is not permitted, and none of the associated events execute. You can control operations to the database using these events:

- **Tree - Begin Drag/Cut/Copy**

If you enable any of the three operations, the **Begin Drag** event fires, and the drag mode is set to Move for the first two operations (cut or copy). The mode is set to Copy for the third operation (cut-copy-paste).

- **Tree - Drag Over Node**

You can attach ER to this event to verify that the node on which the dragged record is about to be dropped 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 is not permitted. If the cursor is not the No Drop cursor, when the record is dropped the event, **Tree - End Drag Drop/Paste**, runs. The same is true of cut-and-paste or copy-and-paste operations.

- **Tree - End Drag Drop/Paste**

You can attach ER to this event to update or insert information that has been moved or copied using the drag. Be aware of the effect of using **Insert Grid Buffer Row** in the **Tree -End Drag Drop/Paste** event, as well as deleting the grid row dragged if the mode is a move.

- **Tree - Cancel Drag Drop/Paste**

No matter the operation, if the user cancels, then this event occurs.

In addition to drag-and-drop or copy/cut/paste, users (and applications) can simply move nodes up and down or change their hierarchical level (indent and outdent). These events are provided in connection with these activities:

- **Node Move Up Verify Before**

When an attempt to move a node up occurs, runtime fires this event. If the system function, **Suppress Node Move Up/Down** is called within this event, then move up is canceled. Otherwise, runtime moves the node upwards one position and then fires the event, **Node is Moved Up**.

- **Node Is Moved Up**

When a node has been moved up by one position (if permitted on the event, **Node Move Up Verify Before**), then the event, **Node is Moved Up**, is fired for that node. Its child nodes are still parented to the moved node. You cannot move a node that is the first node on its level up.

- **Node Move Down Verify Before**

When an attempt to move a node down occurs, runtime fires this event. If the system function, **Suppress Node Move Up/Down** is called within this event, then move down is canceled. Otherwise, runtime moves the node downwards one position and then fires the event, **Node is Moved Down**.

- **Node Is Moved Down**

When a node has been moved down by one position (if permitted on the event, **Node Move Down Verify Before**), then the event, **Node is Moved Down**, is fired for that node. Its child nodes are still parented to the moved node. You cannot move a node that is the last node on its level down.

- **Node Indent Verify Before**

When an attempt to indent a node occurs, runtime fires this event. If the system function, **Suppress Node Indent/Outdent**, is called within this event, then indent is canceled. Otherwise, runtime demotes the node to be a child of its previous sibling (the demoted node remains the parent of its children). Then the event, **Tree Node is Indented**, is fired.

- **Tree Node is Indented**

When a node has been indented (if permitted on the event, **Node Indent Verify Before**), then the event, **Tree Node is Indented**, is fired for that node. Its child nodes are still parented to the indented node. The application must update the node's internal data to reflect its new parent. You cannot indent a node that is the first node on its level.

- **Node Outdent Verify Before**

When an attempt to outdent a node occurs, runtime fires this event. If the system function, **Suppress Node Indent/Outdent**, is called within this event, then outdent is canceled. Otherwise, runtime promotes the node to be a parent of the sibling beneath it. The original parent becomes the sibling of the outdented node. If the outdented node is the first child, then the original parent loses all of its children to the outdented node and becomes a leaf node. The child nodes of the outdented nodes are otherwise unaffected. Then the event, **Tree Node is Outdented**, is fired.

- **Tree Node is Outdented**

When a node has been outdented (if permitted on the event, **Node Outdent Verify Before**), then the event, **Tree Node is Outdented**, is fired for that node. The application must update the node's internal data to reflect its new parent.

- **Tree Node Bitmap is Clicked**

Under most conditions, each node in the tree has a bitmap next to it for display purposes only. You can use **Set Tree Node Clickable Bitmap** to provide an additional bitmap which is clickable; that is, clicking the bitmap causes the event, **Tree Node Bitmap is Clicked**, to fire.

Expanding and Collapsing Nodes

ER exists to signal when a node has been expanded or collapsed. These events occur whether the change in the node's status is due to the user or the application:

- **Tree Node Is Expanding**

This event occurs when a node is expanded for the first time after control initialization.

- **Tree Node Is Collapsing**

This event occurs when a node is collapsed after control initialization.

Example: Using the Tree Node is Expanded Event

In this example, ER is 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 the business units 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
//
// While Loop to pick up the accounts till the Fetch Fails.
While VA frm_ExistAcctLoop is equal to <Zero>
F0901.FetchNext
GB Sub = VA frm_DBSUB
GB ObjAcct = VA frm_DBOBJ
GB Name = VA frm_AcctDesc
GB BusinessUnit = VA frm_CURBU
VA frm_AcctDesc = concat([VA frm_BLANKS],[VA frm_AcctDesc])
GB AccountID = VA frm_AIDF0901
GC AccountID = VA frm_AIDF0901
If SV File_IO_Status          is equal to CO ERROR
VA frm_ExistAcctLoop = "1"
Else
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 = " "

```

Clicking Bitmaps

With the system function, **Set Tree Node Clickable Bitmap**, you can display a bitmap next to a node which fires the event, **Tree Node Bitmap is Clicked**, when the user clicks the bitmap.

Parent Child Control System Functions

This section discusses the system functions unique to the parent child control.

In general, you can use system functions to perform these types of tasks in the parent child control:

- Affect user interactions with the control.
- Acquire information about a node.
- Load data into the control.
- Format data in the control.

Users have the ability to change the tree's hierarchical structure by moving nodes up and down. Users can also increase the indent or outdent of a node as well. To prevent vertical changes to the tree structure, use **Suppress Node Move Up/Down**. To prevent horizontal changes, use **Suppress Node Indent/Outdent**. Typically, you will call these system functions inside the events, **Node Indent Verify Before**, **Node Outdent Verify Before**, **Node Move Up Verify Before**, or **Node Move Down Verify Before**, to prevent the user from performing the action (move up/down or indent/outdent).

Under most conditions, each node in the tree has a bitmap next to it for display purposes only. You can use **Set Tree Node Clickable Bitmap** to provide an additional bitmap which is clickable; that is, clicking the bitmap causes the event, **Tree Node Bitmap is Clicked**, to fire.

You can use system functions to acquire this information about any node in the tree:

- Given its location, the node ID (**Get Node ID**).
- Given its ID, the node row number (**Get Row Number**).
- Given its ID, the node parent, siblings, and child IDs (**Get Related Node ID**).

When application logic is used to load the tree instead of the runtime logic, the application must often insert a tree node at a specific location based on data column values. To achieve this, at design time, select a column to be the node ID column. An application can specify a column as the node ID column. The value in this column will be used as a unique identifier for that row. The node ID column can be any data type, but must be unique. Use the system functions, **Insert Grid Buffer Row By Node ID** and **Get Related Node ID**, to specify a node ID column. Node ID column and these system functions work together: Node ID column needs to exist for these system functions to work. If none of these system functions is called, you do not need to specify a node ID column.

After you specify a node ID column, use the **Insert Grid Buffer Row By Node ID** system function to insert a row at any location, based on its node ID value and Insert As parameter. Use this method to permit an application to insert tree nodes at any position of the tree. Calling the **Insert Grid Buffer Row By Node ID** system function is the only way to add an entry row in the parent child control.

The parent child control provides some default bitmaps for tree nodes that are displayed to indicate the current status of that node (expanded, collapsed, and so forth). You can substitute your own bitmap to display for a node, however, with **Set Tree Node Bitmap**.

For each node, the parent child control can automatically generate and update hierarchical numbers called location indicators. If you enable the feature at design time, then users will see the location indicators button at runtime. By default, the numbering schema is 1, 1.1, 1.2, 1.3, 2, 2.1, and so on. You cannot modify the location indicator in any other way.

To insert an entry row in an empty parent child control, you must first call the system function, **Set Root Node ID**, to assign an ID to the hidden root node. By default, the root node in an empty tree is hidden, and therefore has no ID. After assigning an ID to the root node, then call the system function, **Insert Grid Buffer Row By Node ID**, and pass in the root node ID. If the control does not have a node ID column, runtime produces an error.

Add Action

Description

Use this system function to add a context-based action to the parent child control in Product Sync mode.

Parameters

Parameter	Description
<i>Parent/Child</i>	Input, required. The parent child FC to affect.
<i>Action Name</i>	Input, required. The name of the action to add. Set the parameter to <i><Separator></i> or an applicable object from the object list.
<i>Action ID</i>	Input, required. The ID of the action to add. Set the parameter to <i><Separator></i> or an applicable object from the object list.
<i>Parent Action ID</i>	Input, required. Reserved for future functionality. Set to <i><Root Action></i> .
<i>Icon</i>	Input, required. The icon to display. Set the parameter to <i><Choose Action Bitmap></i> , <i><Default></i> , or <i><Separator></i> .

Additional Notes

If *Action Name*, *Action ID*, or *Icon* are specified as *<Separator>* then the entire action is a separator, and the values of the other parameters are to be ignored.

Attach Path To Segment

Description

Use this system function to add a PSYNC path to a specific PSYNC segment or to all the segments in the PSYNC graph. It recursively adds all of the segments of a path to a given segment, facilitating reuse.

Parameters

Parameter	Description
<i>Parent/Child</i>	Input, required. The parent child FC to affect.
<i>Product Synch ID</i>	Input, required. The PSYNCH node to add the path. Set the parameter to < <i>Currently Selected Row</i> > or an applicable object from the object list.
<i>Path ID</i>	Input, required. The path to add to the segment. Set the parameter to an applicable object from the object list.
<i>Error Code</i>	Input, required. The object to which to assign the return value that indicates whether the function executed without error. Set the parameter to an applicable object from the object list.

Returns

This system function returns one of these values to the object identified by *Error Code*:

Value	Description
0	No error
1	Recursive
2	Other error

Change Row Selection

Description

Use this system function to select or deselect a row in the grid or parent child control in Product Synch mode.

Parameters

Parameter	Description
<i>Parent/Child</i>	Input, required. The parent child FC to affect.
<i>Row</i>	Input, required. The row to affect. Set the parameter to <i><All Rows></i> , <i><Currently Selected Row></i> , or applicable object from the object list.
<i>Select State</i>	Input (EVDT_INT), required. The mode (such as Selected or Deselected) to which to set the row. Set the parameter to <i><Selected></i> (1), <i><Deselected></i> (0), or an applicable object from the object list.

Clear Grid Buffer

Description

Use this system function to clear the GB manually.

Parameters

Parameter	Description
<i>Parent/Child - Grid</i>	Input, required. The parent child FC to affect.

See Also

[Chapter 24, “Understanding Parent Child Controls,” Copy Grid Row To Grid Buffer, page 205](#)

[Chapter 24, “Understanding Parent Child Controls,” Update Grid Buffer Row, page 225](#)

Clear Grid Cell Error

Description

Use this system function to clear an error on a cell, a tree node, a row, a column or the entire parent child control. Also, this system function does not take hidden rows into account; in other words, it affects the user grid instead of the model grid.

Parameters

Parameter	Description
<i>Parent Child</i>	Input, required. The parent child FC to affect.
<i>Row</i>	Input, required. The row in which to clear the error. Set the parameter to an alphanumeric constant (<i><Literal></i>), <i><All Rows></i> , or <i><Currently Selected Row></i> .
<i>Column</i>	Input, required. The column in which to clear the error. Set to <i><All Columns></i> .

Clear QBE Column

Description

Use this system function to clear text from the QBE columns in the control.

Parameters

Parameter	Description
<i>Parent Child</i>	Input, required. The parent child FC to affect.
<i>Column</i>	Input, required. The QBE column to clear. Set to <i><All Columns></i> .

Contact Tree Node

Description

Use this system function to collapse a tree node (for example, so that none of its children are displayed).

Parameters

Parameter	Description
<i>Parent/Child</i>	Input, required. The parent child FC to affect.
<i>Node</i>	Input, required. The target node to affect. Set the parameter to <i><Currently Selected Node></i> , <i><Last Inserted Node></i> , or <i><Currently Expanding/Collapsing Node></i> .

Copy Grid Row To Grid Buffer

Description

Use this system function to write all of the GC fields to GB. Depending on the conditions when this function is initiated, the GC can be either in memory or in the grid.

Parameters

Parameter	Description
<i>Parent/Child - Grid</i>	Input, required. The parent child FC to affect.
<i>Row</i>	Input, required. The row to copy. Set the parameter to an alphanumeric constant (<i><Literal></i>), <i><Currently Selected Row></i> , <i><Currently Expanding/Collapsing Node></i> , or an applicable object from the object list.

See Also

Chapter 24, “Understanding Parent Child Controls,” Clear Grid Buffer, page 204

Chapter 24, “Understanding Parent Child Controls,” Update Grid Buffer Row, page 225

Delete All Actions

Description

Use this system function to remove all context-based actions from the specified parent child control in Product Synch mode.

Parameters

Parameter	Description
<i>Parent/Child</i>	Input, required. The parent child FC to affect.

Delete All Tree Nodes

Description

Use this system function to delete all of the tree nodes (except for the hidden root) from a tree.

Parameters

Parameter	Description
<i>Parent/Child</i>	Input, required. The parent child FC to affect.

Delete Grid Row

Description

Use this function to delete a grid row from the model grid. That is, you can use this system function to affect hidden rows.

Parameters

Parameter	Description
<i>Parent/Child - Grid</i>	Input, required. The parent child FC to affect.
<i>Row</i>	Input, required. The relative location to insert the row. Set the parameter to an alphanumeric constant (<Literal>), <All Rows>, <All Child Nodes Under the Current Node>, <Drag Node Row>, <Currently Selected Row>, or an applicable object from the object list.

Disable Grid

Description

Use this system function to disable a cell, a tree node, a row, the column or the entire parent child control. This system function does not take hidden rows into account when deleting a row based on an absolute row number value.

Parameters

Parameter	Description
<i>Parent Child</i>	Input, required. The parent child FC to affect.
<i>Row</i>	Input, required. The row to be disabled. Set the parameter to an alphanumeric constant (<Literal>), <All Rows>, or <Currently Selected Row>.
<i>Column</i>	Input, required. The column to be disabled. Set to <All Columns>.

See Also

Chapter 24, “Understanding Parent Child Controls,” Update Grid Buffer Row, page 225

Enable Grid

Description

Use this system function to enable a grid cell, a tree node cell, a row, a column or the entire parent child control. This system function does not take hidden rows into account when deleting a row based on an absolute row number value.

Parameters

Parameter	Description
<i>Parent Child</i>	Input, required. The parent child FC to affect.
<i>Row</i>	Input, required. The row to be enabled. Set the parameter to an alphanumeric constant (<Literal>), <All Rows>, or <Currently Selected Row>.
<i>Column</i>	Input, required. The column to be enabled. Set to <All Columns>.

See Also

Chapter 24, “Understanding Parent Child Controls,” Disable Grid, page 206

Expand Tree Node

Description

This system function expands nodes at a given hierarchical level and lower and collapses the rest, all relative to a given node. Expanding a very large tree (for example, one with 1000 nodes or more) can cause a reduction in performance.

Parameters

Parameter	Description
<i>Parent/Child</i>	Input, required. The parent child FC to affect.
<i>Node</i>	Input, required. The target node to expand. Set the parameter to <Currently Selected Node>, <Last Inserted Node>, or <Currently Expanding/Collapsing Node>.

Get Grid Row

Description

Use this function to target a grid row for use in an upcoming function in place of the grid row that would be used ordinarily.

Parameters

Parameter	Description
<i>Parent/Child - Grid</i>	Input, required. The parent child FC to affect.
<i>Row</i>	Input, required. The relative row to target. Set the parameter to an alphanumeric constant (<Literal>), <Blank>, <Zero>, or an applicable object from the object list.

Get Max Grid Rows

Description

Use this system function to count the number of rows in the model grid; that is, it counts both hidden and visible rows.

Parameters

Parameter	Description
<i>Parent/Child - Grid</i>	Input, required. The parent child FC to affect.
<i>Row</i>	Input, required. The relative row to target. Set the parameter to an applicable object from the object list.

Get Next Selected Row

Description

Use this system function to get the next selected row from a starting position in Product Synch mode. This applies to the display grid and not the model grid.

Parameters

Parameter	Description
<i>Parent/Child</i>	Input, required. The parent child FC to affect.
<i>Start Row</i>	Input, required. The row under which to find the next selected row. Typically, you pass <Before First Row> on the first row and then the value of the current row on subsequent calls. Set the parameter to <Before First Row> or an applicable object from the object list.
<i>Selected Row</i>	Output, required. The object to which to assign the value indicating the next selected row. Set the parameter to an applicable object from the object list.

Returns

This system function returns a value to the object identified by *Selected Row*. If no rows are selected after the start row, then the system function returns a value of -1.

Get Node ID

Description

This system function returns the ID of a node at a given location. You can use this system function only if the control has a node ID column.

Parameters

Parameter	Description
<i>Parent Child</i>	Input, required. The parent child FC to affect.
<i>Node Location</i>	Input, required. The location of the node for which you want an ID. Set the parameter to <Root Node>, <Currently Selected Node>, <Last Inserted Node>, <Currently Expanding/Collapsing Node>, <Currently Deleted Node>, <Drag Over Node>, or an applicable object from the object list.
<i>Node ID</i>	Output, required. The object to which you want to assign the return value that designates the node ID. The value must be a variable or control with the same data type as the node ID column; a type mismatch will be checked by runtime engine. Set the parameter to an applicable object from the object list.
<i>Error Code</i>	Output, required. The object to which you want to assign the return value that designates the error code for the transaction.

Returns

This system code returns two values. The node ID itself is returned to the object identified by *Node ID*. A value indicating whether the node was actually found is sent to the object identified by *Error Code* and can have one of two values:

Value	Description
0	The node ID was found successfully.
Nonzero value	The node ID was not found.

See Also

Chapter 24, “Understanding Parent Child Controls,” Get Related Node ID, page 210

Chapter 24, “Understanding Parent Child Controls,” Insert Grid Buffer Row By Node ID, page 214

Get Node Level

Description

This system function returns the hierarchical level of a given node.

Parameters

Parameter	Description
<i>Parent/Child</i>	Input, required. The parent child FC to affect.
<i>Node ID</i>	Output, required. The object to which you want to assign the return value that designates the node ID. The value must be a variable or control with the same data type as the node ID column; a type mismatch will be checked by runtime engine. Set the parameter to <i><Currently Selected Node></i> , <i><Drag Over Node></i> , <i><Currently Expanding/Collapsing Node></i> , or <i><Currently Deleted Node></i> .
<i>Return to</i>	Output, required. The object to which you want to assign the return value that designates the node level. Set the parameter to an applicable object from object list.

Returns

This system function returns the hierarchical level of a given node to the object identified by *Return to*. If this system function fails (for example, the node ID does not exist), then the system function returns a value of -1.

Get Related Node ID

Description

This system function returns the ID of a node related to a reference node.

Parameters

Parameter	Description
<i>Parent Child</i>	Input, required. The parent child FC to affect.
<i>Current node ID</i>	Input, required. The ID of node to use as a reference. Set the parameter to an applicable object from the object list.
<i>Relation</i>	Input, required. The relationship of the target node to the reference node. Set the parameter to <i><Parent Node></i> , <i><First Child Node></i> , <i><Next Sibling Node></i> , <i><Previous Sibling Node></i> .
<i>Node ID</i>	Output, required. The object to which you want to assign the return value that designates the node ID. The value must be a variable or control with the same data type as the node ID column; a type mismatch will be checked by runtime engine. Set the parameter to an applicable object from the object list.
<i>Error Code</i>	Output, required. The object to which you want to assign the return value that designates the error code for the transaction. The value must be integer type variable or control. Set the parameter to an applicable object from the object list.

Returns

This system code returns two values. The node ID itself is returned to the object identified by *Node ID*. A value indicating whether the node was actually found is sent to the object identified by Error Code and can have one of two values:

Value	Description
0	The node ID was found successfully.
Nonzero value	The node ID was not found.

Get Row Number

Description

This system function returns the row number of a given node. You can use this system function only if the control has a node ID column.

Parameters

Parameter	Description
<i>Parent Child</i>	Input, required. The parent child FC to affect.
<i>Node ID</i>	Input, required. The ID of the target node. Set the parameter to an applicable object from the object list.
<i>Row Number</i>	Output, required. The object to which you want to assign the return value that designates the row number.

Returns

This system function returns the next selected row number to the object identified by *Row Number*. If this system function fails (for example, the node ID does not exist), row number is -1.

See Also

Chapter 24, “Understanding Parent Child Controls,” Get Node ID, page 209

Get Selected Context Action

Description

Use this system function to get the action ID of the action that triggered the **OnAction** event. You cannot use this system function in any other event.

Parameters

Parameter	Description
<i>Parent/Child</i>	Input, required. The parent child FC to affect.
<i>Action ID</i>	Output, required. The object to which to assign the value that indicates the action ID of the action that triggered the OnAction event. Set the parameter to an applicable object from the object list.

Returns

This system function returns the ID of the action that triggered the **OnAction** event to the object specified by *Action ID*.

Get Selected Grid Row Count

Description

Use this function to count the number of grid rows in a given range of rows (that is, in the current selection).

Parameters

Parameter	Description
<i>Parent/Child - Grid</i>	Input, required. The parent child FC to affect.
<i>Row</i>	Output, required. The object to which to assign the return value that designates the number of rows. Set the parameter to an applicable object from the object list.

Returns

This system function returns the number of rows in the current selection to the object specified by *Row*.

Get Selected Grid Row Number

Description

Use this function to get the row number for a selected row. Typically, you use this function only when you need to save the row as a variable for future processing.

Parameters

Parameter	Description
<i>Parent/Child - Grid</i>	Input, required. The parent child FC to affect.
<i>Row</i>	Output, required. The object to which to assign the return value that designates the row number. Set the parameter to an applicable object from the object list.

Returns

This system function returns the row number for the selected row to the object specified by *Row*.

Get Tree Node Handle

Description

Use this system function to return the handle of a target node.

Parameters

Parameter	Description
<i>Parent/Child</i>	Input, required. The parent child FC to affect.
<i>Node</i>	Input, optional. The ID of the node for which to get the handle. Leave this parameter blank if you want to identify the node by its position (<i>Index</i> parameter). Set the parameter to <i><Currently Selected Node></i> , <i><Last Inserted Node></i> , <i><Root Node></i> , <i><Currently Expanding/Collapsing Node></i> , or <i><Currently Deleted Node></i> .
<i>Index</i>	Input, optional. The position of the node for which to get the handle. Leave this parameter blank if you want to identify the node by its condition, such as currently selected, last insert, and so forth (<i>Node</i> parameter). Set the parameter to an alphanumeric constant (<i><Literal></i>), <i><Blank></i> , <i><Zero></i> , or an applicable object from the object list.
<i>Handle</i>	Output, required. The object to which to assign the return value that designates the handle of the target node (numeric). Set the parameter to an applicable object from the object list.

Returns

This system function returns the handle for a tree node to the object specified by *Handle*.

Hide Grid Column

Description

Use this system function to hide the entire column.

Parameters

Parameter	Description
<i>Parent/Child - Grid</i>	Input, required. The parent child FC to affect.
<i>Column</i>	Input, required. The column to hide. Set this variable to <i><All Columns></i> .

See Also

Chapter 24, “Understanding Parent Child Controls,” Get Node ID, page 209

Insert Grid Buffer Row

Description

Use this system function to insert a row from the GB into the control. This system function does not take hidden rows into account.

Parameters

Parameter	Description
<i>Parent/Child - Grid</i>	Input, required. The parent child FC to affect.
<i>Row</i>	Input, required. The relative location where to insert the data. Set the parameter to an alphanumeric constant (<Literal>), <After Current Row>, <After Last Row>, <Under Currently Expanding Node>, <Under Drop Node>, <Under Drop Node as First Child>.
<i>Selectable?</i>	Input, required. An indicator of whether the user can select the inserted row. Set the parameter to <Yes> or <No>.
<i>Protectable?</i>	Input, required. An indicator of whether the user can edit the inserted row. Set the parameter to <Yes> or <No>.
<i>Updateable?</i>	Input, required. An indicator of whether runtime will attempt to update the underlying table if the user edits the data and clicks the OK button. Set the parameter to <Yes> or <No>.
<i>Deleteable?</i>	Input, required. An indicator of whether the user can delete the inserted row. Set the parameter to <Yes> or <No>.
<i>Clear After?</i>	Input, required. An indicator of whether runtime should clear the GB automatically immediately after the insert. Set the parameter to <Yes> or <No>.
<i>Expandable?</i>	Input, required. An indicator of whether the user can expand the inserted row. Set the parameter to <Yes> or <No>.

See Also

[Chapter 24, “Understanding Parent Child Controls,” Insert Grid Buffer Row By Node ID, page 214](#)

Insert Grid Buffer Row By Node ID

Description

Use this system function to insert a row from the GB into a location relative to another tree node.

Parameters

Parameter	Description
<i>Parent Child</i>	Input, required. The parent child FC to affect.
<i>Base On Node ID</i>	Input, required. The ID of the target node. Set the parameter to an applicable object from the object list.
<i>Insert As</i>	Input, required. The position to insert the GB row, relative to the target node. Set the parameter to insert the row immediately under the target node and make it a child of the target <First Child Node>, make the row a child of the target node and insert it at the bottom of the target's children <Last Child Node>, insert the row immediately after the target node <Next Sibling Node>, or insert the row immediately before the target node <Previous Sibling Node>.
<i>Selectable</i>	Input, required. An indicator of whether the user can select the inserted row. Set the parameter to <Yes> or <No>.
<i>Protectable</i>	Input, required. An indicator of whether the user can edit the inserted row. Set the parameter to <Yes> or <No>.
<i>Updateable</i>	Input, required. An indicator of whether runtime will attempt to update the underlying table if the user edits the data and clicks the OK button. Set the parameter to <Yes> or <No>.
<i>Deleteable</i>	Input, required. An indicator of whether the user can delete the inserted row. Set the parameter to <Yes> or <No>.
<i>Clear After?</i>	Input, required. An indicator of whether runtime should clear the GB automatically immediately after the insert. Set the parameter to <Yes> or <No>.
<i>Expandable?</i>	Input, required. An indicator of whether the user can expand the inserted row. Set the parameter to <Yes> or <No>.
<i>Insert Mode</i>	Input, required. An indicator of whether the new row is to be inserted into the database or to be updated into the database (assuming such a row already exist in the database), if the parent child control has a business view. If the insert mode is Update and no such row exists in the database, the system will fail to process this row. If the insert mode is Insert and a row already exists in the database, the system will fail to process the row in the database. Set the parameter to <Insert> or <Update>.

Additional Notes

You can insert the row before or after the target node. Alternatively, you can insert the row as a child of the target node. In that case, you can choose to insert it as the first or the last child.

See Also

Chapter 24, “Understanding Parent Child Controls,” Insert Grid Buffer Row, page 213

Set Action

Description

Use this system function to enable or disable a specific context-based action in Product Synch mode.

Parameters

Parameter	Description
<i>Parent/Child</i>	Input, required. The parent child FC to affect.
<i>Action ID</i>	Input, required. The ID of the action to enable or disable. Set the parameter to <i><Enabled></i> , <i><Disabled></i> , or an applicable object from the object list.
<i>State</i>	Input (EVDT_INT), required. An indicator of the state (enabled or disabled) of the action. Set the parameter to <i><Selected></i> (1), <i><Deselected></i> (0), or applicable object from the object list

Additional Notes

Note that *State* can be enabled, disabled, or you can pass in an EVDT_INT variable which will be interpreted to provide the value for the selection state. This makes it possible to reduce if/then logic because one system function can be used to update the enabled state for a context based action.

Set Data Dictionary Item

Description

Use this system function to override form controls and grid columns that are data dictionary (DD) items. This is a complete change; you substitute a different DD item for the existing one. You can also create a new DD item that is not the same type as the old item.

Parameters

Parameter	Description
<i>Parent/Child - Grid</i>	Input, required. The parent child FC to affect.
<i>Column</i>	Input, optional. The column to hide. Because parent child controls have only one column, you do not need to enter a value into this parameter. Set to <i><All Columns></i> .
<i>DD Alias</i>	Input, required. The alias of the DD item that you want to use. Set the parameter to a specific DD alias (<i><Pick DD Item></i>) or an applicable object from the object list
<i>System Code</i>	Input, required. The system code to use when checking for textual overrides to apply to the DD item. Set to <i><default></i> .

Set Data Dictionary Overrides

Description

Use this system function to change a specific DD item property.

Parameters

Parameter	Description
<i>Parent/Child - Grid</i>	Input, required. The parent child FC to affect.
<i>Column</i>	Input, optional. The column to hide. Because parent child controls have only one column, you do not need to enter a value into this parameter. Set to <i><All Columns></i> .
<i>Overrides</i>	Input, required. The override to apply. Set the parameter to the specific type of override to apply (<i><Data Dictionary Overrides></i>).

Set Drag Cursor

Description

Use this system function to assign an image to the cursor when the user performs a drag operation.

Parameters

Parameter	Description
<i>Parent/Child</i>	Input, required. The parent child FC to affect.
<i>Cursor</i>	Input, required. The cursor to apply. Set the parameter to <i><Default></i> or <i><No Drag Cursor></i> .

Set Grid Cell Error

Description

Use this system function to set an error on a cell, a tree node, a row, a column or the entire parent child control. This system function does not take hidden rows into account.

Parameters

Parameter	Description
<i>Parent Child</i>	Input, required. The parent child FC to affect.
<i>Row</i>	Input, required. The row on which to set the error. Set the parameter to an alphanumeric constant (<i><Literal></i>), <i><All Rows></i> , or <i><Currently Selected Row></i> .
<i>Column</i>	Input, required. The column on which to set the error. Set to <i><All Columns></i> .
<i>Error Code</i>	Input, required. The error to set. Set the parameter to an alphanumeric constant (<i><Literal></i>), <i><Blank></i> , <i><Zero></i> , or an applicable object from the object list

Set Grid Color

Description

Use this system function to set a color on a cell, a tree node, a row, a column, or the entire control. You can choose a specific color to set, or you can reset the color, which returns the color of the object to its default value. This system function does not take hidden rows into account.

Parameters

Parameter	Description
<i>Parent/Child - Grid</i>	Input, required. The parent child FC to affect.
<i>Row</i>	Input, required. The relative row on which to set the color. Set the parameter to an alphanumeric constant (<Literal>), <All Rows>, <Currently Selected Row>, applicable object from the object list.
<i>Column</i>	Input, required. The column on which to set the color. Set to <All Columns>.
<i>Color</i>	Input, required. The color to display. Set the parameter to a color from the color palette (<Pick Color>) or the default color <Reset Color>.

Set Grid Column Heading

Description

Use this system function to change the text in a column heading.

Parameters

Parameter	Description
<i>Parent/Child - Grid</i>	Input, required. The parent child FC to affect.
<i>Column</i>	Input, required. The column for which to change the header. Set to <All Columns>.
<i>Text</i>	Input, required. The text to use for the column header. Set the parameter to alphanumeric constant (<Literal>), <Blank>, <Zero>, or applicable object from the object list.

Set Grid Font

Description

Use this system function to set a font for the text in a cell, a tree node, a row, a column, or the entire control. Font in this context includes font type (such as Arial or Times New Roman), style (such as italic or bold), size, effects (strikeout or underline), and color. You can choose a specific font to set, or you can reset the font which returns the font of the object to its default value. This system function does not take hidden rows into account.

Parameters

Parameter	Description
<i>Parent/Child - Grid</i>	Input, required. The parent child FC to affect.
<i>Row</i>	Input, required. The relative row for which to set the font. Set the parameter to an alphanumeric constant (<Literal>), <All Rows>, <Currently Selected Row>, applicable object from the object list.
<i>Column</i>	Input, required. The column on which to set the font. Set to <All Columns>.
<i>Font</i>	Input, required. The font to be set on the row and column. Set the parameter to a font and related settings from the Font dialog (<Pick Font>) or the default font <Reset Color>.

See Also

Chapter 24, "Understanding Parent Child Controls," Insert Grid Buffer Row, page 213

Set Grid Row Bitmap

Description

Use this system function to manipulate the bitmap icon that appears next to the row. This function sets a specific system bitmap, such as a check mark or a trash can, to use as an icon on a specified row header. This system function does not take hidden rows into account.

Parameters

Parameter	Description
<i>Parent/Child - Grid</i>	Input, required. The parent child FC to affect.
<i>Row</i>	Input, required. The relative row for which to set the bitmap. Set the parameter to an alphanumeric constant (<Literal>), <All Rows>, <Currently Selected Row>, applicable object from the object list.
<i>Bitmap</i>	Input, required. The bitmap to be set on the row. Set the parameter to the specific bitmap that you want to apply (<Checkbox>, <X Mark>, and so forth).

Set QBE Column Compare Style

Description

When the application performs a QBE search, this system function permits the application to set which comparison operator to use. For example, if you want to set up a date column QBE field to be > January 1, 2005, you would use QC WorkDate = "010105" and Set QBE Column Compare Style(FC Parent/Child, GC WorkDate, <Greater Than>).

Parameters

Parameter	Description
<i>Parent Child</i>	Input, required. The parent child FC to affect.
<i>Column</i>	Input, required. The column on which to set the QBE comparison operator. Set to <i><All Columns></i> .
<i>Compare Style</i>	Input, required. The comparison operator to use. Set the parameter to <i><Equal To></i> , <i><Not Equal To></i> , <i><Greater Than></i> , <i><Less Than></i> , <i><Greater Than or Equal To></i> , or <i><Less Than or Equal To></i> .

Set Tree Bitmap Scheme

Description

Use this system function to assign bitmap icons to represent the different states that a tree node might be in (open/expanded, closed/collapsed, or leaf/no children).

Parameters

Parameter	Description
<i>Parent/Child</i>	Input, required. The parent child FC to affect.
<i>Open Bitmap</i>	Input, required. The icon to use when the node is in an expanded state; that is, all of its children are displayed. Set the parameter to <i><Choose Tree Bitmap></i> or <i><Default></i> .
<i>Closed Bitmap</i>	Input, required. The icon to use when the node is in a collapsed state; that is, all of its children are hidden. Set the parameter to <i><Choose Tree Bitmap></i> or <i><Default></i> .
<i>Leaf Bitmap</i>	Input, required. The icon to use when the node has no children and therefore cannot be expanded or collapsed. Set the parameter to <i><Choose Tree Bitmap></i> or <i><Default></i> .

Set Tree Node Bitmap

Description

This system function replaces the default bitmap for a node with one of your choosing.

Parameters

Parameter	Description
<i>Parent/Child</i>	Input, required. The parent child FC to affect.
<i>Node</i>	Input, required. The node or row for which to replace bitmap. This value cannot be 0. Set the parameter to <i><Currently Selected Node></i> , <i><Last Inserted Node></i> , <i><Currently Expanding/Collapsing Node></i> .
<i>Bitmap</i>	Input, required. The bitmap image to display. Set the parameter to <i><Choose Tree Bitmap></i> or <i><None></i> .
<i>Overlay?</i>	Input, required. A flag that indicates whether to superimpose the image on the base bitmap image. Overlays are displayed on Windows clients only. Set the parameter to <i><Yes></i> or <i><No></i> .

See Also

Chapter 24, “Understanding Parent Child Controls,” Set Tree Node Clickable Bitmap, page 221

Set Tree Node Bold

Description

This system function sets the text associated with a tree node in bold type face.

Parameters

Parameter	Description
<i>Parent/Child</i>	Input, required. The parent child FC to affect.
<i>Node</i>	Input, required. The tree node to affect. Set the parameter to <i><Currently Selected Node></i> , <i><Last Inserted Node></i> , <i><Currently Expanding/Collapsing Node></i> .
<i>Bold</i>	Input, required. A flag that indicates whether to apply or remove bold type face formatting to the text of the indicated node. Set the parameter to <i><Yes></i> or <i><No></i> .

Set Tree Node Clickable Bitmap

Description

This system function displays an additional node bitmap to the specified tree node or row. The bitmap is clickable (clicking the bitmap causes the event to fire: **Tree Node Bitmap Is Clicked**). The additional bitmap appears to the left of the regular bitmap, so after this system function is called, the tree node has two bitmaps.

Parameters

Parameter	Description
<i>Parent/Child</i>	Input, required. The parent child FC to affect.
<i>Node</i>	Input, required. The node or row to which to add the clickable bitmap. This value cannot be 0. Set the parameter to <i><Current Selected Node></i> , <i><Last Inserted Node></i> , or an applicable object from the object list
<i>Bitmap</i>	Input, required. The bitmap image to display. Set the parameter to <i><Choose Tree Bitmap></i> or <i><None></i> .

See Also

Chapter 24, “Understanding Parent Child Controls,” Set Tree Node Bitmap, page 220

Set Tree Node Handle

Description

Use this system function to set the handle of a target node.

Parameters

Parameter	Description
<i>Parent/Child</i>	Input, required. The parent child FC to affect.
<i>Node</i>	Input, optional. The ID of the node for which to get the handle. Leave this parameter blank if you want to identify the node by its position (Index parameter). Set the parameter to <i><Current Selected Node></i> , <i><Last Inserted Node></i> , <i><Root Node></i> , or <i><Currently Expanding/Collapsing Node></i> .
<i>Index</i>	Input, optional. The position of the node for which to get the handle. Leave this parameter blank if you want to identify the node by its condition, such as currently selected, last insert, and so forth (<i>Node</i> parameter). Set the parameter to an alphanumeric constant (<i><Literal></i>), <i><Blank></i> , <i><Zero></i> , or an applicable object from the object list
<i>Handle</i>	Input, required. The value to use for the handle for the target node. Set the parameter to an applicable object from the object list.

Set Tree Root Node ID

Description

This system function assigns a node ID value to the hidden root node. You can use this system function only if the control has a node ID column.

Parameters

Parameter	Description
<i>Parent Child</i>	Input, required. The parent child FC to affect.
<i>Node ID</i>	Input, required. A variable/control with compatible data type (same type as the node ID column) that holds a value. A type mismatch will be checked by the runtime engine. Set the parameter to an applicable object from the object list.

Show Grid Column

Description

Use this function to show the entire column.

Parameters

Parameter	Description
<i>Parent/Child - Grid</i>	Input, required. The parent child FC to affect.
<i>Column</i>	Input, required. The column to hide. Set to <i><All Columns></i> .

See Also

Chapter 24, “Understanding Parent Child Controls,” Hide Grid Column, page 213

Show N Levels

Description

Use this system function to expand or collapse a tree branch or the entire tree to a uniform hierarchical level.

Parameters

Parameter	Description
<i>Parent Child</i>	Input, required. The parent child FC to affect.
<i>Node/Row</i>	Input, required. The ID of the node or row under which you want to affect. Choose <i><Root Node></i> if you want to affect the entire tree. Set the parameter to <i><Root Node></i> or an applicable object from the object list.
<i>Level</i>	Input, required. The hierarchical level to which to expand or collapse the target node or row's children. Set the parameter to an applicable object from the object list.

Suppress Fetch On Node Expand

Description

Whenever a node is expanded, the system function, **Suppress Fetch on Node Expand**, is called from the event, **Tree Node Is Expanded**. This function tells the runtime engine not to do any fetches because ER will handle the loading of the child nodes.

Parameters

Parameter	Description
<i>Parent Child</i>	Input, required. The parent child FC to affect.

Suppress Grid Line

Description

Use this function to prevent a row from becoming part of the grid. For example, use this system function on the **Write Grid Line Before** event to prevent the line from being written to the grid.

Parameters

Parameter	Description
<i>Parent/Child - Grid</i>	Input, required. The parent child FC to affect.

Suppress Node Indent/Outdent

Description

This system function prevents users from being able to move nodes horizontally within the tree structure.

Parameters

Parameter	Description
<i>Parent Child</i>	Input, required. The parent child FC to affect.

See Also

[Chapter 24, “Understanding Parent Child Controls,” Suppress Node Move Up/Down, page 224](#)

Suppress Node Move Up/Down

Description

This system function prevents users from being able to move nodes vertically within the tree structure.

Parameters

Parameter	Description
<i>Parent Child</i>	Input, required. The parent child FC to affect.

See Also

[Chapter 24, “Understanding Parent Child Controls,” Suppress Node Indent/Outdent, page 224](#)

Update Grid Buffer Row

Description

Use this system function to update a row from the GB into the grid control. This system function does not take hidden rows into account.

Parameters

Parameter	Description
<i>Parent/Child - Grid</i>	Input, required. The parent child FC to affect.
<i>Row</i>	Input, required. The relative row to affect. Set the parameter to an alphanumeric constant (<Literal>), <Currently Selected Row>, <Currently Expanding/Collapsing Node>, or an applicable object from the object list.
<i>Selectable?</i>	Input, required. An indicator of whether the user can select the updated row. Set the parameter to <Yes> or <No>.
<i>Protected?</i>	Input, required. An indicator of whether the user can edit the updated row. Set the parameter to <Yes> or <No>.
<i>Updateable?</i>	Input, required. An indicator of whether runtime will attempt to update the underlying table if the user edits the data and clicks the OK button. Set the parameter to <Yes> or <No>.
<i>Deleteable?</i>	Input, required. An indicator of whether the user can delete the updated row. Set the parameter to <Yes> or <No>.
<i>Clear After?</i>	Input, required. An indicator of whether runtime should clear the GB automatically immediately after the update. Set the parameter to <Yes> or <No>.

Was Grid Cell Value Entered

Description

This system function returns a nonzero value if a specific grid cell or tree node has been changed since last time this system function is called.

Parameters

Parameter	Description
<i>Parent Child</i>	Input, required. The parent child FC to affect.
<i>Column</i>	Input, required. The column containing the grid cell to check. Set to <All Columns>.
<i>Return To</i>	Input, required. The object to which to assign the return value that designates whether the cell or node has changed. Set the parameter to an applicable object from the object list.

CHAPTER 25

Understanding Push Button Controls

This chapter describes push button controls.

Push Button Controls

Use a push button to initiate an action or a set of actions. You can designate a single push button to be the one that is activated when the user presses ENTER by enabling the Default Pushbutton property.

Message forms have a push button on them by default. The button is configurable.

Subforms and message forms do not have a tool bar; instead, users must use push buttons. The push buttons behave the same as the standard push buttons for other form types.

Push Button Events

These events are the only ones that can fire when the user clicks a push button:

- **Button Clicked**
- **Post Button Clicked**
- **Post Button Clicked - Asynch**

The first two always fire when a user clicks a button, and they fire in succession. That is, **Post Button Clicked** fires after runtime executes any logic that might exist on **Button Clicked**. **Post Button Clicked - Asynch** fires only when the button is an OK button.

CHAPTER 26

Understanding Radio Button Controls

This chapter describes radio button controls.

Radio Button Controls

Use radio buttons to indicate choices. Selecting a radio button indicates which function in a set of functions is to be enabled. Radio button option sets should always be mutually exclusive. Enclose a set of radio button controls in a group box control to make them function as a single unit.

You can associate radio buttons with a user-defined code (UDC) field, where each button has a value from the UDC table. You must associate a radio button with a database or data dictionary (DD) item.

Radio Button Design-Time Considerations

A standard use case for a group of radio buttons is to provide users with a choice of operators to use in a query. If you want to use radio buttons for this purpose, use the Filter Criteria property options.

By default, the control returns a Boolean numeric value to indicate its state (0 for deselected and 1 for selected). However, you can change the return value for a radio button control with the Value property.

Radio Button Events

Only one event occurs on a radio button control: **Selection Changed**. Runtime fires the event when the user changes the status of a radio button. Consequently, **Selection Changed** should occur in pairs for radio buttons: Once for the selected control and once for the control that the system deselects.

CHAPTER 27

Understanding Saved Query Controls

This chapter describes saved query controls.

Saved Query Controls

With the saved query control, you provide users with the means to persist query criteria for use against the business view (BV) of the form. The control provides flexibility in the query filters and logic. You can give users as little leeway as choosing a query from a preset list to as much as giving them to input and save the query parameters themselves.

The saved query control is a regular form control and can be added to a form as such, either from the Insert menu or tool bar. Because you must query on a BV, do not place a saved query control on a form type that does not permit BVs (that is, message forms) or on forms where the BV fetch is disabled. All filtering occurs on the BV columns that you choose. A single BV column may be used as a filter more than once on the same form. You can display the query results in a grid or parent child control.

The query control appears by default with two parts: a titled drop-down field and two links (Save Query and Edit Queries). With the control properties, you can opt to display the links to the left of the combo box or beneath it. You can also choose to hide the links. Saved queries reside in the system as user overrides. You can use Cross-Reference and Object Management Workbench (OMW) to find and manipulate them as such. Because they are user overrides, saved queries are translatable.

In brief, these are the steps you perform to implement a saved query control:

1. Start with a form that has one or more BVs attached to it.
2. Place the saved query control on the form and configure it.
3. Place control on the form intended to display query results (that is, a grid or a parent/child control).
4. If you want a BV column to act as a filter, add an edit control for it and then configure it to be filterable by choosing a comparison operator to use.
5. If you want to enable users to choose their own comparison type, add an edit control for each one and configure it with the Runtime Select comparison type.
6. If you want to provide users with a defined list of queries, run the application and make and save the queries (which are saved as user overrides).

Then, find and copy the user overrides into the *PUBLIC list.

Saved Query Control Design-Time Considerations

This table identifies design-time property values you can set that can impact the saved query control significantly:

Desired Result	What To Do
Make links appear to the right of the control.	Set the saved query parameter, Position of Saved Query links, to <i>Right of dropdown list</i> .
Make links appear under the control.	Set the saved query parameter, Position of Saved Query links, to <i>Below dropdown list</i> .
Hide the links.	Set the saved query parameter, Position of Saved Query links, to <i>No links</i> .
Prevent a user from seeing (and therefore changing) a filter field when editing a saved query.	Set the grid column parameter, Allowed in Saved Query, to <i>No</i> .

Runtime Processing for the Saved Query Control

The saved query control is initialized as part of the form initialization process. To initialize the control, runtime checks the user override table for all saved query records that match the current application/form/version combination, user ID, current role or roles (*ALL), and *PUBLIC. If any records are found, runtime populates the drop-down field with the saved query names in alphabetical order.

In some cases, runtime might find duplicate query records between roles. For example, if a single user's query was copied so that it could be accessed by *PUBLIC, then runtime would discover that the query criteria are identical. In this case, runtime returns only the proprietary record (the user's query rather than the *PUBLIC version of the query). In all other instances, runtime returns one of the saved queries without preference as to which one it returns. Notice in this example that the entire saved query record was a duplicate. The saved query control allows for duplicate saved query names; runtime can distinguish between queries with the same name with different criteria (non-duplicates) and queries with the same name with identical criteria (duplicates).

After initialization, runtime interacts with the saved query control only when the user works with queries. If the user chooses a query from the drop-down list, runtime first clears all of the fields that have been flagged with the property value, Allow in Saved Query (including hidden columns and filter fields). Then runtime populates the fields as dictated by the saved query. Special values that require calculation such as user ID, today's date, and so forth, are calculated at the time that they are applied into the field. The act of populating the fields does not instigate a fetch; instead, the user must click Find to instruct the engine to fetch data based on the retrieved saved query.

When a user creates or modifies a saved query, runtime does not write those records to the user override table until form close. On the other hand, when a user deletes a saved query, runtime purges that record from the table immediately.

CHAPTER 28

Understanding Static Text Controls

This chapter discusses static text controls.

Static Text Controls

Use a static text control to display descriptive text, such as a title or instructions. This text is not associated with a control, and the user cannot change it. You can change it during runtime using the **Set Control Text** system function in event rules. You also can make static text clickable by enabling the Clickable property. When a user clicks clickable text, runtime fires the **Text Clicked** event.

No runtime processing occurs on this control.

CHAPTER 29

Understanding Subforms and Subform Aliases

This chapter provides an overview of subforms and describes how to:

- Create subforms.
- Reuse subforms.
- Work with data structures and subforms.
- Work with functions and subforms.

Understanding Subforms

A subform is a control designed for use on a power form or another subform. Although technically a control, subforms have some form characteristics as well. Each subform represents one data view; that is, each subform control can have a single business view (BV) attached to it. Power forms can contain several subforms, so a single power form with multiple subforms enables users to see multiple data views. For example, a user selecting a purchase order in a grid could see related shipping information in one subform and fulfillment information in another subform on the same power form. When the user selects a row in the grid, all of the data is updated and, most importantly, the user does not have to open a new form to see the updated data.

Together, power forms and subforms provide developers with the ability to code:

- Multiple views on a form.
- Multiple grids on a form.
- Multiple tab controls on a form.
- Tab pages with their own business view.
- BVs that can communicate with each other, and even react to selection and data changes that occur in other views on the form.

Subforms have two main characteristics:

- To its parent, it is a control.
- To the controls that it contains, it is a form.

When you place subforms on a power form or subform, the system treats the interactions among them as parent/child relationships, with the power form or subform as the parent and the subforms as the children of the power form and siblings to each other. This hierarchical relationship governs logical relationships. It is also represented visually in the Form Design Aid (FDA) Application Tree View as a hierarchy to help you understand the relationships.

Similarly to a form, a subform owns the data flows between the interface view and the database, including browse, insert, update, or delete. In fact, you create a subform as if it were a form, and then place it as a control. Do not let its appearance fool you, however. A subform is really a control and the FDA interface treats it accordingly. For example, if subform B is contained in power form A, when you select B, the FDA control bar still displays the title of A. Additionally, if you view the data structure, you will see the data structure for A and not for B.

Understanding Subform Design-Time Considerations

This list describes many of the conditions and factors that you should take into account while designing the subform in FDA:

- Subforms have their own BVs (only one per subform), grids, and filters that pass logic between other subforms.
- Subforms can be used as tab pages, or they can be placed on a tab page.
- Multiple tab controls are permitted unless the subform is inside a tab page.

If the subform is inside a tab page, it cannot have tab controls. However, this restriction cannot be enforced programmatically when you reuse subforms. Therefore, when developing reusable subforms, carefully consider the context in which you expect to use them to prevent poor user interface design.

- Toolbar and form/row exits are not permitted on a subform.
- Subforms do not contain scroll bars; you must display all controls within the form.
- You can add action buttons (Cancel, OK, and so forth) to a subform.

No default action buttons are defined by FDA. Default actions can be placed anywhere on the subform. For example, the default action, Save, can be placed beneath the grid. Use the dotted lines at the top and bottom of a subform during design time as guides for where to place more common default actions. For example, buttons such as Select or Find belong at the top of the form while buttons such as Save or Cancel belong at the bottom.

After designing the subform, you must return to the parent form and map the parent data fields to the child so that the two can share data as you intend. After associating a data structure with the parent, use the Mapping Links property to establish the data mapping between the parent and subform.

Note. Because subforms can act as parents, they have the Mapping Links property. However, you can only create mappings on the parent level. If you cannot find the child subform in the Link To field, then you might have mistakenly entered the data mapping property for the child instead of the parent.

These property values are significant for subforms:

- Show Subform Header

This property enables you to hide or show the title bar of the subform. You must show the header if you want to make the subform collapsible (with the Collapsible property).

- Collapsible

This property determines whether the user can choose to view just the title bar (header) of the subform. The ability to collapse subforms is useful on forms that contain a large number of subforms. You must show the header (with the Show Header property) if you want to make the subform collapsible.

- Transaction

This property determines where the subform falls within the transaction boundary for the form as a whole. Additionally, subforms support the form-level properties: *Update on Businessview* and *Fetch on Businessview*. Finally, to enhance reuse, you can associate any number of functions with a subform at design time. Then, during runtime, the parent of the subform can invoke any of those functions with the **Call Function** system function.

See Also

[Chapter 29, “Understanding Subforms and Subform Aliases,” Working with Data Structures and Subforms, page 242](#)

[Chapter 29, “Understanding Subforms and Subform Aliases,” Working with Functions and Subforms, page 242](#)

Understanding Subform Events

These events can fire on the subform during runtime:

- **Notified by Child**
- **Notified by Parent**
- **Tab Page is Initialized**
- **Tab Page is Selected**
- **Row Is Selected**
- **Grid Record is Fetched**
- **Write Grid Line-Before**
- **Write Grid Line-After**
- **Row Is Unselected**
- **Last Grid Record Has Been Read**
- **Add Record to DB - Before**
- **Add Record to DB - After**
- **Update Record to DB - Before**
- **Update Record to DB - After**
- **Post Commit**

Additionally, a number of events that start with WIZARD can fire on this control as well, but only when on a wizard form.

Notified by Child and **Notified by Parent** fire when the child sends data to the parent and vice versa, respectively. Use the **Notified by Parent** event to run any business logic that you want the application to execute after the parent of the subform calls the **Notify Child** system function. This logic is usually based on the information passed to this subform from its parent. Use the **Notified by Child** event to run any business logic that you want the application to execute after its child calls the **Notify Parent** system function. This logic is usually based on the information returned from its child.

See Also

[Chapter 34, “Understanding Wizard Controls,” page 269](#)

Understanding Subform Runtime Processing

You use subforms to browse or update records using one BV; you cannot place more than one BV on a subform. Subforms can contain most FDA controls (with the notable exception of the parent child control) and may or may not have a grid.

This section describes how runtime processes the subform control.

Control Initialization

Each subform ultimately resides on a parent power form. When the power form initializes, all of its subforms (including the child subforms of its subforms) are initialized. First, runtime initializes these objects in order:

- Business view columns (BC).
- Subform controls.
- Static text.
- Helps.
- Event rules (ER) structures.

The, runtime begins detail data selection and sequencing if the grid option, Automatically Find On Entry, is enabled.

Subform Push Buttons

Subforms do not have a tool bar; instead, you must add push buttons to perform “standard” functions. The push buttons behave the same as the standard push buttons for other form types.

If a subform is on a power browse form, these push buttons are available:

- Select
- Find

If a subform is on a power edit form, these push buttons are available:

- Clear
- Find
- Delete
- Save

Find

Find is a standard push button that is available on all subforms. When the user clicks it, runtime fires the **Button Clicked** event. If no errors exist in the filter fields, runtime performs data selection and sequencing for the grid or other control if no grid is present. If one or more records are fetched, then runtime loads the control sets the mode to Update. If no records were fetched and a grid is present, runtime sets the mode to Add. Then runtime fires the **Post Button Clicked** event.

Select

Select is a standard push button that is available on subforms placed on power browse forms. When clicked, runtime performs these actions:

1. Load the mapping link value from the subform parent.
2. Fire **Button Clicked**.
3. Fire **Post Button Clicked**.

Clear

Clear is a standard push button that is available on subforms placed on power edit forms. When clicked, runtime performs these actions:

1. Fire **Button Clicked**.
2. Clear all controls on the subform.
3. Clear errors on the subform.
4. Fire **Post Button Clicked**.

Delete

Delete is a standard push button that is available on all subforms placed on power edit forms. The actual delete from the database does not occur at this point. Runtime verifies the intention to delete when the user clicks the Delete button, and then commits the deletion when the user clicks the Save button. Consequently, if the user clicks the Cancel button, the records are not purged from the database.

Save

Save is a standard push button that is available on all subforms placed on power edit forms. It validates the information on the subform and updates or adds to the database.

Note. If the subform contains a grid control, the save button processing behaves similarly to the OK button on a headerless/detail form. If the subform does not contain a grid, the Save button processing behaves similar to the OK button on a fix/inspect form.

See Also

[Chapter 8, “Understanding Fix/Inspect Forms,” Fix/Inspect Runtime Processing, page 59](#)

[Chapter 9, “Understanding Header Detail Forms,” Header Detail Runtime Processing, page 66](#)

Creating Subforms

This section provides an overview of subform creation describes how to:

- Create a subform without a power form.
- Create a subform on a power form.
- Create a subform as a tab page.

Understanding Subform Creation

You can create a subform in one of two ways: without a power form, or as a control either directly on a power form or on the tab page of a power form. These two options give you the flexibility to code subforms and power forms independently to accommodate different design schedules. Therefore, you might choose to create a subform outside of the context of a power form if the power form itself is not ready yet or if your group is responsible only for creating a subform that other groups will use on their power forms, for example. Furthermore, you can only create browse subforms directly on a power form, so if you want to create an edit subform, you must create it independently of a given power form.

No matter how you create the subform, it exists as its own entity in the system. Therefore, you can find and insert any subform onto any power form.

When you create it, a blue hashed line appears at the top and bottom of a subform. Use these lines as guides for button placement. Place initial actions, such as find, at the top.

Place concluding actions, such as save, near the bottom. Controls may only exist within the boundary of the subform; you cannot size the subform to be smaller than the area where its child controls reside. This restriction includes hidden controls, so if FDA does not permit you to shrink a subform, ensure that you have no hidden controls preventing the resize.

Creating a Subform without a Power Form

To create a subform without a power form:

1. In FDA, choose a subform type (reusable browse or reusable edit) from the Form menu under Create.
2. On Subform Properties, configure the properties for the subform.

Note. When you save the subform, you are actually creating an application to contain the subform. Keep that in mind when you later insert an alias to the subform on a power form. When you search for the subform, you must first search for the application that contains the subform, and then you can select the subform itself.

Creating a Subform on a Power Form

To create a subform on a power form:

1. In FDA, open the power form that you want to use to contain the subform.
2. If you want to insert the subform on a tab page, click the tab control to insert into.
3. Choose Subform from the Insert menu.
4. On Subform Properties, configure the properties for the subform.

5. If you are inserting the subform directly on the power form (and not on a tab page), click to place the subform where you want it on the power form (as you would any other control).

If you later change your mind, you can always drag the subform from a tab page to the main power form or vice versa.

Creating a Subform as a Tab Page

To create a subform as a tab page:

1. In FDA, open the form to which you want to add tab pages.
2. Add a tab page to the form.
3. Choose the tab page and choose Subform or Subform Alias from the Insert menu.

Alternatively, drag a subform from another position on the parent form and drop it on the tab page.

Reusing Subforms

This chapter provides an overview of subform reuse and describes how to reuse a subform on a power form.

Understanding Subform Reuse

Subforms can exist as discrete objects in the system. You can place the subform itself on a power form; this action is referred to as *embedding*. You can also reference an existing subform on a power form with an *alias*; this action is referred to as *reusing*. Typically, developers reuse subforms that are designed to appear on numerous forms, such as the display of address book information. In this way, you can more easily standardize the interface.

A reused subform is actually just a pointer; therefore, it is unaware of its parent and children in any given context. If you want a reused subform to communicate with its parent or children, you must do so through ER. While you cannot embed a subform within another subform, you can reuse a subform within another subform. In other words, you can embed or reuse a subform within a reusable subform, but you cannot embed or reuse a subform within an embedded subform.

Inserting reusable subforms in a form is different from inserting an embedded subform. They exist as two different control types in the user interface. Therefore, the FDA Menu/Toolbars contain two different insert actions. If you insert an alias (that is, an embedded subform), you are prompted for the application. If you insert a reusable subform, you are prompted for the subform. You cannot insert a subform onto a form until the application the subform has been defined in has been saved.

To make reusing subforms effective, you must plan carefully and be cautious when altering subforms. For example, if both a parent and a child are reused, then you must set up their data mappings in a way that facilitates their passing information to each other. You should use variables as often as possible, especially for business view columns (BCs), grid columns (GCs), and form controls (FCs). Establish a naming convention for the variables so that you can determine which ones were created for the purpose of mapping. That way, if another developer wants to reuse the subform and sees several "extra" variables, that developer will understand that the variables are not extraneous.

Knowing that any subform can be reused, you must be careful when changing a subform, even just resizing it. Anyone who is currently pointing to the subform will see the changes you make. It is possible that if you make the subform a little larger that it will no longer fit on someone else's form. Even worse, if you change the data structure by removing elements from it, you might break someone else's application.

Note. Always determine the full affect of changing a subform, especially its data structure. If you reuse subforms in the applications, check them occasionally to ensure that the subforms have not been altered in such a way as to negatively affect the applications.

Reusing a Subform on a Power Form

To insert (reuse) a subform on a power form:

1. In FDA, open the power form that you want to use to contain the subform.
2. If you want to insert the subform on a tab page, click the tab control to insert into.
3. Choose Subform Alias from the Insert menu.
4. On Work with Applications, choose the application containing the subform you want to insert.
5. On Work with Subforms, choose the subform you want to insert.

The system inserts an alias to the subform on the power form or tab page.

Note. Always determine the full affect of changing a subform. If you reuse subforms in the applications, check them occasionally to ensure that the subforms have not been altered in such a way as to negatively affect the applications.

Working with Data Structures and Subforms

Subforms are self-contained; the elements within it cannot be accessed by other subforms or the form. Therefore, to communicate with each other, each subform must present an interface that can be used to pass data into and out of the subform. The interface is presented as a data structure and work similarly to form and report interconnects. When developing event rules, the data structure items will appear as variable type subform interconnection (SI).

Mapping a Parent's Variables to a Child Subform

To map parent variables to a child subform:

1. Select the parent and choose Form Properties from the Form menu.
2. Click the Mapping Link tab.
3. In the Link To field, choose the child subform to which you want to map data.
You cannot map to a grandchild.
4. Choose an element of the child interface and map a variable to it.

Note. Changing the parent assignment after mapping links have been set or event rules have been developed could cause the application to fail.

Working with Functions and Subforms

To enhance reuse, you can associate any number of functions with a subform at design time. Then, during runtime, the parent of the subform can invoke any of those functions with the **Call Function** system function.

Adding a Function to a Subform

To add a function to a subform:

1. Click the subform to which you want to add the function and choose Form, Create, Function.

The Application Tree View browser switches to the Logical Hierarchy view, and the function appears in the tree as a child object of the subform. (Functions do not appear in the tree if set to the Physical Hierarchy view.)

2. Right-click the function in the tree and choose Edit ER.

The Event Rules Design form appears.

3. Create the logic for the function as you would for an ER.

Subform System Functions

This section describes the system function unique to subforms.

Call Function

Description

This system function enables the parent to invoke any of the functions on any of its children.

Parameters

Parameter	Description
<i>Subform</i>	Input, required. The subform to affect.
<i>Function</i>	Input, required. The function to invoke. Set the parameter to an applicable object from the object list.

See Also

Chapter 29, “Understanding Subforms and Subform Aliases,” Working with Functions and Subforms, page 242

Enable Subform

Description

This system function makes the subform available for user entry and system use.

Parameters

Parameter	Description
<i>Subform</i>	Input, required. The subform FC to affect. If the subform has children, you can enable all of its children as well.

Disable Subform

Description

This system function makes the subform unavailable for user entry and system use. You cannot invoke functions on a disabled subform.

Parameters

Parameter	Description
<i>Subform</i>	Input, required. The subform FC to affect. If the subform has children, you can disable all of its children as well.

Hide Subform

Description

This system function hides the subform from the user, although the system can still access it. For example, you can invoke functions on a hidden subform.

Parameters

Parameter	Description
<i>Subform</i>	Input, required. The subform FC to affect. If the subform has children, you can hide all of its children as well.

Show Subform

Description

This system function displays a previously-hidden form to the user.

Parameters

Parameter	Description
<i>Subform</i>	Input, required. The subform FC to affect. If the subform has children, you can show all of its children as well.

Update Parent

Description

This system function causes the data in the data structure to be distributed appropriately. It has no parameters.

Notify Parent

Description

This system function triggers the **Notified by Child** event. It has no parameters.

Get Error Count

Description

This system function returns the number of errors that are set on a subform, including its children, if desired.

Parameters

Parameter	Description
<i>Subform</i>	Input, required. The subform FC to affect. If the subform has children, you can include the errors set on them in the count as well.
<i>Number</i>	Input, required. The object to which to assign the return value. Set the parameter to an applicable object from the object list.

Returns

This system function returns the total number of errors set on the objects indicated. Runtime returns the value to the object indicated by *Number*.

Get Warning Count

Description

This system function returns the number of warnings that are set on a subform, including its children, if desired.

Parameters

Parameter	Description
<i>Subform</i>	Input, required. The subform FC to affect. If the subform has children, you can include the warnings set on them in the count as well.
<i>Number</i>	Input, required. The object to which to assign the return value. Set the parameter to an applicable object from the object list.

Returns

This system function returns the total number of warnings set on the objects indicated. Runtime returns the value to the object indicated by *Number*.

Get Subform ID

Description

This system function acquires the ID of the child subform relative to the current form.

Parameters

Parameter	Description
<i>Subform ID</i>	Input, required. The object to which to assign the return value. Set the parameter to an applicable object from the object list.

Returns

This system function returns the ID of the child subform to the object specified by *Subform ID*.

Notify Child

Description

This system function triggers the **Notified by Parent** event.

Parameters

Parameter	Description
<i>Subform</i>	Input, required. The child that you want to contact. Set the parameter to <i><All Children></i> or an applicable object from the object list.

Trigger Default Action

Description

This system function enables you to “push a button” on a subform programmatically.

Parameters

Parameter	Description
<i>Subform</i>	Input, required. The subform FC to affect.
<i>Default Action</i>	Input required. The “button” to “push.” Set the parameter to <i><Subform Save></i> , <i><Subform Delete></i> , <i><Subform Find></i> , <i><Subform Clear></i> , <i><Subform Select></i> .

Expand Subform

Description

This system function expands a subform if it is currently collapsed. It has no effect if the Collapsible property is disabled.

Parameters

Parameter	Description
<i>Subform</i>	Input, required. The subform FC to affect. Set the parameter to <i><Current Subform></i> , <i><All Children></i> , or an applicable object from the object list.

Collapse Subform

Description

This system function collapses a subform if it is currently expanded. It has no effect if the Collapsible property is cleared.

Parameters

Parameter	Description
<i>Subform</i>	Input, required. The subform FC to affect. Set the parameter to <i><Current Subform></i> , <i><All Children></i> , or an applicable object from the object list.

CHAPTER 30

Understanding Tab and Tab Page Controls

This chapter provides an overview of tab controls discusses how to create one.

Understanding Tab and Tab Page Controls

You can create a control that enables you to split a form into different tabbed pages. Tabs enable you to use multiple controls on a single form. You can group the control functions by placing related controls on different tab pages for a single form. You can cut and paste controls from one page to other pages.

The form has a single business view (BV). One commit for the form exists on the OK button. You can use system functions such as **Set Focus** to add additional functions 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 strategy enables you to see several pages at the same time.

Additionally, you can embed or reuse subforms on a tab page, or you can specify a subform to act as the tab page itself.

Creating Tab Controls

This section describes how to create a tab control.

Creating a Tab Control

To create a tab control:

1. On the form with which you are working, choose Tab Control from the Insert Controls tool bar.
Page Properties appears. It indicates which page of information you are on.
2. Complete the Event Rules Title.
the form appears with a tab at the top, named as you indicated.

3. Position and resize the control.

4. Choose Tab Page from the Insert Controls tool bar 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.

All tab pages appear as children of the tab control in the Application Tree View.

5. Add controls to the tab pages as if they were individual forms.

Tab Control System Functions

These system functions are specifically applicable to tab controls. They are located in the Control folder.

Disable Tab Page

Description

Use this control to render all controls on a tab page unavailable for entry both by the end user and programmatically. A disabled control is still visible.

Parameters

Parameter	Description
<i>Tab Control</i>	Input, required. The tab form control (FC) to affect.
<i>Tab Page</i>	Input, required. The tab page to disable. Set the parameter to a tab page from the list of objects.

See Also

[Chapter 30, “Understanding Tab and Tab Page Controls,” Enable Tab Page, page 250](#)

Enable Tab Page

Description

Use this system function to render all controls on a tab page available for entry both by the end user and programmatically.

Parameters

Parameter	Description
<i>Tab Control</i>	Input, required. The tab FC to affect.
<i>Tab Page</i>	Input, required. The tab page to enable. Set the parameter to a tab page from the list of objects.

See Also

[Chapter 30, “Understanding Tab and Tab Page Controls,” Disable Tab Page, page 250](#)

Hide Tab Page

Description

Use this system function to prevent the user from seeing (and therefore interacting with) the controls on a tab page. Hidden controls can be manipulated programmatically.

Parameters

Parameter	Description
<i>Tab Control</i>	Input, required. The tab FC to affect.
<i>Tab Page</i>	Input, required. The tab page to hide. Set the parameter to a tab page from the list of objects.

Additional Notes

This function can only be called on **Post Dialog is Initialized**. At all other times, use **Disable Tab Page** instead.

See Also

Chapter 30, “Understanding Tab and Tab Page Controls,” Disable Tab Page, page 250

Set Current Tab Page

Description

Use this system function to programmatically “click” a tab page, thereby bringing it to the forefront and making it active.

Parameters

Parameter	Description
<i>Tab Control</i>	Input, required. The tab FC to affect.
<i>Tab Page</i>	Input, required. The tab page to display. Set the parameter to a tab page from the list of objects.

Set Tab Page Text

Description

Use this system function to change the title of a tab page in a given instance.

Parameters

Parameter	Description
<i>Tab Control</i>	Input, required. The tab FC to affect.
<i>Tab Page</i>	Input, required. The tab page for which to change the title. Set the parameter to a tab page from the list of objects.
<i>Text</i>	Input, required. The text to show as the label for the control. Set the parameter to an alphanumeric constant (<Literal>), <Blank>, <Zero>, or an applicable object from the object list.

CHAPTER 31

Understanding Text Block Controls

This chapter describes text block controls.

Text Block Controls

You can use a text block 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 **Text Clicked** event so that you can click the text to connect to a different form. You can also use several system functions with this control. The text block control is particularly useful for Web applications.

Text Block Control Design-Time Considerations

Unlike most Form Design Aid (FDA) controls, text block properties in the Property Browser do not mirror those in the properties dialog that appears when you double-click the control. Set standard properties (such as height, title, and so forth) in the browser. Create the text strings themselves with the properties dialog. For each text string in the control, you can designate whether it is clickable, and you can control its font and color. For even more control over the appearance of the content of the control, you can include HTML tags as part of the text segments. The tags do not appear at runtime, but the engine formats the control as indicated.

Text Block Events

Only one event can fire on the text block control during runtime: **Text Clicked**.

Text Block System Functions

System functions for text block controls are located in the Text Control Functions folder.

This section discusses the system functions for text block controls.

Add Segment

Description

Use this system function to add a text segment (clickable or not) to the text block.

Parameters

Parameter	Description
<i>Text Control</i>	Input, required. The text control form control (FC) to affect.
<i>Text</i>	Input, required. The content of the text segment to add. Set the parameter to an alphanumeric constant (<Literal>), <Blank>, <Zero>, or an applicable object from the object list.
<i>Font</i>	Input, required. The font (including type, style, color, and so forth) to apply to the text segment. Set the parameter to choose a font and its properties (<Pick Font>), or to revert to the default font and property settings (<Reset Font>).
<i>Clickable</i>	Input, required. An indicator of whether the text segment is clickable. Set the parameter to <Yes> or <No>.
<i>SegmentID</i>	Output, required. The variable to which to assign the return value. Set the parameter to an applicable object from the object list.

Returns

This system function returns the ID of the segment you added to the object indicated by *SegmentID*.

Get Last Clicked Segment

Description

Use this system function to acquire the ID of the segment that was clicked which caused the current event (**Text Clicked**) to fire.

Parameters

Parameter	Description
<i>Text Control</i>	Input, required. The text control FC to affect.
<i>SegmentID</i>	Output, required. The variable to which to assign the return value. Set the parameter to an applicable object from the object list.

Returns

This system function returns the ID of the segment just clicked to the object indicated by *SegmentID*.

Get Segment Information

Description

Use this system function to acquire the textual content of a given segment, along with whether the segment is clickable.

Parameters

Parameter	Description
<i>Text Control</i>	Input, required. The text control FC to affect.
<i>Text</i>	Output, required. The variable to which to return the textual content of the text segment. Set the parameter to an applicable object from the object list.
<i>Clickable</i>	Output, required. The variable to which to assign the return value. Set the parameter to an applicable object from the object list.
<i>SegmentID</i>	Input, required. The ID of the segment to affect. Set the parameter to an applicable object from the object list.

Returns

This system function returns the text of the segment and an indicator of whether it is clickable to the objects indicated by *Text* and *Clickable*, respectively.

Remove Segment

Description

Use this system function to delete a text segment from a given text block control.

Parameters

Parameter	Description
<i>Text Control</i>	Input, required. The text control FC to affect.
<i>SegmentID</i>	Input, required. The ID of the segment to affect. Set the parameter to an applicable object from the object list.

Update Segment

Description

Use this system function to change text, font, color, and clickability of a given text segment in a text block control.

Parameters

Parameter	Description
<i>Text Control</i>	Input, required. The text control form control (FC) to affect.
<i>Text</i>	Input, required. The text that you want to display in the text segment. Set the parameter to an alphanumeric constant (<Literal>), <Blank>, <Zero>, or an applicable object from the object list.
<i>Font</i>	Input, required. The font (including type, style, color, and so forth) to apply to the text segment. Set the parameter to choose a font and its properties (<Pick Font>), or to revert to the default font and property settings (<Reset Font>).
<i>Clickable</i>	Input, required. An indicator of whether the text segment is clickable. Set the parameter to <Yes> or <No>.
<i>SegmentID</i>	Input, required. The ID of the segment to affect. Set the parameter to an applicable object from the object list.

CHAPTER 32

Understanding Text Search Controls

This chapter describes the text search control.

Text Search Controls

You can add only one text search control to any given form. The underlying business view on the form can use the results of the text search as it would the results of a regular database query. You can vary the width of the control, but not the height.

When you build the business view for the form containing the text search control, you might consider including the data dictionary items, txtscr and txtsum. Txtscr displays the score for each match returned; that is, the extent to which the result matches that for which the user originally searched. Txtsum displays the summary for the returned value; that is, the context in which the match occurred. When placed in a grid, the system populates the column automatically when the user executes a find.

Note. You must enable the business view attached to the form for text searches using Object Management Workbench (OMW), or the text search function will not work properly on the form.

This control has no unique property settings in Form Design Aid (FDA).

You cannot apply logic to the text search control, so it has no unique system functions. No events fire in response to user or runtime interaction with the control. The interconnectivity level has no impact on this control.

CHAPTER 33

Understanding Tree Controls

This chapter discusses tree controls.

Tree Controls

Tree controls display data in a hierarchical format. You can have multiple controls on a single form if you need more than one tree.

The tree itself is assembled from data that you have placed into cache. The data structure you create for this purpose must load at least three columns into cache:

- Node Value

This value is used primarily for placing the row as a node in the tree. Every node in the tree requires a value which corresponds to its position in the tree, based on parent node value. The secondary purpose of node value is for identification.

- Node Description

This value is used to label each node in the tree for the benefit of the users.

- Parent Value

This value is used to indicate which row is its parent. The value corresponds to Node Value. Runtime adds this row as a child node of its parent in the tree.

Based on this data, and given which node value is to be used as the parent, runtime can construct a tree. For example, consider this table of data:

Node Value	Node Description	Parent Value
1	Alpha Manufacturing	0
2	Branch A	1
3	Branch B	1
4	Plant x	2
5	Plant y	2
6	Plant z	3

Given that the root node value is 1, runtime could assemble a tree with this structure:

```
Alpha Manufacturing
  Branch A
    Plant x
    Plant y
  Branch B
    Plant z
```

Consequently, you must ensure that the data in the node value and parent value columns correspond such that runtime can derive the hierarchy. Additionally, many of the system functions that you can use to manipulate the tree rely on node value as an identifier. Therefore, you might want to ensure that those values are unique.

You can also use table I/O to provide node data by associating a given node with a particular table handle using the **Set Tree Node Handle** system function.

Tree controls have no control-specific properties in Form Design Aid (FDA). Additionally, no automatic runtime processing occurs for the control. Tree controls are manipulated by system functions exclusively.

Tree Control Events

These events can fire on the tree control during runtime:

- **Set Focus On Tree**

Fires when the tree control acquires focus.

- **Tree Node Selected**

Fires when the user clicks a tree node.

- **Tree Node Is Expanding**

Fires when the user or engine expands a node. This event fires only on the first expand incident for a particular node in a session.

- **Tree Node Is Collapsing**

Fires when the user or engine collapses a node.

- **Double Click on Leaf Node**

Fires when the user double-clicks a leaf node.

- **Get Custom Tree Node**

Fires when the user or engine expands a node for the first time in the session. It also fires if page-at-a-time processing is enabled and the user loads a new page.

- **Tree Node Is Deleted**

Fires when the user or engine deletes a tree node.

- **Kill Focus On Tree**

Fires when the tree control loses focus.

Tree Control System Functions

This chapter discusses the system functions unique to the tree control.

Bulk Tree Load

Description

Use this system function to load the entire contents of the tree control from cache.

Parameters

Parameter	Description
<i>Tree Control</i>	Input, required. The tree control form control (FC) to affect.
<i>Cache Name</i>	Input, required. The name of the cache from which to acquire data for the contents of the tree. Set the parameter to an applicable object from the object list.
<i>Data Structure</i>	Input, required. The name of the data structure used to organize the cache. Double-click <Get Structure Name> to select the structure to use.
<i>Node Value Column</i>	Input, required. The data to use for identification. This value is not a true ID in the traditional sense; in this system function, it is used primarily for matching with values in the <i>Parent Value Column</i> to determine parent/child relationships. If you want to use node value as a true identifier, ensure that each row will have a unique value in this column (many tree control system functions use node value as an ID). Set the parameter to an applicable object from the object list.
<i>Node Description Column</i>	Input, required. The label to display for each node. Set the parameter to an applicable object from the object list.
<i>Parent Value Column</i>	Input, required. The data to use to determine parent/child relationships. This value indicates which row is the parent of the current row. When runtime constructs the tree, it will place the current row as a child of the row with the corresponding value in <i>Node Value Column</i> . Set the parameter to an applicable object from the object list.
<i>Root Node Value</i>	Input, required. The value that indicates the root node. Ultimately, the uppermost node of the tree, the root node, has no parent. Runtime compares this value with the values in the <i>Node Column Value</i> parameter. The row with the matching value becomes the root node.

Contract Tree Node

Description

Use this system function to collapse a given node.

Parameters

Parameter	Description
<i>Tree Control</i>	Input, required. The tree control FC to affect.
<i>Node</i>	Input, required. The node to collapse. Set the parameter to: <i><Currently Selected Node></i> , <i><Last Inserted Node></i> , or <i><Currently Expanding/Collapsing Node></i> .

Delete Tree Node

Description

Use this system function to delete a node (and its children) from the tree. You can also delete just the children of a node or all of the nodes in the tree.

Parameters

Parameter	Description
<i>Tree Control</i>	Input, required. The tree control FC to affect.
<i>Node</i>	Input, required. The node to delete. Set the parameter to: <i><Currently Selected Node></i> , <i><Last Inserted Node></i> , <i><All Child Nodes Under the Current Node></i> , <i><All Nodes></i> , <i><With Specified Node Value></i> . Use the last option in conjunction with the <i>Node Value</i> parameter.
<i>Node Value</i>	Input, optional. The node to delete. This parameter is required only if you set <i>Node</i> to <i><With Specified Node Value></i> . Set the parameter to an applicable object from the object list.

Expand Tree Node

Description

Use this system function to expand a node (provided that the node is expandable).

Parameters

Parameter	Description
<i>Tree Control</i>	Input, required. The tree control FC to affect.
<i>Node</i>	Input, required. The node to expand. Set the parameter to: <i><Currently Selected Node></i> , or <i><Last Inserted Node></i> .

Get Node Information

Description

Use this system function to acquire the display text of a node and its associated node value

Parameters

Parameter	Description
<i>Tree Control</i>	Input, required. The tree control FC to affect.
<i>Node</i>	Input, required. The node about which to acquire information. Set the parameter to: <i><Currently Selected Node></i> , <i><Last Inserted Node></i> , or <i><Currently Expanding/Collapsing Node></i> .
<i>Node Text</i>	Output, required. The object to which to return the text displayed with the node. The data type of this parameter should be the same as that is displayed by the node. For example, if the node is displaying a numeric value, this parameter should be a numeric value. Set the parameter to an applicable object from the object list.
<i>Node Value</i>	Output, required. The object to which to return the value of the node. The data type of this parameter should be the same as that of the value currently stored in the node. For example, if the node currently has a numeric value, this parameter should be a numeric value. Set the parameter to an applicable object from the object list.

Additional Notes

Store the node value while inserting the node or by using the **Set Node Information** system function. A typical use is to store a key value associated with the tree node. Data of any type can be stored in the node value as long as you maintain consistency in storing and retrieving the value.

Returns

This system function returns the text associated with the node to the object indicated by *Node Text*, and the node value to the object indicated by *Node Value*.

Get Node Level

Description

Use this system function to determine the vertical level of a node in the tree. The root node is 0, so its first child is at a level of 1, its second at 2, and so on.

Parameters

Parameter	Description
<i>Tree Control</i>	Input, required. The tree control FC to affect.
<i>Node</i>	Input, required. The node about which to acquire information. Set the parameter to: <i><Currently Selected Node></i> , <i><Currently Expanding/Collapsing Node></i> , or <i><Currently Deleted Node></i> .
<i>Return To</i>	Output (numeric), required. The object to which to return the node level. Set the parameter to an applicable object from the object list.

Returns

This system function returns the level of the node to the object indicated by *Return To*.

Get Tree Node Handle

Description

Use this system function to acquire the table handle associated with a particular tree node.

Parameters

Parameter	Description
<i>Tree Control</i>	Input, required. The tree control FC to affect.
<i>Node</i>	Input, required. The node for which to acquire the handle. Set the parameter to: <i><Currently Selected Node></i> , <i><Last Inserted Node></i> , <i><Root Node></i> , <i><Currently Expanding/Collapsing Node></i> , or <i><Currently Deleted Node></i> .
<i>Index</i>	Input, required. The index of the handle. Applications can associate more than one handles to one tree node. Use this parameter to differentiate multiple handles for the same node. Set the parameter to an alphanumeric constant (<i><Literal></i>), <i><Blank></i> , <i><Zero></i> , or an applicable object from the object list.
<i>Handle</i>	Output, required. The object to which to return the handle associated with the node. Set the parameter to an applicable object from the object list.

Returns

This system function returns the handle associated with the identified tree node to the object specified by *Handle*.

Insert Tree Node

Description

Use this system function to insert a node into the tree.

Parameters

Parameter	Description
<i>Tree Control</i>	Input, required. The tree control FC to affect.
<i>Node</i>	Input, required. The position in the tree where you want to insert the node, relative to this node. Set the parameter to insert the node as a child of the current node (<Under Currently Selected Node>), to insert the node as a sibling of the current node (<After Currently Selected Node>), <After Last Inserted Node>, <Under Root Node>, <Under Currently Expanding Node>, or <Under Specified Parent Value>. Use the last option in conjunction with the <i>Parent Value</i> parameter.
<i>Node Text</i>	Input, required. The object to which to return the text displayed with the node. The data type of this parameter should be the same as that is displayed by the node. For example, if the node is displaying a numeric value, this parameter should be a numeric value. Set the parameter to an applicable object from the object list.
<i>Node Value</i>	Input, required. The object to which to return the value of the node. The data type of this parameter should be the same as that of the value currently stored in the node. For example, if the node currently has a numeric value, this parameter should be a numeric value. Set the parameter to an applicable object from the object list.
<i>Expandable?</i>	Input, required. An indicator of whether the user can expand and collapse the node. Set the parameter to <Yes> or <No>.
<i>Parent Value</i>	Input, optional. The ID of the node you want to become the parent of the node you are inserting. Use this parameter if you set <i>Node</i> to <Under Specified Parent Value>. Set the parameter to an applicable object from the object list.

Set Bitmap Scheme

Description

Use this system function to set the default bitmap schema that will be used by the tree nodes in the tree control.

Parameters

Parameter	Description
<i>Tree Control</i>	Input, required. The tree control FC to affect.
<i>Open Bitmap</i>	Input, required. The bitmap to display for a node when it has been expanded. Set the parameter to a particular bitmap (double-click <Choose Tree Bitmap> to browse for one) or the standard bitmap (<Default>).
<i>Closed Bitmap</i>	Input, required. The bitmap to display for a node when it has been collapsed. Set the parameter to a particular bitmap (double-click <Choose Tree Bitmap> to browse for one) or the standard bitmap (<Default>).
<i>Leaf Bitmap</i>	Input, required. The bitmap to display for a leaf node with no children. Set the parameter to a particular bitmap (double-click <Choose Tree Bitmap> to browse for one) or the standard bitmap (<Default>).

Additional Notes

The bitmaps must be of size 16 x 16 and should reside in the treebmps subdirectory of the resource directory (for example: d:\b7\appl_pgf\res\treebmps). If a bitmap fails to load, the application uses the corresponding standard bitmap. The ideal place for using this system function is during on event, **Dialog Is Initialized**.

Set Node Bitmap

Description

Use this system function to set a bitmap for a particular node, no matter its state.

Parameters

Parameter	Description
<i>Tree Control</i>	Input, required. The tree control FC to affect.
<i>Node</i>	Input, required. The node for which to set the bitmap. Set the parameter to: <Currently Selected Node>, <Last Inserted Node>, or <Currently Expanding/Collapsing Node>.
<i>Bitmap</i>	Input, required. The bitmap to display for the node. Set the parameter to a particular bitmap (double-click <Choose Tree Bitmap> to browse for one) or the standard bitmap (<Default>).

Additional Notes

After the bitmap for a tree node is set using this system function, the default scheme changes for that node. The system does not apply the expanded and collapsed bitmaps to the node.

The bitmaps must be of size 16 x 16 and should reside in the treebmps subdirectory of the resource directory (for example: d:\b7\appl_pgf\res\treebmps). If a bitmap fails to load, the application uses the corresponding standard bitmap. The ideal place for using this system function is during on event, **Dialog Is Initialized**.

Set Node Information

Description

Use this system function to change the node value and the text displayed for the node.

Parameters

Parameter	Description
<i>Tree Control</i>	Input, required. The tree control FC to affect.
<i>Node</i>	Input, required. The node to affect. Set the parameter to: <i><Currently Selected Node></i> , <i><Last Inserted Node></i> , or <i><Currently Expanding/Collapsing Node></i> .
<i>Node Text</i>	Input, required. The text to display for the node. Set the parameter to an applicable object from the object list.
<i>Node Value</i>	Input, required. The value to apply to the node. This value is often used for identification purposes, so you might want to ensure that it is unique. Set the parameter to an applicable object from the object list.

Additional Notes

A typical use of a node value is to store a key value associated with the tree node. Data of any type can be stored in the node value as long as you maintain consistency in storing and retrieving the value.

Set Node Text

Description

Use this system function to apply or change the text associated with a given node. This is the text that users can see.

Parameters

Parameter	Description
<i>Tree Control</i>	Input, required. The tree control FC to affect.
<i>Node Text</i>	Input, required. The text to apply to the node for display. Set the parameter to an applicable object from the object list.
<i>Node Value</i>	Input, required. The value of the node to affect. Set the parameter to <i><Root Node></i> or an applicable object from the object list.

Set Tree Node Handle

Description

Use this system function to associate a handle with a particular tree node.

Parameters

Parameter	Description
<i>Tree Control</i>	Input, required. The tree control FC to affect.
<i>Node</i>	Input, required. The node to which to associate the handle. Set the parameter to: <i><Currently Selected Node></i> , <i><Last Inserted Node></i> , <i><Root Node></i> , or <i><Currently Expanding/Collapsing Node></i> .
<i>Index</i>	Input, required. The index of the handle. Applications can associate more than one handles to one tree node. Use this parameter to differentiate multiple handles for the same node. Set the parameter to an alphanumeric constant (<i><Literal></i>), <i><Blank></i> , <i><Zero></i> , or an applicable object from the object list.
<i>Handle</i>	Input, required. The object to which to return the handle associated with the node. Set the parameter to an applicable object from the object list.

CHAPTER 34

Understanding Wizard Controls

This chapter discusses wizard controls.

Wizard Controls

The *wizard control* is the primary component of the wizard. It can reside only on the wizard form type, and only one wizard control can exist on any given wizard form. Each wizard control contains multiple *pages*; each page is comprised of one *subform* or *alias*. Each page corresponds to a task that the user must complete to finish the wizard. During runtime, the wizard displays the first page and waits for the user to complete the task. Typically, you use the pages to prompt the user to provide input. After completing the task, the user clicks the Next button, and the wizard validates page one data and displays the second page in the list (index). In a standard scenario, the user continues in this fashion until the last page is reached, at which point the user clicks Finish and the wizard re-validates and commits the data before terminating.

You can opt to display the *progress list* itself (which also enables the user to jump between tasks by clicking a task in the progress list), and you can also display a progress indicator which shows how much of the wizard is complete without listing the individual tasks. The progress list indicates which pages have been completed, which ones have errors, and which ones have warnings by displaying icons next to affected tasks. You cannot alter these icons programmatically. Additionally, the control automatically provides certain buttons on each page of the wizard. It provides a Previous button on all but the first page, a Next button on all but the last page, a Finish button on the last page, and an Exit button on all pages. You can add additional buttons such as Save and Cancel, but you cannot remove these default buttons.

You can enable users to start a wizard and then save it to finish later. If a user saves, the control does not automate saving the data for you. Instead, you must react to the **WIZARD:Save for Re-entry** event and save the data programmatically. If you enable revisits on the wizard, the interface provides a Save For Later button as well.

The subforms included in the wizard have no form- or control-based limitations; that is, you can employ the full range of subform features and system functions. Therefore, to create an effective wizard application, you must be proficient with subforms.

Each subform is parented to the wizard form which means that subforms cannot communicate with each other directly; they must share data using the wizard form. Data can be passed in one of three ways: from page to wizard form (child to parent), from wizard form to page (parent to child), or in both directions. Only data in the child page data structure can be passed to the parent wizard form. Consequently, you must plan the data structures carefully. In general, objects in the data structure will fall into one of two categories: items which the page will want to share with other pages, and items which the page will require such as input from other pages or the wizard form itself.

You can embed subforms and use an alias to a reusable subform. You can programmatically reorder the pages (therefore reordering the tasks). You can also indicate a starting page other than the first one in the list. As the user progresses through the wizard, tasks that the user has completed are referred to as *upstream tasks*. Those which the user has yet to complete are referred to as *downstream tasks*.

You are not limited to forms inside the wizard. You can use form interconnections to link to forms outside of the wizard. If you use an interconnection, the forms that appear are called *satellite pages*. Like the wizard itself, satellite pages expand to fill the entire frame; the standard EnterpriseOne navigation menu is hidden. When the user clicks OK on the satellite page, the wizard reappears displaying the page containing the object that triggered the interconnection.

Forms have different statuses including indeterminate (no state), complete, and incomplete (but continue). No state is the initial, pristine state of a page and it indicates that the page has been initialized but not yet visited. Complete indicates that the page has been visited, finished successfully, and validated. Incomplete indicates that the page has been visited but not yet finished or validated. All pages in the wizard must be at a status of complete before runtime can commit data.

Runtime validates all wizard pages when the user clicks Finish, even hidden and disabled pages. To prevent a page from being validated and saved, enable the Form Design Aid (FDA) property, Suppress Validation and Save. During runtime, use the system function, **Suppress Wizard Page Validation and Save**. As a rule of thumb, call the Suppress Wizard Page Validation and Save system function for all hidden and disabled pages in the wizard to prevent Finish button processing on them.

When a wizard application is launched, runtime sets the status of all pages to no state as part of the initialization process. When the user enters a page of indeterminate status, runtime changes its status to incomplete to reflect the fact that the user has visited the page. These are the only times that runtime sets the page status. Application logic is expected to manage page status otherwise. However, runtime will not permit the application to assign a complete status to a page until all its errors have been resolved.

All pages must be at a status of complete for the Finish button process to succeed, even hidden and disabled pages, unless validation is suppressed for those pages. If any page is not set to complete, the Finish button process will fail and no data will be saved. To set page status, use the **Set Wizard Page Status** system function.

See Also

[Chapter 29, “Understanding Subforms and Subform Aliases,” page 235](#)

[Chapter 16, “Understanding Wizard Forms,” page 103](#)

Wizard Control Design-Time Considerations

The wizard control permits the standard control property settings; likewise, the standard options are available for subforms and forms as well. It is recommended that you always enable transactions for all objects in the wizard, however. Since the runtime engine gathers and saves all data and then commits it in a single transaction if the wizard finishes with no errors, disabling transactions on any wizard object is counterproductive.

In addition to the standard control property values, the wizard control includes these wizard control-specific values as well:

- Progress Indicator

If you want to show how much of the wizard has been completed, you can choose to do so as a percentage value or as the number of tasks completed out of the whole (X of Y).

- Enable Re-entry Save

When enabled, this option displays the Save For Later button on every page. You can add logic to the button that enables users to stop using the wizard before completing it, save their work up to that point, and then return to the wizard later to finish. Because the wizard hides the EnterpriseOne navigation menu, you might consider enabling this option for lengthy wizards. Users cannot launch other EnterpriseOne applications if they cannot access the EnterpriseOne navigation menu.

Note. Enabling this option only displays the Save For Later button. You must add application logic to save data and then load it and return to the correct page when the user returns. The wizard will not commit data to the database until the entire wizard is completed.

- **Enable Progress List**

When enabled, this option displays the list of tasks (each task corresponds to a wizard page) comprising the wizard. As the user completes the tasks, his or her advancement is indicated in the progress list.

Users can jump to up- or downstream tasks by choosing a task in the list. The user cannot jump to disabled, hidden, or unvisited pages using the progress list.

- **Suppress Validation and Save**

This option is applied on individual subforms. Typically, you apply it to subforms which you have also disabled or hidden. You can use this option to prevent such pages from being processed at runtime, thereby making the wizard more efficient. Furthermore, the Finish button process requires that every validated page be at a status of complete. Depending on how you design the application, it might not be possible for disabled and hidden pages to achieve this status. Therefore, you must suppress validation and save on such pages or the wizard application will never commit data to the database.

The ability to suppress validation and save functions can be enabled and disabled during runtime with the **Suppress Wizard Page Validation and Save** system function.

Implementing Re-entry Save

When you select the Enable Re-entry Save option, the application must perform data saves and any other functions you want the application to provide if a user decides to save the wizard and return later.

This list describes a method for implementing re-entry save when each wizard page contains a separate business view:

1. Create a working table for each page.
2. On the **Wizard:Save For Reentry** event of each wizard page, save the data into the working tables.
3. When the user reenters, on the **Wizard:Subform is Initialized** event of each wizard page, load data from the working table and assign the values to form controls and variables.

This event occurs after the system applies next numbers and default values, so they will not override the user's earlier data.

This list describes a method for implementing re-entry save when each wizard page shares the same business view or has no business view:

1. Have the wizard control collect data from each page and store it in event rule (ER) variables.
The mapping links between the wizard pages and the wizard form must be bidirectional so that the wizard form can push data into each page.
2. On the **Wizard is Finished** event, have the wizard control save the data using table I/O or business functions.
3. Create one working table to hold reentry data.

4. On the **Wizard:Save For Reentry** event of each wizard page, pass the data from the child subform so the parent wizard form can save the data into the working table using table I/O or business functions.
5. When the user reenters, on the **Post Wizard is Initialized** event, have the wizard control load data from the working table and into ER variables.

Setting ER variables on this event enables you to override the next numbers applied by the system.

6. On the **Wizard:Subform is Entered** event, the system automatically populates subform interconnect (SI) variables from the parent wizard form. The page must assign the SI variables to form controls to display them.

Wizard Control Events

Wizard forms and their subforms have these wizard form-specific events (some occur on the wizard form and some on its subforms), and they typically occur roughly in this order:

- **Wizard is Initialized**

This form-level event indicates that the form and control have been created, security has been applied, system variables have been initialized, and form data structures have been loaded. It is the only event on which you should use the **Set Wizard Form Mode** system function (to change the form mode to add, update, or copy), and the only one of two events on which you should use **Set Selected Wizard Page** (to start at a page other than the first visible, enabled page in the index).

Wizard is Initialized is also where you should hide, show, disable, or enable a page, or suppress page processing for a page, as appropriate. Finally, this is also when you should rearrange the page order if necessary (**Set Wizard Page Index**).

- **Post Wizard is Initialized**

This form-level event fires immediately after **Wizard is Initialized**, indicating that the initialization process is complete and runtime is poised to enter the first page in the wizard. Any logic that should be applied to the **Wizard is Initialized** event can be applied here also. You can use the **Set Selected Wizard Page** system function on this event, but not after; runtime determines which page is the "first page" to enter based in part on whether you set a selected page.

- **WIZARD:Subform is Initialized**

This subform-level event indicates that a subform has been initialized (including the standard processing for subforms in add mode). The event fires for each subform in the wizard control, in order. This event is called only once during the life cycle of a wizard form per session. If the user worked on the wizard previously and saved (that is, if this is a reentry), this event can be used to load previous session data and then assign the data to form controls and ER variables.

- **WIZARD:Subform is Entered**

This subform-level event indicates that the page status is incomplete, that parent mapping links have been reestablished, and that the SI values have been updated. The event fires every time a user visits a page by accessing it from a previous page. The SI values should contain all current information. Assign SI values to form controls and grid rows if necessary. Set filter values and QBEs at this point if the Automatically Find On Entry option is enabled.

- **WIZARD:Post Subform is Entered**

This subform-level event indicates that any processing required by the Automatically Find on Entry option is complete. This event is called every time a user visits this page from an earlier page. This is called after the **Subform Entered** event and after the automatic data fetch if Automatically Find On Entry is enabled.

- **WIZARD:Save for Re-entry**

This subform-level event fires on every page, except those on which **Suppress Validation and Save** is enabled, when the user clicks the Save for Re-entry button. the application should save the data on the current page to a temporary working table on this event.

- **WIZARD:Validate Subform**

This subform-level event occurs before runtime conducts customized validation of the form controls and grid rows. Runtime always performs the data dictionary validations unless the Suppress Validation and Save option is enabled for the page. At this point, the application code should the page status to complete by calling system function, **Set Wizard Page Status**.

- **WIZARD:Subform is Exited**

This subform-level event fires once for each subform in the control after the transactions have been processed and runtime is preparing to close the wizard form. It is the last subform-level event to fire for each subform before the wizard closes. On this event, the application should copy form controls or grid rows to SI variables, then call the **Update Parent** subform system function to update the wizard form. You can perform this call conditionally, such as when the user clicks Next or Finish.

- **Page is Exited - Before**

This form-level event indicates that subform validation is complete. It is the last event to fire before runtime exits the current subform.

- **Page is Exited - After**

This form-level event fires immediately after any logic on **Page is Exited - Before** finishes processing.

- **Wizard is Finished - Before**

This form-level event indicates that page-level validation is complete and that runtime is about to begin passing data from the wizard to the business view.

- **Wizard is Finished - After**

This form-level event fires after the business view is loaded with data from the wizard form but before the data is committed to the database. It is on this event that you should code any data operation that needs to be included in the transaction. Do not trigger a form interconnect on this event.

- **Wizard is Exited**

This form-level event fires before form close; it is the last event to fire. This is the point at which you should add code to clean up caches and other resources used by the wizard. If you require a form interconnect (such as for a confirmation form), this event occurs outside of the transaction boundary and so is safe to use to trigger the interconnection.

Note. Even though wizard control may resemble a tab control, none of the tab page events are fired for wizard pages. Wizard pages are not tab pages.

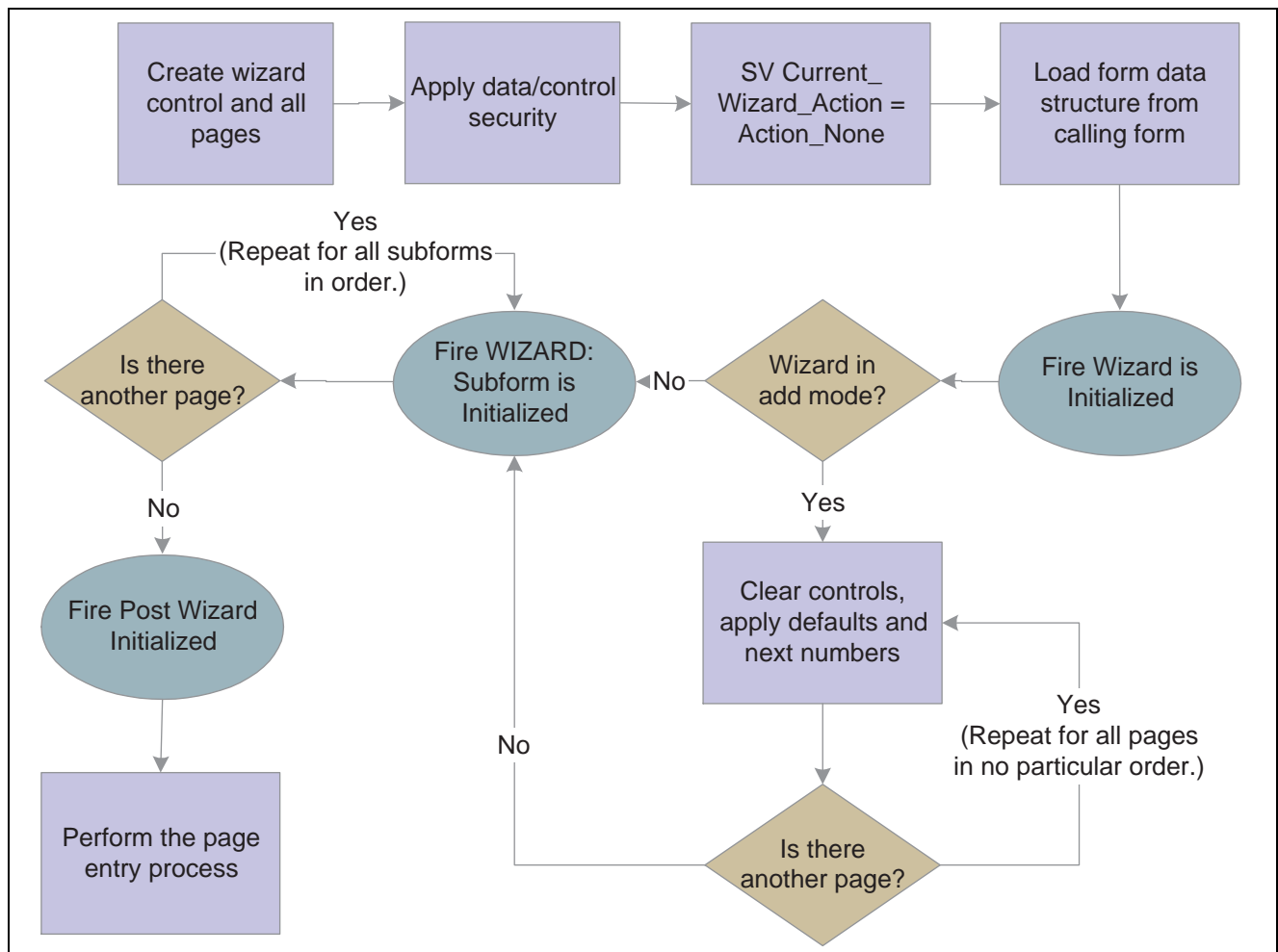
Wizard Control Runtime Processing

This section describes the runtime processing of the wizard control.

Initialization

Form initialization is the only point at which you can change the form mode (add, edit, or update), and you should do so with **Set Wizard Form Mode** on the **Wizard is Initialized** event because runtime checks the form mode immediately after firing **Wizard is Initialized**.

This flowchart illustrates how runtime initializes the wizard form:



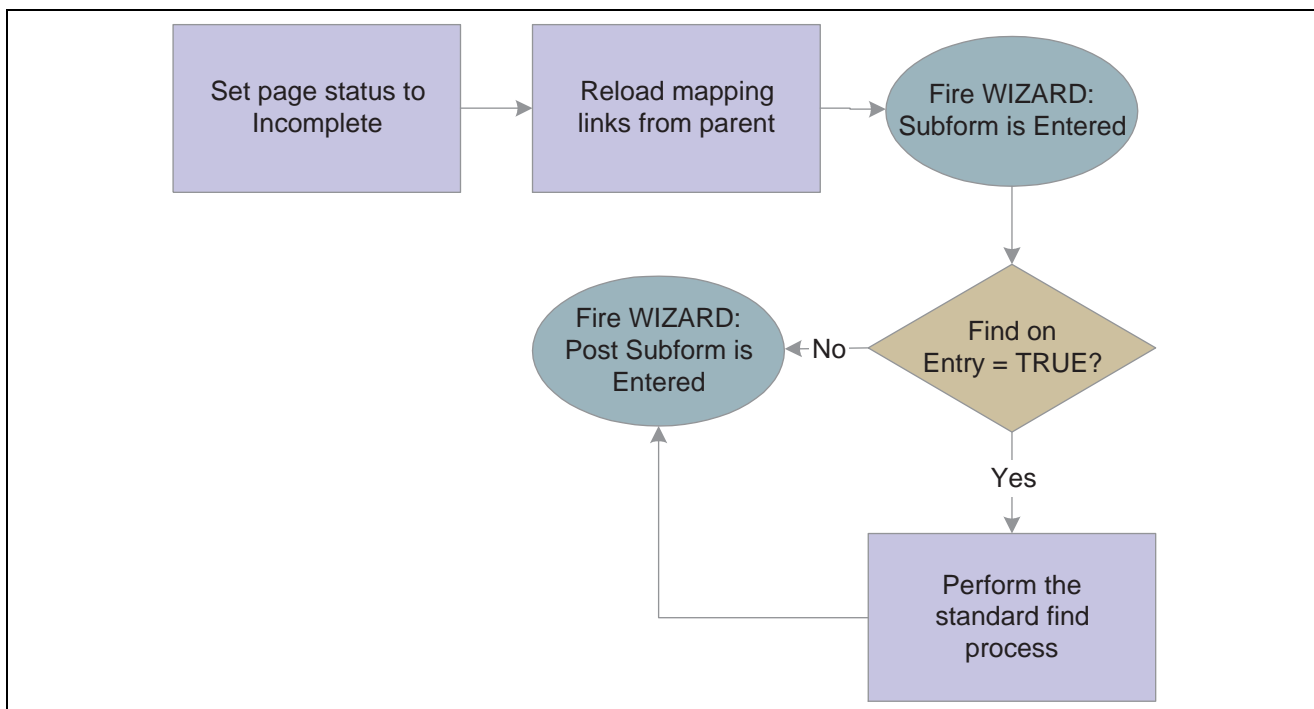
Wizard form initialization

Page Entry

After the wizard form is initialized, runtime enters the first page in the wizard control. Runtime determines the first page to be the first enabled and visible page it encounters, starting from either the first page in the index or the selected page indicated by the **Set Selected Wizard Page** system function.

When user enters a new page, the system automatically populates the SI variables from the parent form variables before **Wizard:Subform is Entered** fires. You can place logic to populate subform fields with this data on either event that fires during page entry.

This flowchart illustrates how runtime processes page entry:



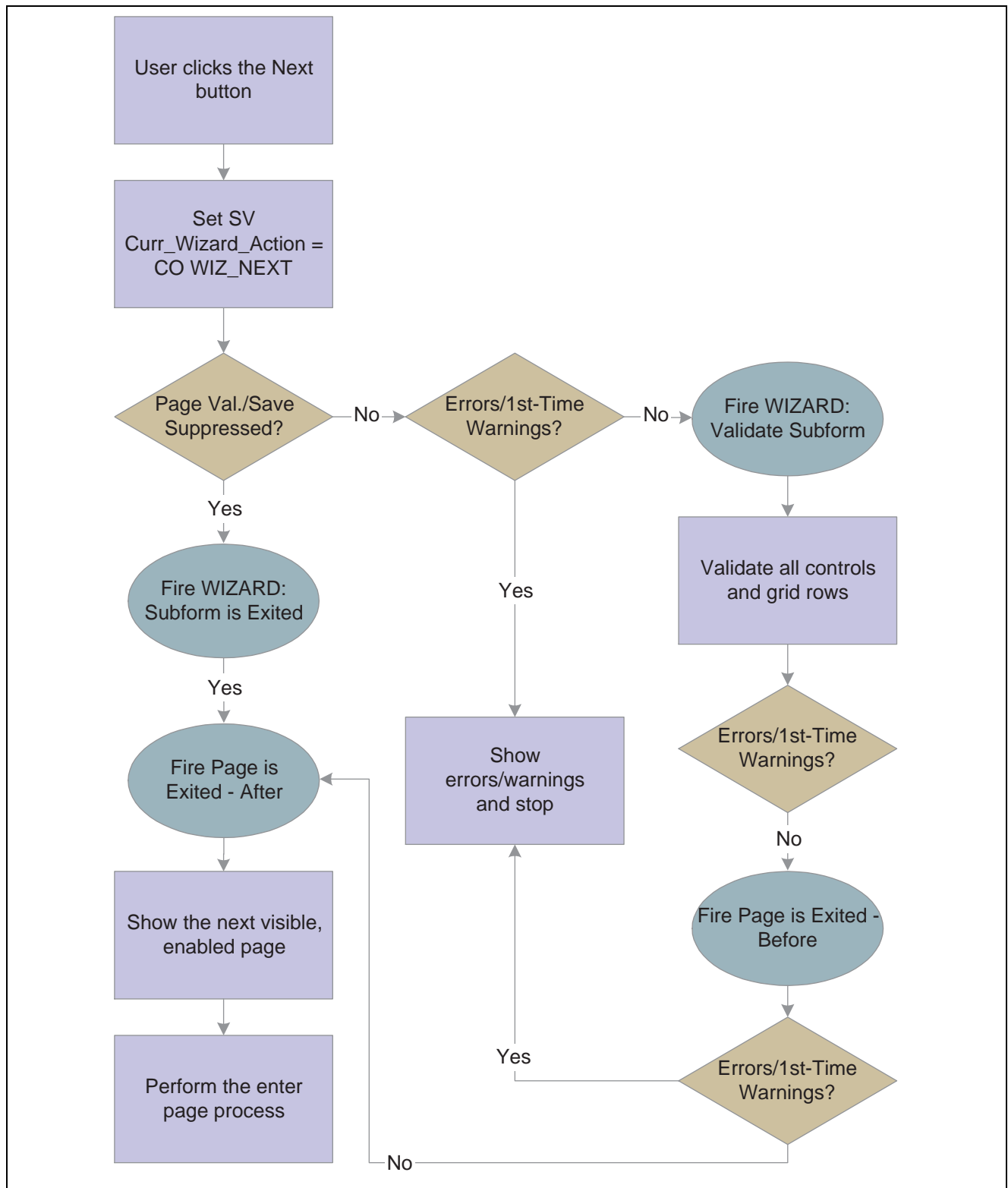
Wizard control page entry processing

Next Button Processing

The Next button causes runtime to validate the current page. If no errors occur due to validation, then runtime displays the next downstream, visible, enabled page in the index. If errors occur, runtime does not display the next page; instead, it highlights the errors so that the user can fix them before moving forward.

When user exits a page, the system does not automatically populate the SI variables to the parent form variables. Therefore, the application must assign appropriate SI variables and call the **Update Parent** subform system function in the **WIZARD:Subform is Exited** event, if necessary.

This flowchart illustrates the runtime processing that occurs on the current page when the user clicks the Next button:

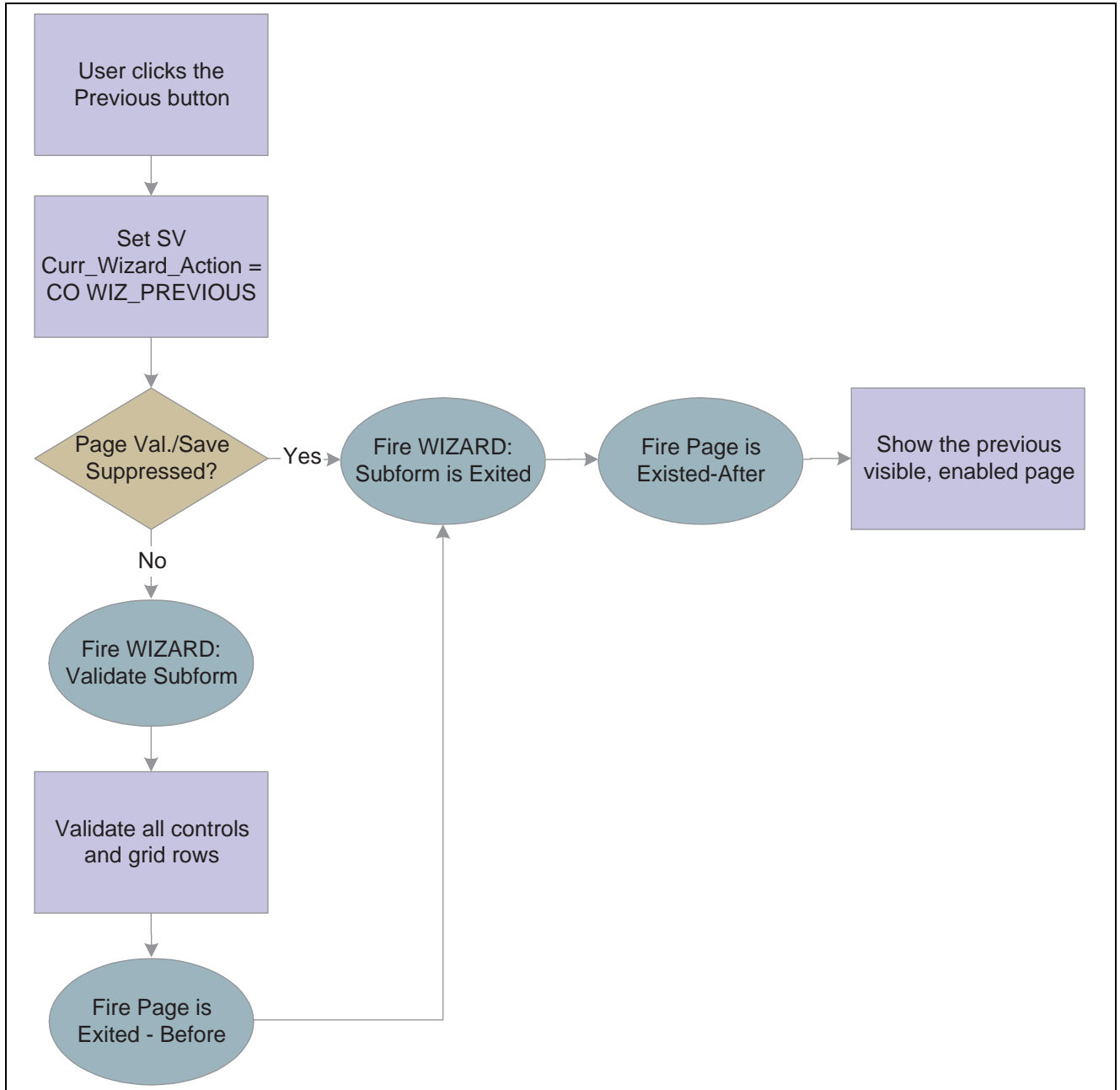


Previous Button Processing

When the user clicks the Previous button, runtime displays the previous upstream, visible, enabled page in the index, including the data entered by the user. To prevent the data from being cleared, runtime does not run the enter page process. If errors occur during validation, runtime marks the pages with errors in the progress list (if visible), but it always displays the previous page, nonetheless.

If the user wants to change previously entered data, he or she may do so, but must click Next to save the changes.

This flowchart illustrates the runtime processing that occurs when the user clicks the Previous button:



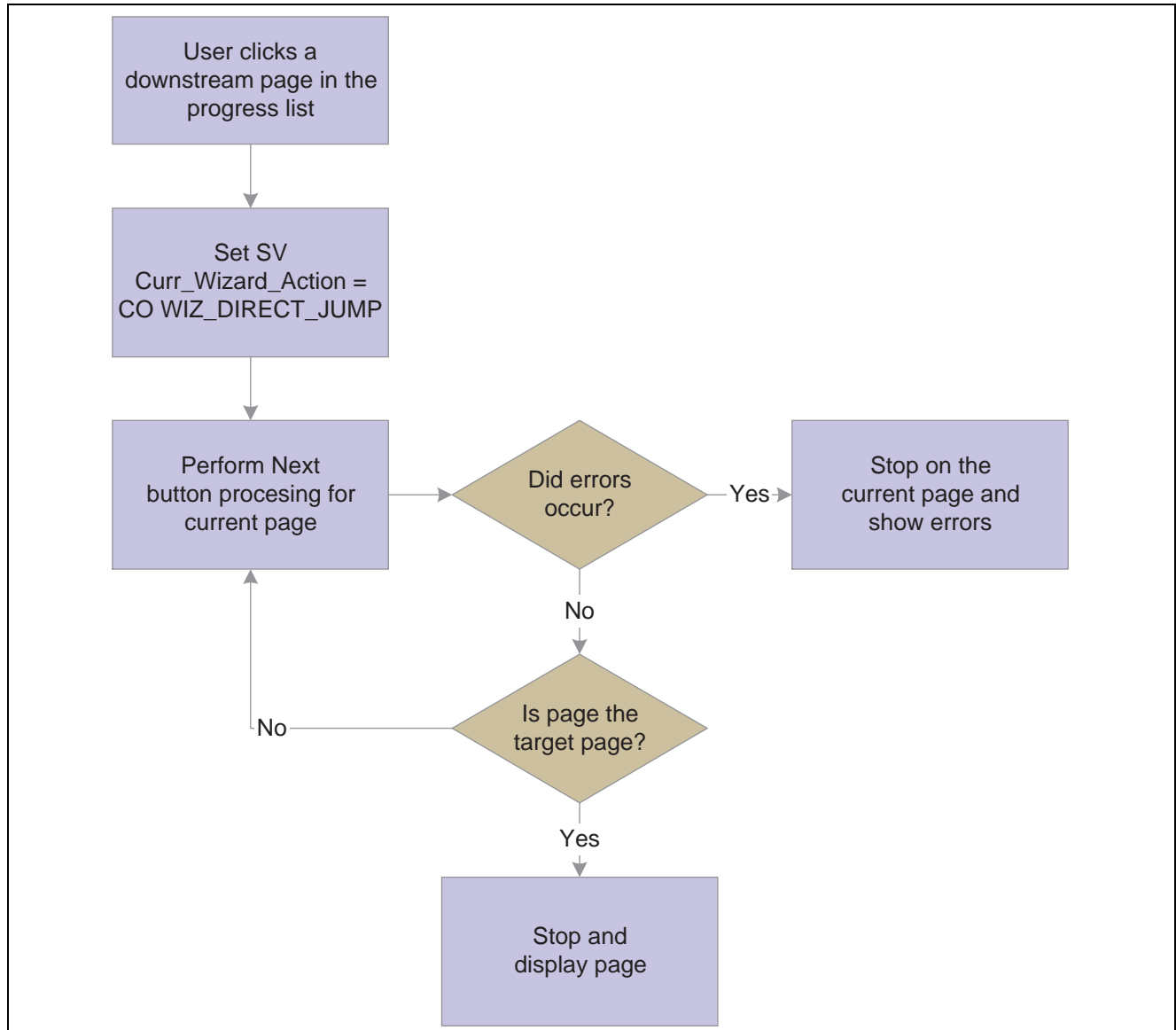
Wizard control Previous button processing

Jumping Up- and Downstream

When the progress list is enabled, users can access previously-visited pages directly, effectively skipping one or more tasks if they choose a page that is not immediately before or after the current page in the index.

When a user jumps downstream, runtime loops through each page between the start and target inclusively, applying Next button processing which includes error processing. If a page between the starting page and the target page throws an error in the course of Next button processing, runtime halts processing and displays the page on which the error occurred. If the user successfully jumps to the target page, runtime performs the page entry process.

This flowchart illustrates the runtime processing that occurs when the user tries to jump to a downstream page:

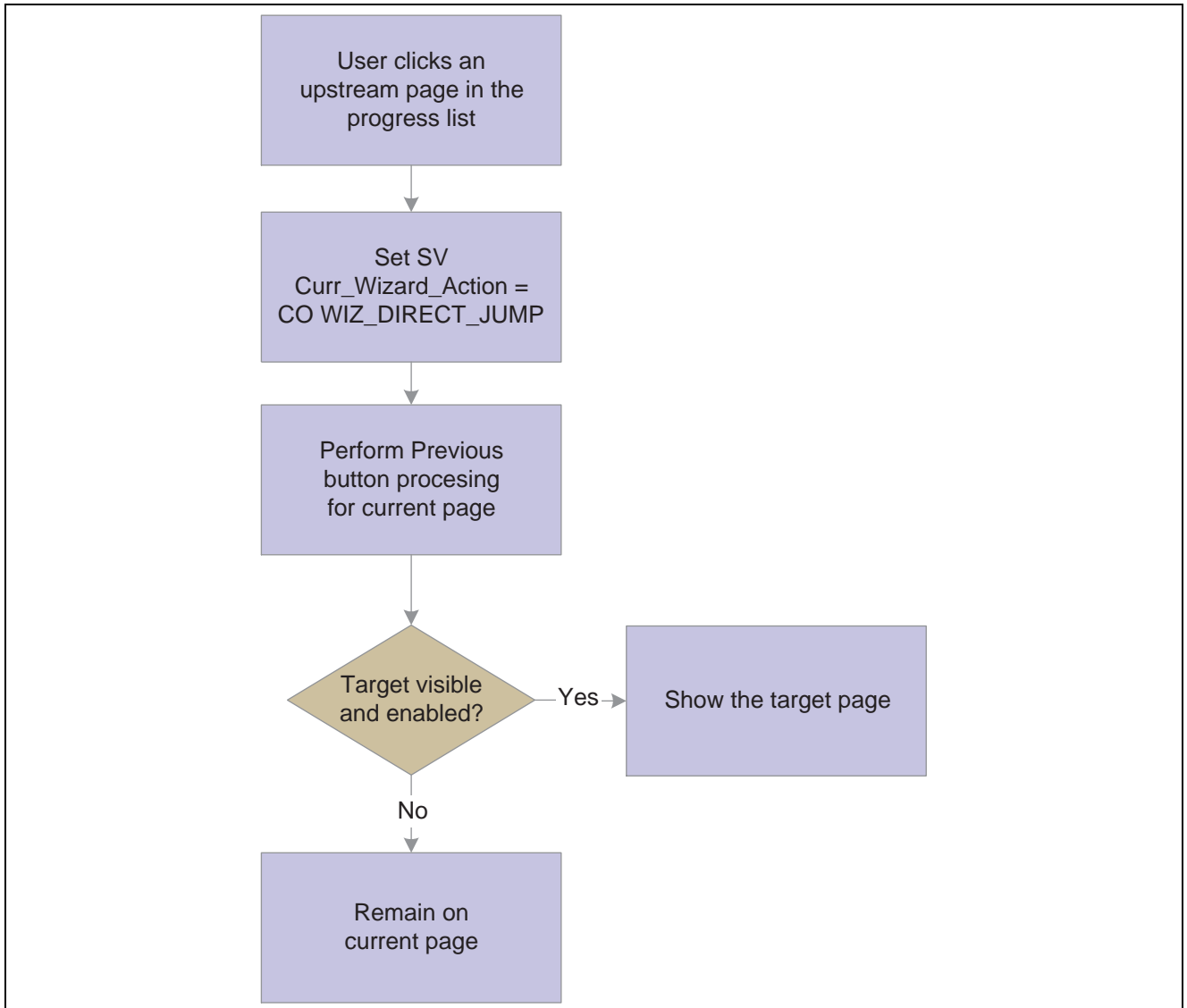


Wizard control downstream jump processing

When a user jumps upstream, runtime performs Previous button processing on the current page and then displays the target page as long as that page is visible and enabled. If the processing produces errors, runtime displays them, but it does not prevent the display of the target page. Runtime does not process the pages between the current page and the destination page because it is assumed that upstream pages should not be affected by downstream pages. Furthermore, runtime does not run the page entry process because nothing should change on the destination page.

If the user wants to change previously entered data, he or she may do so, but must click Next to save the changes.

This flowchart illustrates the runtime processing that occurs when the user tries to jump to an upstream page:

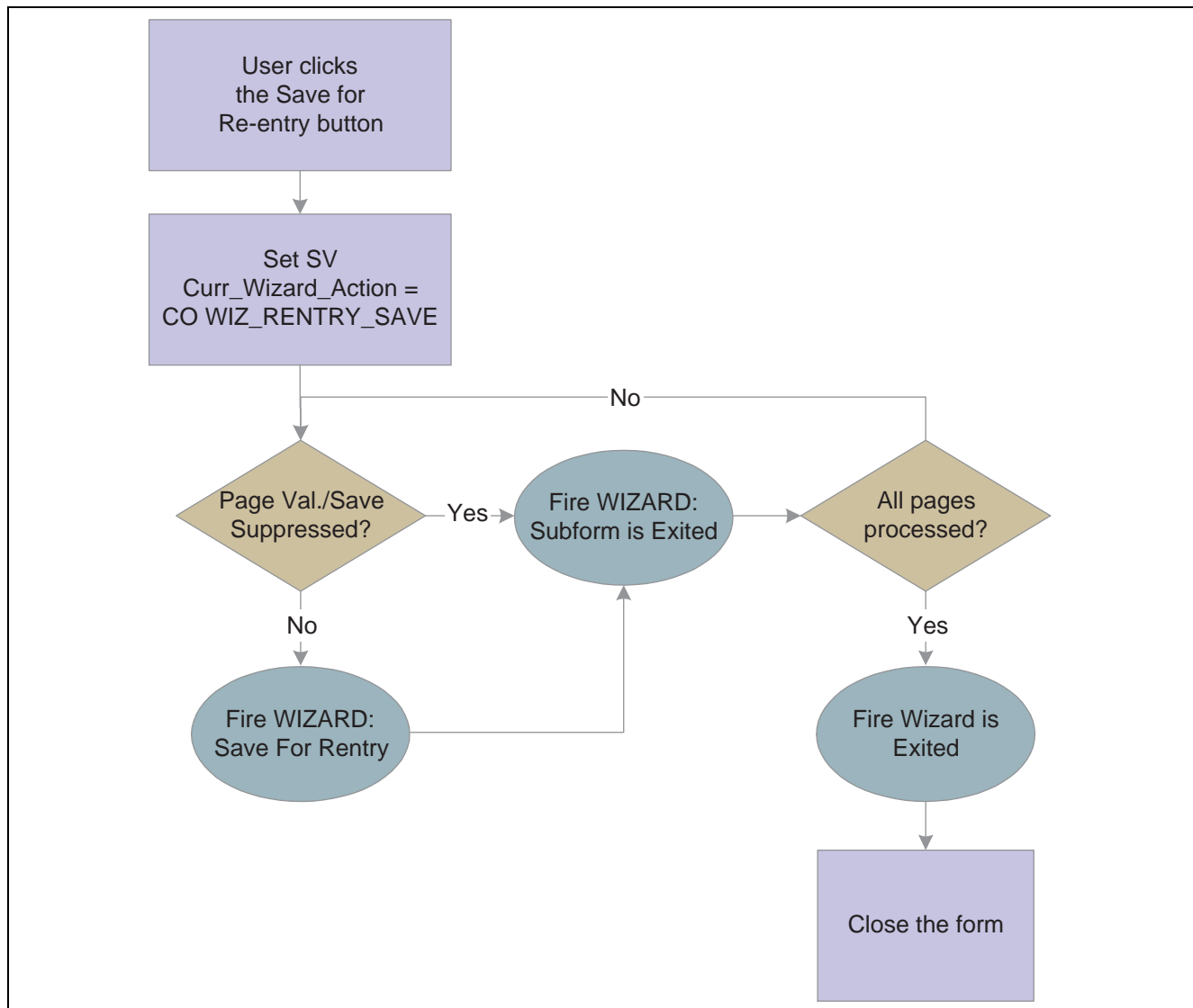


Wizard control upstream jump processing

Save for Re-entry Button Processing

If you enable it, the user can click the Save for Re-entry button to save the data to that point and close the form. You are responsible for creating logic to perform the data save (and to enable the user to return to the correct spot later).

This flowchart illustrates the runtime processing that occurs when the user clicks the Save for Re-entry button:



Wizard control Save for Re-Entry button processing

Cancel Button Processing

When the user clicks Cancel, runtime confirms that the user actually wants to quit without saving any changes. If the cancel is confirmed, runtime sets *SV Curr_Wizard_Action = CO WIZ_CANCEL*, fires **Wizard:Subform is Exited** for every page in the wizard, followed by **Wizard is Exited**. Then runtime closes the wizard form.

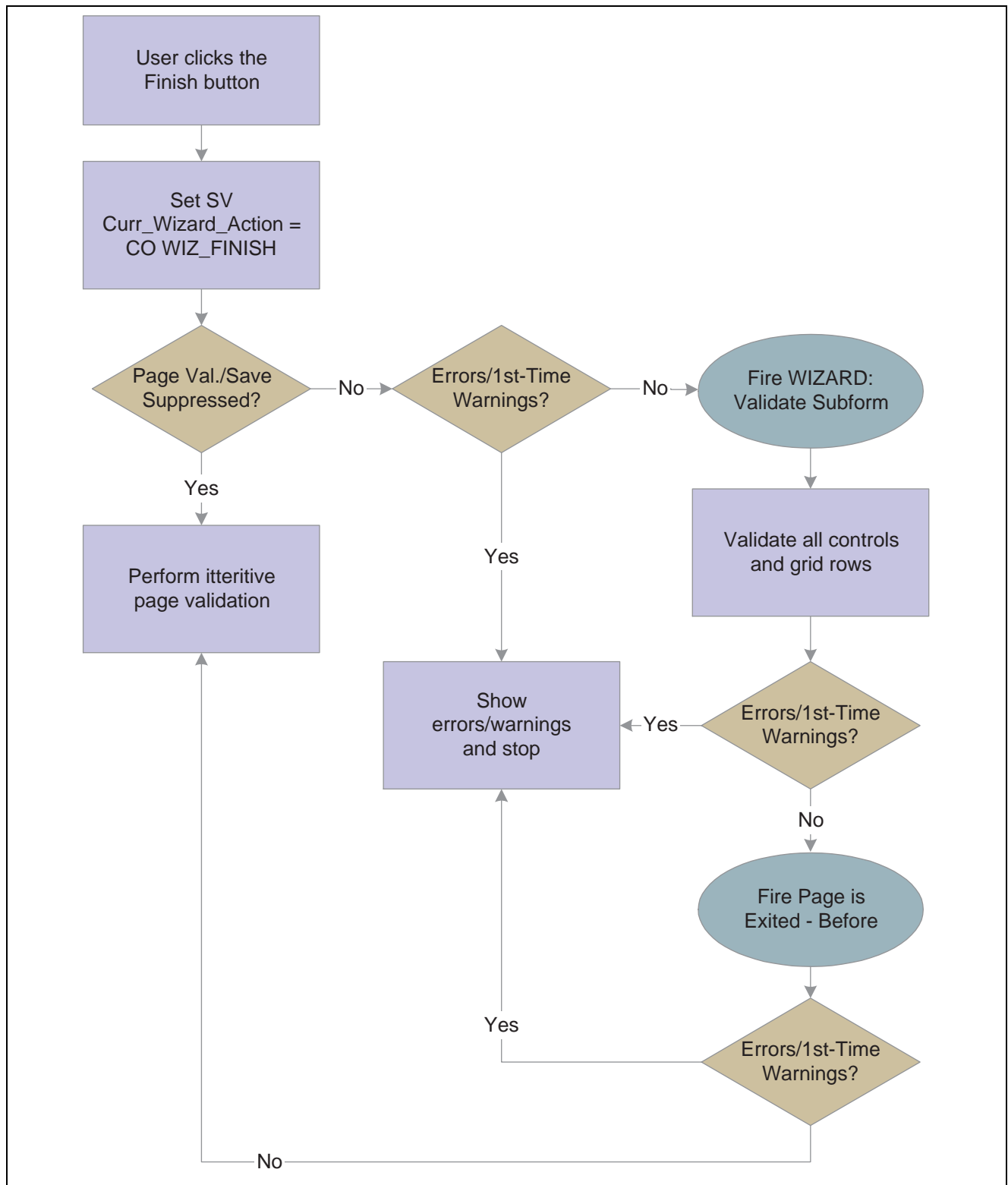
Finish Button Processing

This flowchart illustrates the runtime processing that occurs on the current page when the user clicks the Finish button:

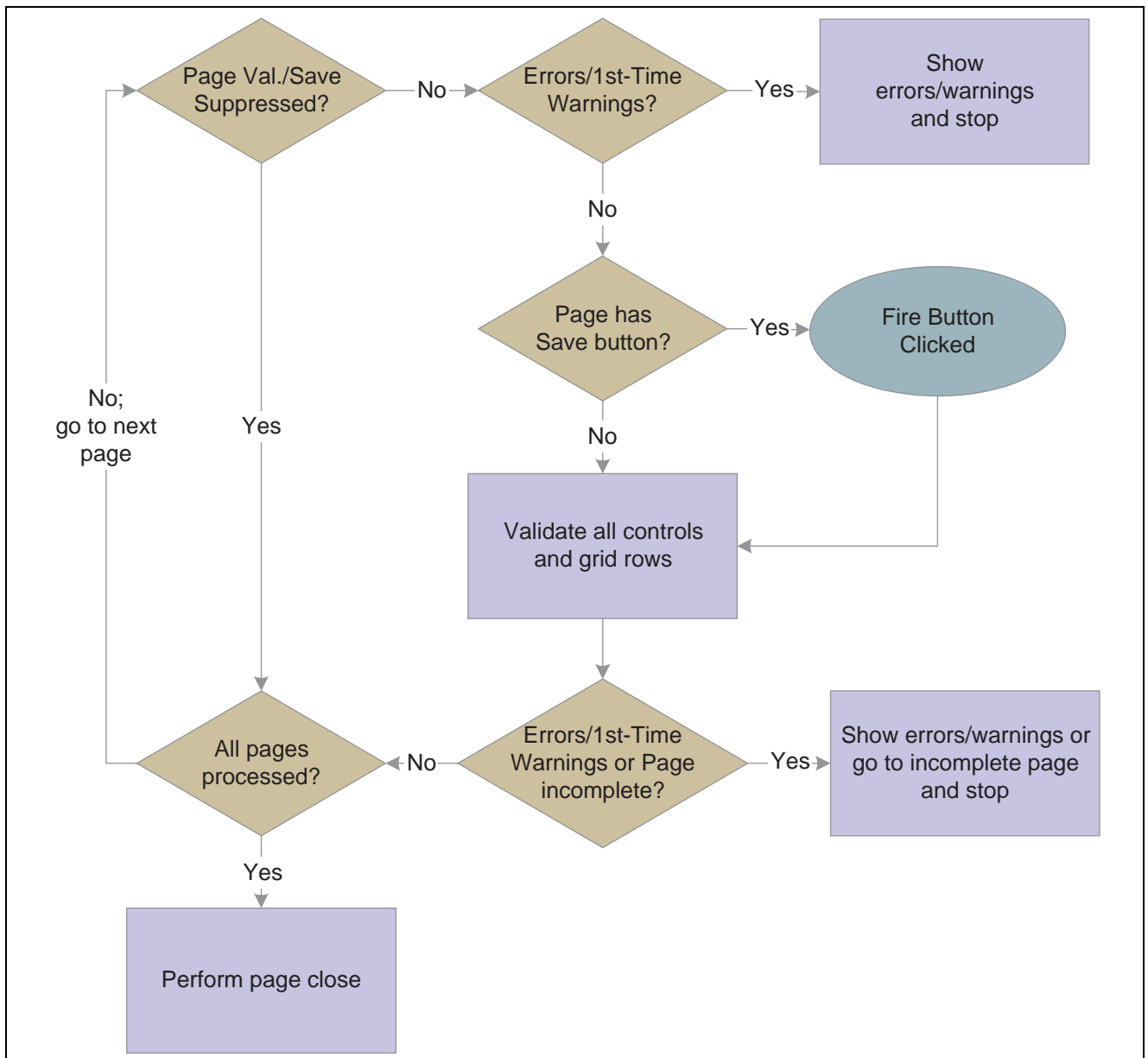
This flowchart illustrates the runtime processing that occurs for each page in the wizard after the successful validation of the last page:

Finally, runtime is ready to perform transaction processing. Runtime processes each subform in order, writing the data to the business view, but not committing to the database. If a subform has a save button that was clicked, runtime performs standard save button processing. After all of the pages have been processed, runtime commits all database changes in one transaction and then proceeds to close each subform and then the wizard form.

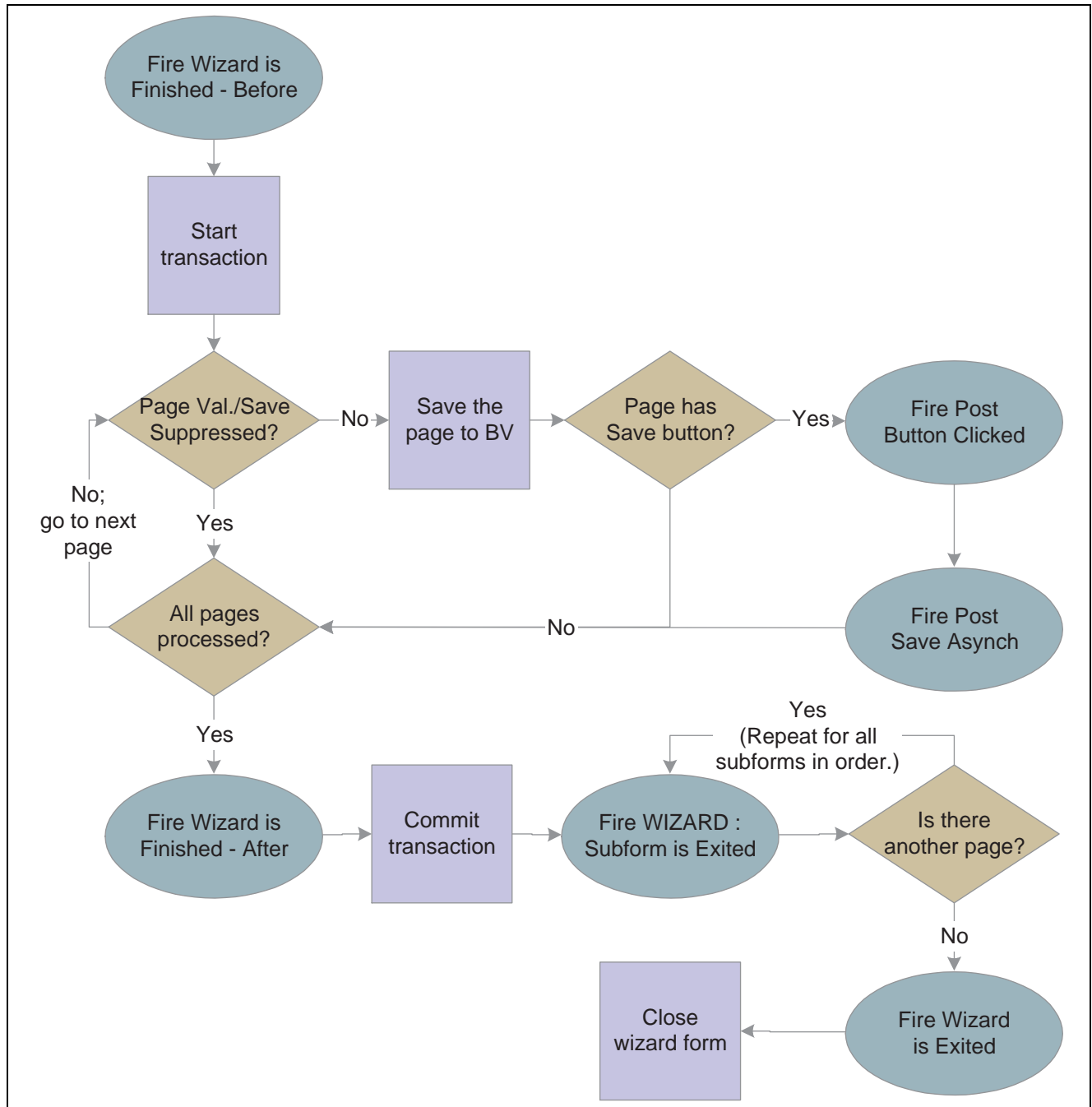
This three-part flowchart illustrates the runtime processing that occurs after the successful validation of all pages in the wizard:



Wizard form Finish button processing, part 1 of 3



Wizard control Finish button processing, part 2 of 3



Wizard control Finish button processing, part 3 of 3

Wizard Control Transaction Processing

All data is saved with a single transaction when the user clicks the Finish button. Contrast this behavior with clicking the Next button which validates data, but does not save it to the business view. Consequently, it is recommended that you limit the number of pages in a wizard for performance purposes.

The save transaction boundary is composed of and defined by this sequence of events:

1. Runtime initiates the transaction.
2. Runtime saves each page into the business view.
3. Runtime fires the **Post Button Clicked** event if it encounters a Save button on a page (the event fires for each button encountered).
4. Runtime fires the **Post Button Async** event for each Save button instance in a separate thread.
5. Runtime fires the wizard control event: **Wizard is Finished-After**.
6. Runtime performs the commit/rollback transaction.

Do *not* call form interconnects during any of the events inside the transaction boundary. Opening a new form will prolong the transaction and severely impact system performance.

For this reason, do not launch a separate “confirmation form” from any of these events. Instead, it is recommended that you use the last page of the wizard as the confirmation page. If it must be a separate form, then launch a confirmation form in the **Wizard is Exited** event. You can use the system variable *SV_Curr_Wizard_Action* and launch only the confirmation form when the user clicks the Finish button.

Wizard Control System Functions

You can use the **Disable Subform** and **Enable Subform** system functions to hide and show pages. Because the wizard form is the parent of all subforms, only the wizard form can disable, enable, hide, or show a given page. This section lists the system functions that are specific to the wizard form and the wizard control.

Get Current Wizard Page ID

Description

Use this system function to get the ID of the current page in the wizard. (The control ID of a subform can be found in the properties for the subform in FDA.)

Parameters

Parameter	Description
<i>Wizard Control</i>	The wizard form control (FC) to affect.
<i>Wizard Page ID</i>	Input (integer), required. The object to which to assign the return value that indicates the ID of the current page. Set the parameter to an applicable object from the object list

Returns

The system function returns the ID of the current wizard page to the object indicated by *Wizard Page ID*.

Get Wizard Page Index

Description

Use this system function to retrieve the index of a selected wizard page in relationship to the other wizard pages. The list of wizard pages starts with an index of one.

Parameters

Parameter	Description
<i>Wizard Control</i>	The wizard FC to affect.
<i>Subform</i>	Input, required. The page for which you want the index value. Set the parameter to an applicable object from the object list.
<i>Index</i>	Input (integer), required. The object to which to assign the return value that indicates the point in the list where the subform resides. This parameter is 1-based. Set the parameter to an applicable object from the object list.

Returns

The system function returns the index value of the subform to the object defined by *Index*.

Set Selected Wizard Page

Description

Use this system function to make one of the wizard pages active; that is, “selected.”

Parameters

Parameter	Description
<i>Wizard Control</i>	The wizard FC to affect.
<i>Subform</i>	Input, required. The page you want to make active. Set the parameter to an applicable object from the object list.

Additional Notes

This function is specifically designed to enable the user to change the initial page that will be displayed on a wizard control. If the Enable Re-entry Save option is selected, you can use this system function to display the page where the user left off when he or she restarts the wizard, for example.

Set Selected Wizard Page should not be called outside of the initialization of a wizard control (**Wizard is Initialized** or **Wizard Post-Initialized** events). Calling this function outside of the initialization of a wizard control might cause the wizard to malfunction and so is unsupported.

If a page is visible and enabled, this function enables you to skip it only if it has a status of complete or incomplete.

Set Wizard Form Mode

Description

Use this system function to change the runtime mode of the wizard form to update, add, or copy. Use this function in the **Wizard is Initialized** event only.

Parameters

Parameter	Description
<i>Mode</i>	Input, required. The runtime mode to which to set the wizard form. Set the parameter to <Update Mode> (1), <Add Mode> (2), or <Copy Mode> (3).

Set Wizard Page Index

Description

Use this system function to change the index of a selected wizard page, effectively "reordering" the pages. The list of wizard pages starts with an index of one.

Parameters

Parameter	Description
<i>Wizard Control</i>	The wizard FC to affect.
<i>Subform</i>	Input, required. The page for which you want to change the index value. Set the parameter to an applicable object from the object list.
<i>Index</i>	Input (integer), required. The index value to assign to the page. This parameter is 1-based. Set the parameter to an applicable object from the object list.

Set Wizard Page Status

Description

Use this system function to set the status of a page in the wizard. The default status of a wizard page is to have no status.

Parameters

Parameter	Description
<i>Subform</i>	Input, required. The subform FC to affect.
<i>Status</i>	Input (integer), required. The status to which to set the page. Set the parameter to <Complete> (1), <Incomplete> (2), or applicable object from the object list.

Additional Notes

The default status of a wizard page is to have no status. An enabled and visible page with no status will not permit the user to move the next page. Changing the status to either complete or incomplete will permit the user to move to the next page.

All enabled and visible pages must be set to a complete status in order for completion processing to take place and permit the user to complete the wizard.

Suppress Wizard Page Validation and Save

Description

Use this system function to prevent runtime validation and save functions for a specified wizard page. Typically, you suppress validation and save on invisible or disabled wizard pages.

Parameters

Parameter	Description
<i>Subform</i>	Input, required. The subform FC to affect.
<i>Suppress</i>	Input (integer), required. A flag indicating whether to validate and save the page. Set the parameter to <Yes> (1) or <No> (2).

APPENDIX A

System Functions in Form Design Aid

This appendix discusses most of the system functions that you can access through Form Design Aid (FDA).

System Functions

You access and apply most system functions through Form Design Aid (FDA). System functions are visually grouped into folders based on function or control type. This appendix mirrors that system function folder organization; however, it does not describe the system functions that are detailed in a chapter devoted to a particular control type. Furthermore, some system functions are described in other locations entirely.

Note. Do not use the system functions in the Deprecated or Obsolete folders.

See Also

EnterpriseOne 8.93 APIs

Control

These system functions are applied on the control level and work for most control types. System functions that apply to a specific control type do not appear here; they are described in the chapter dedicated to that control type.

See Also

[Chapter 20, “Understanding Edit Controls,” Edit Control System Functions, page 138](#)

[Chapter 30, “Understanding Tab and Tab Page Controls,” Tab Control System Functions, page 250](#)

Clear Control Error

Description

Use this system function to clear the errors that have been set on a control.

Parameters

Parameter	Description
<i>Control</i>	Input, required. The form control (FC) to affect.

See Also

[Appendix A, “System Functions in Form Design Aid,” Set Control Error, page 291](#)

Disable Control

Description

Use this system function to render a control unavailable for entry both by the user and programmatically. A disabled control is still visible.

Parameters

Parameter	Description
<i>Control</i>	Input, required. The FC to affect.

See Also

[Appendix A, “System Functions in Form Design Aid,” Enable Control, page 290](#)

Enable Control

Description

Use this system function to render a control available for entry both to the end user and programmatically.

Parameters

Parameter	Description
<i>Control</i>	Input, required. The FC to affect.

See Also

[Appendix A, “System Functions in Form Design Aid,” Disable Control, page 290](#)

Go to Url

Description

Use this system function to insert a functional URL into the application.

Parameters

Parameter	Description
<i>URL</i>	Input, required. The fully-qualified URL to which to link. Set the parameter to an alphanumeric constant (<Literal>), <Blank>, <Zero>, or an applicable object from the object list.

Hide Control

Description

Use this system function to prevent the user from seeing (and therefore interacting with) a control. Hidden controls can be manipulated programmatically.

Parameters

Parameter	Description
<i>Control</i>	Input, required. The FC to affect.

See Also

[Appendix A, “System Functions in Form Design Aid,” Disable Control, page 290](#)

[Appendix A, “System Functions in Form Design Aid,” Show Control, page 293](#)

Set Control Error

Description

Use this system function to set an error on a control.

Parameters

Parameter	Description
<i>Control</i>	Input, required. The FC to affect.
<i>Error Code</i>	Input, required. The error to set on the control. Set the parameter to an alphanumeric constant (<Literal>), <Blank>, <Zero>, or an applicable object from the object list.

See Also

[Appendix A, “System Functions in Form Design Aid,” Clear Control Error, page 289](#)

Set Control Text

Description

Use this system function to change the label of a control in a given instance.

Parameters

Parameter	Description
<i>Control</i>	Input, required. The FC to affect.
<i>Text</i>	Input, required. The text to show as the label for the control. Set the parameter to an alphanumeric constant (<Literal>), <Blank>, <Zero>, or an applicable object from the object list.

See Also

[Appendix A, “System Functions in Form Design Aid,” Clear Control Error, page 289](#)

Set Data Dictionary Item

Description

Use this system function to choose the data dictionary (DD) item to use for a control.

Parameters

Parameter	Description
<i>Control</i>	Input, required. The FC to affect.
<i>DD Alias</i>	Input, required. The DD item to which to apply to the control. Set the parameter to choose a DD item from the Data Dictionary dialog (<Pick DD Item>) or an applicable object from the object list.
<i>System Code</i>	Input, required. The system code to use when determining whether to apply system code-based jargon to the DD item text fields. Set the parameter to match the current system code (<Default>) or an applicable object from the object list.

Additional Notes

The control must be a DD item, not database item. When changing them, the DD items to be switched must be of the same type except for one case: You can change a string to a character but not vice versa. If you make a change between string types, the maximum size of the new string item will be the smaller size of the two switched items.

Set Data Dictionary Overrides

Description

Use this system function to apply DD overrides of any type to a control based on any kind of data item (included BV items).

Parameters

Parameter	Description
<i>Control</i>	Input, required. The FC to affect.
<i>Overrides</i>	Input, required. The override to apply. Set the value to <Data Dictionary Overrides>, or double-click <Data Dictionary Overrides> to choose specific overrides to set.

See Also

Appendix A, “System Functions in Form Design Aid,” Set Control Text, page 291

Set Statusbar Text

Description

Use this system function to set the text in the status bar for a given control, identified by its associated DD item.

Parameters

Parameter	Description
<i>DD Alias</i>	Input, required. The DD item to affect. Set the parameter to an alphanumeric constant (<Literal>), <Blank>, or <Zero>.

Additional Notes

To clear the status bar text, use this system function with a *DD Alias* parameter of <Blank> or <Zero>.

Show Control

Description

Use this system function to enable the user to see (and therefore interact with) a hidden control.

Parameters

Parameter	Description
<i>Control</i>	Input, required. The FC to affect.

See Also

Appendix A, “System Functions in Form Design Aid,” Set Control Text, page 291

Was Value Entered

Description

Use this system function to determine if a form control has been changed. A return value of zero indicates no change, and a return value of one indicates a change.

Parameters

Parameter	Description
<i>Control</i>	Input, required. The FC to affect.
<i>Return To</i>	Input, required. The object to which to return the Boolean. Set the parameter to an applicable object from the object list.

Additional Notes

Two flags track the changes: the form flag and the control flag. When a control is changed, both flags are set to one. When **Was Value Entered** is called and a specific control is selected for the control parameter, the current value of the control flag is returned. The control value is then set to zero. The form flag remains the same. When <All Controls> is selected for the control parameter, the current value of the form flag is returned. The form flag value is then set to zero, but the control flag remains the same.

Returns

This system function can return one of these values:

Value	Description
0	The control did not change.
1	The control changed.

General

These system functions affect applications in a variety of ways. They reside in the General folder in FDA.

Cancel User Transaction

Description

Use this system function to cancel any transaction committed by the user. The current transaction is assumed to be the one to cancel, so no input parameters are necessary.

Continue Custom Data Fetch

Description

Use this system function to set up a custom fetch routine for page-at-a-time processing. Called on the **Get Custom Grid Row** event, it causes runtime to add lines to the grid (one at a time) until the page is full. No input parameters are necessary.

Copy Currency Information

Description

Use this system function to copy currency information (type and number of decimal places) between controls.

Parameters

Parameter	Description
<i>To Control</i>	Input (math numeric), required. The control to which to copy the currency data. Set the parameter to an applicable object from the object list.
<i>From Control</i>	Input (math numeric), required. The control from which to copy the currency data. Set the parameter to an applicable object from the object list.

Launch Batch Application

Description

Use this system function to establish a report interconnection and launch a batch application (report).

Parameters

Parameter	Description
<i>Report Name</i>	Input (string), required. The report to launch. Set the parameter to an alphanumeric constant (<Literal>), <Null>, or an applicable object from the object list.
<i>Version Name</i>	Input (string), required. The version of the report to launch. Set the parameter to an alphanumeric constant (<Literal>), <Null>, or an applicable object from the object list.
<i>Print Preview?</i>	Input (character: Y/N), required. An indicator of whether to display a print preview of the report. Set the parameter to an alphanumeric constant (<Literal>), <Null>, or an applicable object from the object list.
<i>Data Selection?</i>	Input (character: Y/N), required. An indicator of whether to provide the user the opportunity to override the default data selection for the report. Set the parameter to an alphanumeric constant (<Literal>), <Null>, or an applicable object from the object list.
<i>Data Sequencing?</i>	Input (character: Y/N), required. An indicator of whether to provide the user the opportunity to override the default data sequencing for the report. Set the parameter to an alphanumeric constant (<Literal>), <Null>, or an applicable object from the object list.
<i>Push Specs Only?</i>	Input (character: Y/N), required. An indicator of whether to submit the specifications from the version only. Set the parameter to an alphanumeric constant (<Literal>), <Null>, or an applicable object from the object list.
<i>PO Template Name</i>	Input (string), required. The processing option data structure to use. Set the parameter to an alphanumeric constant (<Literal>), <Null>, or an applicable object from the object list.
<i>Prompt for Values?</i>	Input (character: Y/N), required. An indicator of whether to prompt the user for processing option values. Set the parameter to an alphanumeric constant (<Literal>), <Null>, or an applicable object from the object list.
<i>Date Last Executed</i>	Input (JDEDATE), required. The source for the date indicating when the report was run. Set the parameter to an alphanumeric constant (<Literal>), <Null>, or an applicable object from the object list.
<i>Data Source Override</i>	Input (string), required. The source for the data upon which to base the report. Set the parameter to an alphanumeric constant (<Literal>), <Null>, or an applicable object from the object list.
<i>JDE Log?</i>	Input (character: Y/N), required. An indicator of whether to generate a JDE.log file. Set the parameter to an alphanumeric constant (<Literal>), <Null>, or an applicable object from the object list.
<i>JDEDebug Log?</i>	Input (character: Y/N), required. An indicator of whether to generate a Jdedebug.log file. Set the parameter to an alphanumeric constant (<Literal>), <Null>, or an applicable object from the object list.
<i>UBE Logging Level?</i>	Input (integer), required. The level of detail to apply when compiling Jdedebug.log. Set the parameter to an alphanumeric constant (<Literal>), <Null>, or an applicable object from the object list.

Parameter	Description
<i>Jargon Code</i>	Input (string), required. The system code corresponding to the jargon values you want to apply. Set the parameter to an alphanumeric constant (<Literal>), <Null>, or an applicable object from the object list.
<i>Cover Page?</i>	Input (character: Y/N), required. An indicator of whether to print a cover page for the report. Set the parameter to an alphanumeric constant (<Literal>), <Null>, or an applicable object from the object list.
<i>Job Queue Name</i>	Input (string), required. The name to use to identify the job after it has been submitted to the job queue. Set the parameter to an alphanumeric constant (<Literal>), <Null>, or an applicable object from the object list.
<i>TC Prompting?</i>	Input (character: Y/N), required. An indicator of whether to prompt the user for table conversion option values. Set the parameter to an alphanumeric constant (<Literal>), <Null>, or an applicable object from the object list.
<i>Process Type</i>	Input (character), required. The type of process to run. Set the parameter to an alphanumeric constant (<Literal>), <Null>, or an applicable object from the object list.

Launch Processing Options Dialog

Description

When launching a batch application (report), use this system function to launch the processing options dialog for it.

Parameters

Parameter	Description
<i>Object Name</i>	Input, required. The report being launched. Set the parameter to an alphanumeric constant (<Literal>), <Null>, or an applicable object from the object list.
<i>Version Name</i>	Input, required. The version of the report being launched. Set the parameter to an alphanumeric constant (<Literal>), <Null>, or an applicable object from the object list.

Press Button

Description

Use this system function to “click” a button in the current application programmatically.

Parameters

Parameter	Description
<i>Control</i>	Input, required. The button to “click.” Set the parameter to an applicable object from the object list.

Additional Notes

This is applicable for enabled (but not necessarily visible) form controls and HCs. The event, Button Clicked, fires for the control that is pushed. Note that this system function moves the focus to the control that was activated.

Run Executable

Description

Use this system function to launch an executable program outside of EnterpriseOne.

Parameters

Parameter	Description
<i>EXE Directory</i>	Input (string), required. The location of the executable. Set the parameter to an alphanumeric constant (<Literal>), <Blank>, <Zero>, or an applicable object from the object list.
<i>EXE Name</i>	Input (string), required. The name of the executable. Set the parameter to an alphanumeric constant (<Literal>), <Blank>, <Zero>, or an applicable object from the object list.
<i>Parameters #1</i>	Input (string), required. The first input parameter to pass to the executable. Set the parameter to an alphanumeric constant (<Literal>), <Blank>, <Zero>, or an applicable object from the object list.
<i>Parameters #2</i>	Input (string), required. The second input parameter to pass to the executable. Set the parameter to an alphanumeric constant (<Literal>), <Blank>, <Zero>, or an applicable object from the object list.
<i>Parameters #3</i>	Input (string), required. The third input parameter to pass to the executable. Set the parameter to an alphanumeric constant (<Literal>), <Blank>, <Zero>, or an applicable object from the object list.
<i>Working Directory</i>	Input (string), required. The location where the executable should place its temporary files. Set the parameter to an alphanumeric constant (<Literal>), <Blank>, <Zero>, or an applicable object from the object list.

Set Control Focus

Description

Use this system function to place focus on a specific control.

Parameters

Parameter	Description
<i>Control</i>	Input, required. The control to receive the focus. Set the parameter to an applicable object from the object list.

Additional Notes

This is applicable for form controls and HCs. The event, **Control Is Entered**, fires on the control that receives focus.

Set Form Title

Description

Use this system function to change the title of the current form.

Parameters

Parameter	Description
<i>New Title</i>	Input, required. The title to display on the form. Set the parameter to an alphanumeric constant (<Literal>), <Blank>, <Zero>, or an applicable object from the object list.

Set Time Zone On Form

Description

Use this system function to set the time zone for the current form.

Parameters

Parameter	Description
<i>Time Zone</i>	Input, required. The time zone to use for the current form. Set the parameter to an applicable object from the object list.

Stop Processing

Description

Use this system function to stop runtime from processing the ER on the current event. No parameters are necessary.

Suppress Add

Description

Use this system function to prevent the runtime engine from executing a database add. No parameters are required. Call this system function on the **Add Rec to DB - Before** or **Add Grid Rec to DB - Before** event rule, as appropriate.

Suppress Default Visual Assist Form

Description

Use this system function to prevent the default form from appearing when the user clicks a visual assist. No parameters are required. Call this system function on the **Visual Assist Button Clicked** event, followed by a call to the form that you want to open instead.

Suppress Delete

Description

Use this system function to prevent the runtime engine from executing a database delete. No parameters are required. Call this system function on the **Delete Rec to DB - Before** or **Delete Grid Rec to DB - Before** event rule, as appropriate.

Suppress Find

Description

Use this system function to prevent the runtime engine from executing a database fetch. No parameters are required.

Suppress Update

Description

Use this system function to prevent the runtime engine from executing a database update. No parameters are required. Call this system function on the **Update Rec to DB - Before** or **Update Grid Rec to DB - Before** event rule, as appropriate.

Time Between

Description

Use this system function to calculate the amount of time that passed between two dates.

Parameters

Parameter	Description
<i>Start UTC</i>	Input (JDEUTime), required. The first date in Universal Time Code (UTC). Set the parameter to an alphanumeric constant (<Literal>) or an applicable object from the object list.
<i>End UTC</i>	Input (JDEUTime), required. The second date in Universal Time Code (UTC). Set the parameter to an alphanumeric constant (<Literal>) or an applicable object from the object list.
<i>Days</i>	Input, required. The object to which to assign the number of days that have passed. Set the parameter to an applicable object from the object list.
<i>Hours</i>	Input, required. The object to which to assign the number of hours that have passed. Set the parameter to an applicable object from the object list.
<i>Minutes</i>	Input, required. The object to which to assign the number of minutes that have passed. Set the parameter to an applicable object from the object list.
<i>Seconds</i>	Input, required. The object to which to assign the number of seconds that have passed. Set the parameter to an applicable object from the object list.

Returns

This system function returns the difference in days, hours, minutes, and seconds between two dates to the objects identified by *Days*, *Hours*, *Minutes*, and *Seconds*, respectively.

Was Form Record Fetched

Description

Use this system function to determine whether the interactive engine for the current form fetched a form record.

Parameters

Parameter	Description
<i>Return To</i>	Input, required. The object to which to return the value. Set the parameter to an applicable object from the object list.

Returns

This system function can return these values:

Value	Description
0	Either no fetch attempt was made, or a fetch attempt failed.
1	A record was fetched successfully.

Messaging

You should use only one of the system functions in this group: **Send Message Extended**. The other system functions in the Messaging folder are intended for internal PeopleSoft development use only.

Send Message Extended

Description

This system function enables your application to send email messages to users, groups, and so forth.

Parameters

Parameter	Description
<i>To recipient</i>	Input, optional. The account or accounts to which to send the email.
<i>Cc recipient</i>	Input, optional. The account or accounts to which to send a courtesy copy of the email.
<i>Bcc recipient</i>	Input, optional. The account or accounts to which to send a blind courtesy copy of the email.
<i>Mailbox</i>	Input, required. The mailbox name to which to deliver the email. The mailbox is used only if the mail is delivered to Work Center. For mail delivered externally (such as SMTP mail), this parameter is ignored. Set the parameter to a specific mailbox or to an applicable object from the object list.
<i>Subject</i>	Input, required. The text to display in the subject line of the email. If the value is <i><Blank></i> or <i><Zero></i> , and you are basing the email on a DD item using the <i>Message</i> parameter, then the system sets the subject line to the DD item description, if one exists. Set the parameter to an alphanumeric constant (<i><Literal></i>), <i><Blank></i> , <i><Zero></i> , or an applicable object from the object list.
<i>Text</i>	Input, required. The text to display in the body of the email. Set the parameter to an alphanumeric constant (<i><Literal></i>), <i><Blank></i> , <i><Zero></i> , or an applicable object from the object list.
<i>Shortcut</i>	Input, required. A link to an EnterpriseOne application. Set the parameter to the application to which to link or to <i><None></i> .
<i>Message</i>	Input, optional. The text to display in the body of the email, based on a DD item glossary. The recipient formatting preferences (for dates, times, and numeric values) as well as language preference (should a translation for this DD item be available) are used when composing the text that represents the message. Set the parameter to the DD item you want to use, or to <i><None></i> .
<i>Media Object Name</i>	Input, optional. The name of the media object to include in the email. Set the parameter to an applicable object from the object list, or to <i><None></i> .
<i>Media Object Key</i>	Input, optional. The key of the media object to include in the email. Set the parameter to an applicable object from the object list, or to <i><None></i> .

Additional Notes

The **Send Message Extended** system function supports multiple ways to define the recipient of a mail. You can dictate that the message is for a limited group (such as individuals, distribution lists, and so forth), or you can make the recipients dynamic. The delivery method is based on each user's email preferences. You must send the message using at least one of the recipient parameters, although which one you use is immaterial to the system.

When mapping a recipient parameter, these options are available:

- *AB Number*

To send a message to a single user, enter the address book number of a user as the recipient. The mail will be sent to the default contact (contact number 0) for this address book number. Set the parameter to an applicable object from the object list.

Note. EnterpriseOne version 8.10 applications do not employ contacts; therefore, email is sent directly to a user based on the address book number.

- *Contact*

To send a message to an individual in a user's contact list, enter the address book number of a user and then the number of the contact. Set the parameters to an applicable object from the object list.

Note. EnterpriseOne version 8.10 applications do not employ contacts; therefore, this parameter has no effect.

- *Grouped Distribution List*

To send a message to the members of a distribution list, enter the address book number of the list and its structure type. Set the parameters to an applicable object from the object list.

- *Hierarchical Distribution List*

To send a message to the members of a hierarchical distribution list, enter the distribution list structure type, and the address book number of the node to start from in the list. Set the parameters to an applicable object from the object list.

Note. This option is available only from within the EnterpriseOne workflow modeler.

- *SMTP Address*

To send a message to a single user, enter the SMTP address of the user as the recipient. Set the parameter to an applicable object from the object list.

- *Define Dynamic Recipient*

This option enables the selection of any kind of recipient at runtime, as opposed to choosing the kind of recipient at design time (AB Number, Contact, Grouped Distribution List, Hierarchy Distribution List, or SMTP Address).

All the parameters must be mapped to objects from the available object list. At runtime, the recipient is chosen dynamically based on the value of the *Recipient Type*:

- *Recipient Type* is '00':

This is the equivalent of selecting <None>.

- *Recipient Type* is '01':

This is the equivalent of selecting <Contact>. The subfields *Address book Number* and *Contact Number* are used to determine the recipient.

- *Recipient Type* is '02':

This is the equivalent of selecting <AB Number>. The sub field *Address book Number* is used to determine the recipient.

- *Recipient Type* is '03':

This is the equivalent of selecting <Grouped Distribution List>. The subfields *Address book Number* and *Structure Type* are used to determine the recipients where *Address book Number* is the AB number for the distribution list and *Structure Type* is the organizational structure, based on UDC 01/TS.

- *Recipient Type* is '04':

This is the equivalent of selecting *<Hierarchical Distribution List>*. The subfields *Address book Number* and *Structure Type* are used to determine the recipients where *Structure Type* is the structure and list to use and *Address book Number* is the point in the hierarchy from which to start.

Note. This hierarchical resolution is available only when you send the email to the PeopleSoft EnterpriseOne work center.

- *Recipient Type* is '05':

This is the equivalent of selecting *<SMTP Address>*. The sub field *SMTP Address* will be used to determine the recipient.

- *Other values*

Do not use other values, as they are reserved for future use. The list of supported recipient types is defined by the UDC 98/SM.

- *None*

To not specify a recipient (use *None* when a recipient is optional).

The body of the email can be preset text (*Text*), or can be based on a DD item (*<Message>*). In either case, you can include a media object (*<Media Object Name>* and *<Media Object Key>*) and or a link directly to an EnterpriseOne application (*<Shortcut>*) as well.

Attachments can be sent with the mail, by providing the *Media Object Name* and *Media Object Key* parameters.

The system function will retrieve the attachments stored within the Media Object specified, and add the data to the mail sent. Only the Media Object 'RTF Text' and 'URL File' attachment types are supported.

Mail Merge & Doc Gen (Web Only)

These system functions enable you to automate mail merge and document generation tasks. They constitute a part of a larger process, as this outline illustrates:

1. Create an RTF in Microsoft Word to use as the template for a merge.

Use Word's field feature to indicate where to place text at merge time. You can create multiple RTF files and then use the CompositeGeneration business function (BSFN B980043) to create a single template from them.

2. Upload the template.

3. Call the **Get XML Data Model** system function to process the template.

Among other files, the system function creates an XML file to populate with data for the merge.

4. Create a business function to populate the XML file and then run it.

5. Run the merge.

Delete Document

Description

Use this system function to delete a generated document.

Parameters

Parameter	Description
<i>Document ID</i>	Input, required. The document to affect. Set the parameter to an applicable object from the object list.
<i>Return Code</i>	Output (string), required. The object to which to assign the code that indicates the success of the delete. Set the parameter to an applicable object from the object list. failed success

Returns

This system function can return these values:

Value	Description
FAILED	Indicates the process was unsuccessful.
SUCCESS	Indicates the process completed normally.

Display Document

Description

Use this system function to display a generated document to the user.

Parameters

Parameter	Description
<i>Document ID</i>	Input, required. The document to affect. Set the parameter to an applicable object from the object list.
<i>Return Code</i>	Output (string), required. The object to which to assign the code that indicates the success of the display. Set the parameter to an applicable object from the object list.

Returns

This system function can return these values:

Value	Description
FAILED	Indicates the process was unsuccessful.
SUCCESS	Indicates the process completed normally.

Download Template

Description

Use this system function to download a mail merge template in RTF format so you can edit it.

Parameters

Parameter	Description
<i>Template ID</i>	Input, required. The template to affect. Set the parameter to an applicable object from the object list.
<i>Return Code</i>	Output (string), required. The object to which to assign the code that indicates the success of the download. Set the parameter to an applicable object from the object list.

Returns

This system function can return these values:

Value	Description
FAILED	Indicates the process was unsuccessful.
SUCCESS	Indicates the process completed normally.

Download Template for Doc Gen

Description

Use this system function to download a document generation template in RTF format so you can edit it.

Parameters

Parameter	Description
<i>Template ID</i>	Input, required. The template to affect. Set the parameter to an applicable object from the object list.
<i>Return Code</i>	Output (string), required. The object to which to assign the code that indicates the success of the download. Set the parameter to an applicable object from the object list.
<i>Version</i>	Input, required. The version of the template to download. Set the parameter to an applicable object from the object list.

Returns

This system function can return these values:

Value	Description
FAILED	Indicates the process was unsuccessful.
SUCCESS	Indicates the process completed normally.

Get XML Data Model

Description

Use this system function to process a template after uploading. This system function breaks the template into three parts: an XSL file, an XML file, and one or more image files (if any images were included in the template). Then, use a business function to populate the XML file with the data to merge.

Parameters

Parameter	Description
<i>Template ID</i>	Input, required. The template to affect. Set the parameter to an applicable object from the object list.
<i>XML Data ID</i>	Input, required. The ID to assign to the XML file. Set the parameter to an applicable object from the object list.
<i>Data Type</i>	Input, required. The template type. Set the parameter to <i><Mail Merge></i> or <i><Doc Gen></i> .
<i>Status</i>	Output (string), required. The object to which to assign the code that indicates the success of the acquire. Set the parameter to an applicable object from the object list.

Returns

This system function can return these values:

Value	Description
FAILED	Indicates the process was unsuccessful.
SUCCESS	Indicates the process completed normally.

Run Doc Gen and Display

Description

Use this system function to run a document generation operation and display the results to the user. You can choose to save the resulting document.

Parameters

Parameter	Description
<i>Template ID</i>	Input, required. The document generation template to use. Use an already-downloaded template. Set the parameter to an applicable object from the object list.
<i>Data File ID</i>	Input, required. The document generation template to use. Set the parameter to an applicable object from the object list.
<i>Save</i>	Input, required. An indicator of whether to save the results of the operation to a separate file. Set the parameter to <code><TRUE></code> or <code><FALSE></code> .
<i>Document ID</i>	Input, required. The ID to assign to the document, should you choose to save it. Set the parameter to an applicable object from the object list.
<i>Status</i>	Output (string), required. The object to which to assign the code that indicates the success of the operation. Set the parameter to an applicable object from the object list.

Returns

This system function can return these values:

Value	Description
FAILED	Indicates the process was unsuccessful.
SUCCESS	Indicates the process completed normally.

See Also

[Appendix A, “System Functions in Form Design Aid,” Download Template for Doc Gen, page 305](#)

[Appendix A, “System Functions in Form Design Aid,” Get XML Data Model, page 306](#)

Run Mail Merge and Display

Description

Use this system function to run a mail merge operation that consists of the first two variable sets in the XML file, and then display the results to the user. The resulting document is saved.

Parameters

Parameter	Description
<i>Template ID</i>	Input, required. The document generation template to use. Use an already-downloaded template. Set the parameter to an applicable object from the object list.
<i>Data File ID</i>	Input, required. The mail merge template to use. Set the parameter to an applicable object from the object list.
<i>Save</i>	Input, required. An indicator of whether to save the results of the operation to a separate file. Set the parameter to <code><TRUE></code> or <code><FALSE></code> .
<i>Document ID</i>	Input, required. The ID to assign to the resulting document. Set the parameter to an applicable object from the object list.
<i>Status</i>	Output (string), required. The object to which to assign the code that indicates the success of the operation. Set the parameter to an applicable object from the object list.

Returns

This system function can return these values:

Value	Description
FAILED	Indicates the process was unsuccessful.
SUCCESS	Indicates the process completed normally.

See Also

[Appendix A, “System Functions in Form Design Aid,” Download Template, page 305](#)

[Appendix A, “System Functions in Form Design Aid,” Get XML Data Model, page 306](#)

Run Multiple Mail Merge

Description

Use this system function to run a full mail merge operation.

Parameters

Parameter	Description
<i>Template ID</i>	Input, required. The document generation template to use. Use an already-downloaded template. Set the parameter to an applicable object from the object list.
<i>Data File ID</i>	Input, required. The mail merge template to use. Set the parameter to an applicable object from the object list.
<i>Document ID</i>	Input, required. The ID to assign to the resulting document. Set the parameter to an applicable object from the object list.

Additional Notes

This system function launches an operation that merges all of the data sets in the XML file with the template to create a .pdf. The process runs asynchronously, so it will not overtax the web server. Before running a system function such as **Display Document**, ensure that the operation has completed. To do so, perform a table I/O on column FNDFUF1 in table F980042. If the return string is PENDING, the operation is still running. If the return string is SUCCESS, then the operation is complete.

See Also

[Appendix A, “System Functions in Form Design Aid,” Download Template, page 305](#)

[Appendix A, “System Functions in Form Design Aid,” Get XML Data Model, page 306](#)

Upload Template

Description

After designing it, use this system function to upload a mail merge template.

Parameters

Parameter	Description
<i>Template ID</i>	Input, required. The template to affect. Set the parameter to an applicable object from the object list.
<i>Return Code</i>	Output (string), required. The object to which to assign the code that indicates the success of the upload. Set the parameter to an applicable object from the object list.
<i>File (full path)</i>	Output (string), required. The object to which to assign the full path name of the uploaded file. Set the parameter to an applicable object from the object list.

Returns

This system function can return these values:

Value	Description
FAILED	Indicates the process was unsuccessful.
SUCCESS	Indicates the process completed normally.

Upload Template for Doc Gen

Description

After designing it, use this system function to upload a document generation template.

Parameters

Parameter	Description
<i>Template ID</i>	Input, required. The template to affect. Set the parameter to an applicable object from the object list.
<i>Return Code</i>	Output (string), required. The object to which to assign the code that indicates the success of the upload. Set the parameter to an applicable object from the object list.
<i>File (full path)</i>	Output (string), required. The object to which to assign the full path name of the uploaded file. Set the parameter to an applicable object from the object list.
<i>Version</i>	Input, required. The version of the template to upload. Set the parameter to an applicable object from the object list.

Returns

This system function can return these values:

Value	Description
FAILED	Indicates the process was unsuccessful.
SUCCESS	Indicates the process completed normally.

Glossary of PeopleSoft Terms

absence entitlement	This element defines rules for granting paid time off for valid absences, such as sick time, vacation, and maternity leave. An absence entitlement element defines the entitlement amount, frequency, and entitlement period.
absence take	This element defines the conditions that must be met before a payee is entitled to take paid time off.
academic career	In PeopleSoft Enterprise Campus Solutions, all course work that a student undertakes at an academic institution and that is grouped in a single student record. For example, a university that has an undergraduate school, a graduate school, and various professional schools might define several academic careers—an undergraduate career, a graduate career, and separate careers for each professional school (law school, medical school, dental school, and so on).
academic institution	In PeopleSoft Enterprise Campus Solutions, an entity (such as a university or college) that is independent of other similar entities and that has its own set of rules and business processes.
academic organization	In PeopleSoft Enterprise Campus Solutions, an entity that is part of the administrative structure within an academic institution. At the lowest level, an academic organization might be an academic department. At the highest level, an academic organization can represent a division.
academic plan	In PeopleSoft Enterprise Campus Solutions, an area of study—such as a major, minor, or specialization—that exists within an academic program or academic career.
academic program	In PeopleSoft Enterprise Campus Solutions, the entity to which a student applies and is admitted and from which the student graduates.
accounting class	In PeopleSoft Enterprise Performance Management, the accounting class defines how a resource is treated for generally accepted accounting practices. The Inventory class indicates whether a resource becomes part of a balance sheet account, such as inventory or fixed assets, while the Non-inventory class indicates that the resource is treated as an expense of the period during which it occurs.
accounting date	The accounting date indicates when a transaction is recognized, as opposed to the date the transaction actually occurred. The accounting date and transaction date can be the same. The accounting date determines the period in the general ledger to which the transaction is to be posted. You can only select an accounting date that falls within an open period in the ledger to which you are posting. The accounting date for an item is normally the invoice date.
accounting split	The accounting split method indicates how expenses are allocated or divided among one or more sets of accounting ChartFields.
accumulator	You use an accumulator to store cumulative values of defined items as they are processed. You can accumulate a single value over time or multiple values over time. For example, an accumulator could consist of all voluntary deductions, or all company deductions, enabling you to accumulate amounts. It allows total flexibility for time periods and values accumulated.
action reason	The reason an employee's job or employment information is updated. The action reason is entered in two parts: a personnel action, such as a promotion, termination, or change from one pay group to another—and a reason for that action. Action reasons are used by PeopleSoft Human Resources, PeopleSoft Benefits Administration,

	PeopleSoft Stock Administration, and the COBRA Administration feature of the Base Benefits business process.
action template	In PeopleSoft Receivables, outlines a set of escalating actions that the system or user performs based on the period of time that a customer or item has been in an action plan for a specific condition.
activity	<p>In PeopleSoft Enterprise Learning Management, an instance of a catalog item (sometimes called a class) that is available for enrollment. The activity defines such things as the costs that are associated with the offering, enrollment limits and deadlines, and waitlisting capacities.</p> <p>In PeopleSoft Enterprise Performance Management, the work of an organization and the aggregation of actions that are used for activity-based costing.</p> <p>In PeopleSoft Project Costing, the unit of work that provides a further breakdown of projects—usually into specific tasks.</p> <p>In PeopleSoft Workflow, a specific transaction that you might need to perform in a business process. Because it consists of the steps that are used to perform a transaction, it is also known as a step map.</p>
address usage	In PeopleSoft Enterprise Campus Solutions, a grouping of address types defining the order in which the address types are used. For example, you might define an address usage code to process addresses in the following order: billing address, dormitory address, home address, and then work address.
adjustment calendar	In PeopleSoft Enterprise Campus Solutions, the adjustment calendar controls how a particular charge is adjusted on a student's account when the student drops classes or withdraws from a term. The charge adjustment is based on how much time has elapsed from a predetermined date, and it is determined as a percentage of the original charge amount.
administrative function	In PeopleSoft Enterprise Campus Solutions, a particular functional area that processes checklists, communication, and comments. The administrative function identifies which variable data is added to a person's checklist or communication record when a specific checklist code, communication category, or comment is assigned to the student. This key data enables you to trace that checklist, communication, or comment back to a specific processing event in a functional area.
admit type	In PeopleSoft Enterprise Campus Solutions, a designation used to distinguish first-year applications from transfer applications.
agreement	In PeopleSoft eSettlements, provides a way to group and specify processing options, such as payment terms, pay from a bank, and notifications by a buyer and supplier location combination.
allocation rule	In PeopleSoft Enterprise Incentive Management, an expression within compensation plans that enables the system to assign transactions to nodes and participants. During transaction allocation, the allocation engine traverses the compensation structure from the current node to the root node, checking each node for plans that contain allocation rules.
alternate account	A feature in PeopleSoft General Ledger that enables you to create a statutory chart of accounts and enter statutory account transactions at the detail transaction level, as required for recording and reporting by some national governments.
analysis database	In PeopleSoft Enterprise Campus Solutions, database tables that store large amounts of student information that may not appear in standard report formats. The analysis database tables contain keys for all objects in a report that an application program can use to reference other student-record objects that are not contained in the printed report. For instance, the analysis database contains data on courses that are considered for satisfying a requirement but that are rejected. It also contains information on

	courses captured by global limits. An analysis database is used in PeopleSoft Enterprise Academic Advisement.
AR specialist	Abbreviation for <i>receivables specialist</i> . In PeopleSoft Receivables, an individual in who tracks and resolves deductions and disputed items.
arbitration plan	In PeopleSoft Enterprise Pricer, defines how price rules are to be applied to the base price when the transaction is priced.
assessment rule	In PeopleSoft Receivables, a user-defined rule that the system uses to evaluate the condition of a customer's account or of individual items to determine whether to generate a follow-up action.
asset class	An asset group used for reporting purposes. It can be used in conjunction with the asset category to refine asset classification.
attribute/value pair	In PeopleSoft Directory Interface, relates the data that makes up an entry in the directory information tree.
audience	In PeopleSoft Enterprise Campus Solutions, a segment of the database that relates to an initiative, or a membership organization that is based on constituent attributes rather than a dues-paying structure. Examples of audiences include the Class of '65 and Undergraduate Arts & Sciences.
authentication server	A server that is set up to verify users of the system.
base time period	In PeopleSoft Business Planning, the lowest level time period in a calendar.
benchmark job	In PeopleSoft Workforce Analytics, a benchmark job is a job code for which there is corresponding salary survey data from published, third-party sources.
billing career	In PeopleSoft Enterprise Campus Solutions, the one career under which other careers are grouped for billing purposes if a student is active simultaneously in multiple careers.
bio bit or bio brief	In PeopleSoft Enterprise Campus Solutions, a report that summarizes information stored in the system about a particular constituent. You can generate standard or specialized reports.
book	In PeopleSoft Asset Management, used for storing financial and tax information, such as costs, depreciation attributes, and retirement information on assets.
branch	A tree node that rolls up to nodes above it in the hierarchy, as defined in PeopleSoft Tree Manager.
budgetary account only	An account used by the system only and not by users; this type of account does not accept transactions. You can only budget with this account. Formerly called "system-maintained account."
budget check	In commitment control, the processing of source transactions against control budget ledgers, to see if they pass, fail, or pass with a warning.
budget control	In commitment control, budget control ensures that commitments and expenditures don't exceed budgets. It enables you to track transactions against corresponding budgets and terminate a document's cycle if the defined budget conditions are not met. For example, you can prevent a purchase order from being dispatched to a vendor if there are insufficient funds in the related budget to support it.
budget period	The interval of time (such as 12 months or 4 quarters) into which a period is divided for budgetary and reporting purposes. The ChartField allows maximum flexibility to define operational accounting time periods without restriction to only one calendar.

business event	<p>In PeopleSoft Receivables, defines the processing characteristics for the Receivable Update process for a draft activity.</p> <p>In PeopleSoft Sales Incentive Management, an original business transaction or activity that may justify the creation of a PeopleSoft Enterprise Incentive Management event (a sale, for example).</p>
business unit	A corporation or a subset of a corporation that is independent with regard to one or more operational or accounting functions.
buyer	In PeopleSoft eSettlements, an organization (or business unit, as opposed to an individual) that transacts with suppliers (vendors) within the system. A buyer creates payments for purchases that are made in the system.
campus	In PeopleSoft Enterprise Campus Solutions, an entity that is usually associated with a distinct physical administrative unit, that belongs to a single academic institution, that uses a unique course catalog, and that produces a common transcript for students within the same academic career.
catalog item	In PeopleSoft Enterprise Learning Management, a specific topic that a learner can study and have tracked. For example, "Introduction to Microsoft Word." A catalog item contains general information about the topic and includes a course code, description, categorization, keywords, and delivery methods. A catalog item can have one or more learning activities.
catalog map	In PeopleSoft Catalog Management, translates values from the catalog source data to the format of the company's catalog.
catalog partner	In PeopleSoft Catalog Management, shares responsibility with the enterprise catalog manager for maintaining catalog content.
categorization	Associates partner offerings with catalog offerings and groups them into enterprise catalog categories.
category	In PeopleSoft Enterprise Campus Solutions, a broad grouping to which specific comments or communications (contexts) are assigned. Category codes are also linked to 3C access groups so that you can assign data-entry or view-only privileges across functions.
channel	In PeopleSoft MultiChannel Framework, email, chat, voice (computer telephone integration [CTI]), or a generic event.
ChartField	A field that stores a chart of accounts, resources, and so on, depending on the PeopleSoft application. ChartField values represent individual account numbers, department codes, and so forth.
ChartField balancing	You can require specific ChartFields to match up (balance) on the debit and the credit side of a transaction.
ChartField combination edit	The process of editing journal lines for valid ChartField combinations based on user-defined rules.
ChartKey	One or more fields that uniquely identify each row in a table. Some tables contain only one field as the key, while others require a combination.
checkbook	In PeopleSoft Promotions Management, enables you to view financial data (such as planned, incurred, and actual amounts) that is related to funds and trade promotions.
checklist code	In PeopleSoft Enterprise Campus Solutions, a code that represents a list of planned or completed action items that can be assigned to a staff member, volunteer, or unit. Checklists enable you to view all action assignments on one page.

class	<p>In PeopleSoft Enterprise Campus Solutions, a specific offering of a course component within an academic term.</p> <p>See also <i>course</i>.</p>
Class ChartField	<p>A ChartField value that identifies a unique appropriation budget key when you combine it with a fund, department ID, and program code, as well as a budget period. Formerly called <i>sub-classification</i>.</p>
clearance	<p>In PeopleSoft Enterprise Campus Solutions, the period of time during which a constituent in PeopleSoft Contributor Relations is approved for involvement in an initiative or an action. Clearances are used to prevent development officers from making multiple requests to a constituent during the same time period.</p>
clone	<p>In PeopleCode, to make a unique copy. In contrast, to <i>copy</i> may mean making a new reference to an object, so if the underlying object is changed, both the copy and the original change.</p>
cohort	<p>In PeopleSoft Enterprise Campus Solutions, the highest level of the three-level classification structure that you define for enrollment management. You can define a cohort level, link it to other levels, and set enrollment target numbers for it.</p> <p>See also <i>population</i> and <i>division</i>.</p>
collection	<p>To make a set of documents available for searching in Verity, you must first create at least one collection. A collection is set of directories and files that allow search application users to use the Verity search engine to quickly find and display source documents that match search criteria. A collection is a set of statistics and pointers to the source documents, stored in a proprietary format on a file server. Because a collection can only store information for a single location, PeopleSoft maintains a set of collections (one per language code) for each search index object.</p>
collection rule	<p>In PeopleSoft Receivables, a user-defined rule that defines actions to take for a customer based on both the amount and the number of days past due for outstanding balances.</p>
comm key	<p>See <i>communication key</i>.</p>
communication key	<p>In PeopleSoft Enterprise Campus Solutions, a single code for entering a combination of communication category, communication context, communication method, communication direction, and standard letter code. Communication keys (also called <i>comm keys</i> or <i>speed keys</i>) can be created for background processes as well as for specific users.</p>
compensation object	<p>In PeopleSoft Enterprise Incentive Management, a node within a compensation structure. Compensation objects are the building blocks that make up a compensation structure's hierarchical representation.</p>
compensation structure	<p>In PeopleSoft Enterprise Incentive Management, a hierarchical relationship of compensation objects that represents the compensation-related relationship between the objects.</p>
condition	<p>In PeopleSoft Receivables, occurs when there is a change of status for a customer's account, such as reaching a credit limit or exceeding a user-defined balance due.</p>
configuration parameter catalog	<p>Used to configure an external system with PeopleSoft. For example, a configuration parameter catalog might set up configuration and communication parameters for an external server.</p>
configuration plan	<p>In PeopleSoft Enterprise Incentive Management, configuration plans hold allocation information for common variables (not incentive rules) and are attached to a node without a participant. Configuration plans are not processed by transactions.</p>

constituents	In PeopleSoft Enterprise Campus Solutions, friends, alumni, organizations, foundations, or other entities affiliated with the institution, and about which the institution maintains information. The constituent types delivered with PeopleSoft Enterprise Contributor Relations Solutions are based on those defined by the Council for the Advancement and Support of Education (CASE).
content reference	Content references are pointers to content registered in the portal registry. These are typically either URLs or iScripts. Content references fall into three categories: target content, templates, and template pagelets.
context	<p>In PeopleCode, determines which buffer fields can be contextually referenced and which is the current row of data on each scroll level when a PeopleCode program is running.</p> <p>In PeopleSoft Enterprise Campus Solutions, a specific instance of a comment or communication. One or more contexts are assigned to a category, which you link to 3C access groups so that you can assign data-entry or view-only privileges across functions.</p> <p>In PeopleSoft Enterprise Incentive Management, a mechanism that is used to determine the scope of a processing run. PeopleSoft Enterprise Incentive Management uses three types of context: plan, period, and run-level.</p>
control table	Stores information that controls the processing of an application. This type of processing might be consistent throughout an organization, or it might be used only by portions of the organization for more limited sharing of data.
cost profile	A combination of a receipt cost method, a cost flow, and a deplete cost method. A profile is associated with a cost book and determines how items in that book are valued, as well as how the material movement of the item is valued for the book.
cost row	A cost transaction and amount for a set of ChartFields.
course	<p>In PeopleSoft Enterprise Campus Solutions, a course that is offered by a school and that is typically described in a course catalog. A course has a standard syllabus and credit level; however, these may be modified at the class level. Courses can contain multiple components such as lecture, discussion, and lab.</p> <p>See also <i>class</i>.</p>
course share set	In PeopleSoft Enterprise Campus Solutions, a tag that defines a set of requirement groups that can share courses. Course share sets are used in PeopleSoft Enterprise Academic Advisement.
current learning	In PeopleSoft Enterprise Learning Management, a self-service repository for all of a learner's in-progress learning activities and programs.
data acquisition	In PeopleSoft Enterprise Incentive Management, the process during which raw business transactions are acquired from external source systems and fed into the operational data store (ODS).
data elements	<p>Data elements, at their simplest level, define a subset of data and the rules by which to group them.</p> <p>For Workforce Analytics, data elements are rules that tell the system what measures to retrieve about your workforce groups.</p>
dataset	A data grouping that enables role-based filtering and distribution of data. You can limit the range and quantity of data that is displayed for a user by associating dataset rules with user roles. The result of dataset rules is a set of data that is appropriate for the user's roles.
delivery method	In PeopleSoft Enterprise Learning Management, identifies the primary type of delivery method in which a particular learning activity is offered. Also provides

	<p>default values for the learning activity, such as cost and language. This is primarily used to help learners search the catalog for the type of delivery from which they learn best. Because PeopleSoft Enterprise Learning Management is a blended learning system, it does not enforce the delivery method.</p> <p>In PeopleSoft Supply Chain Management, identifies the method by which goods are shipped to their destinations (such as truck, air, rail, and so on). The delivery method is specified when creating shipment schedules.</p>
delivery method type	In PeopleSoft Enterprise Learning Management, identifies how learning activities can be delivered—for example, through online learning, classroom instruction, seminars, books, and so forth—in an organization. The type determines whether the delivery method includes scheduled components.
directory information tree	In PeopleSoft Directory Interface, the representation of a directory's hierarchical structure.
division	<p>In PeopleSoft Enterprise Campus Solutions, the lowest level of the three-level classification structure that you define in PeopleSoft Enterprise Recruiting and Admissions for enrollment management. You can define a division level, link it to other levels, and set enrollment target numbers for it.</p> <p>See also <i>population</i> and <i>cohort</i>.</p>
document sequencing	A flexible method that sequentially numbers the financial transactions (for example, bills, purchase orders, invoices, and payments) in the system for statutory reporting and for tracking commercial transaction activity.
dynamic detail tree	A tree that takes its detail values—dynamic details—directly from a table in the database, rather than from a range of values that are entered by the user.
edit table	A table in the database that has its own record definition, such as the Department table. As fields are entered into a PeopleSoft application, they can be validated against an edit table to ensure data integrity throughout the system.
effective date	A method of dating information in PeopleSoft applications. You can predate information to add historical data to your system, or postdate information in order to enter it before it actually goes into effect. By using effective dates, you don't delete values; you enter a new value with a current effective date.
EIM ledger	Abbreviation for <i>Enterprise Incentive Management ledger</i> . In PeopleSoft Enterprise Incentive Management, an object to handle incremental result gathering within the scope of a participant. The ledger captures a result set with all of the appropriate traces to the data origin and to the processing steps of which it is a result.
elimination set	In PeopleSoft General Ledger, a related group of intercompany accounts that is processed during consolidations.
entry event	In PeopleSoft General Ledger, Receivables, Payables, Purchasing, and Billing, a business process that generates multiple debits and credits resulting from single transactions to produce standard, supplemental accounting entries.
equitization	In PeopleSoft General Ledger, a business process that enables parent companies to calculate the net income of subsidiaries on a monthly basis and adjust that amount to increase the investment amount and equity income amount before performing consolidations.
equity item limit	In PeopleSoft Enterprise Campus Solutions, the amounts of funds set by the institution to be awarded with discretionary or gift funds. The limit could be reduced by amounts equal to such things as expected family contribution (EFC) or parent contribution. Students are packaged by Equity Item Type Groups and Related Equity Item Types. This limit can be used to assure that similar student populations are packaged equally.

event	<p>A predefined point either in the Component Processor flow or in the program flow. As each point is encountered, the event activates each component, triggering any PeopleCode program that is associated with that component and that event. Examples of events are FieldChange, SavePreChange, and RowDelete.</p> <p>In PeopleSoft Human Resources, also refers to an incident that affects benefits eligibility.</p>
event propagation process	<p>In PeopleSoft Sales Incentive Management, a process that determines, through logic, the propagation of an original PeopleSoft Enterprise Incentive Management event and creates a derivative (duplicate) of the original event to be processed by other objects. Sales Incentive Management uses this mechanism to implement splits, roll-ups, and so on. Event propagation determines who receives the credit.</p>
exception	<p>In PeopleSoft Receivables, an item that either is a deduction or is in dispute.</p>
exclusive pricing	<p>In PeopleSoft Order Management, a type of arbitration plan that is associated with a price rule. Exclusive pricing is used to price sales order transactions.</p>
fact	<p>In PeopleSoft applications, facts are numeric data values from fields from a source database as well as an analytic application. A fact can be anything you want to measure your business by, for example, revenue, actual, budget data, or sales numbers. A fact is stored on a fact table.</p>
financial aid term	<p>In PeopleSoft Enterprise Campus Solutions, a combination of a period of time that the school determines as an instructional accounting period and an academic career. It is created and defined during the setup process. Only terms eligible for financial aid are set up for each financial aid career.</p>
forecast item	<p>A logical entity with a unique set of descriptive demand and forecast data that is used as the basis to forecast demand. You create forecast items for a wide range of uses, but they ultimately represent things that you buy, sell, or use in your organization and for which you require a predictable usage.</p>
fund	<p>In PeopleSoft Promotions Management, a budget that can be used to fund promotional activity. There are four funding methods: top down, fixed accrual, rolling accrual, and zero-based accrual.</p>
gap	<p>In PeopleSoft Enterprise Campus Solutions, an artificial figure that sets aside an amount of unmet financial aid need that is not funded with Title IV funds. A gap can be used to prevent fully funding any student to conserve funds, or it can be used to preserve unmet financial aid need so that institutional funds can be awarded.</p>
generic process type	<p>In PeopleSoft Process Scheduler, process types are identified by a generic process type. For example, the generic process type SQR includes all SQR process types, such as SQR process and SQR report.</p>
gift table	<p>In PeopleSoft Enterprise Campus Solutions, a table or so-called <i>donor pyramid</i> describing the number and size of gifts that you expect will be needed to successfully complete the campaign in PeopleSoft Contributor Relations. The gift table enables you to estimate the number of donors and prospects that you need at each gift level to reach the campaign goal.</p>
GL business unit	<p>Abbreviation for <i>general ledger business unit</i>. A unit in an organization that is an independent entity for accounting purposes. It maintains its own set of accounting books.</p> <p>See also <i>business unit</i>.</p>
GL entry template	<p>Abbreviation for <i>general ledger entry template</i>. In PeopleSoft Enterprise Campus Solutions, a template that defines how a particular item is sent to the general ledger. An item-type maps to the general ledger, and the GL entry template can involve multiple general ledger accounts. The entry to the general ledger is further controlled</p>

by high-level flags that control the summarization and the type of accounting—that is, accrual or cash.

GL Interface process

Abbreviation for *General Ledger Interface process*. In PeopleSoft Enterprise Campus Solutions, a process that is used to send transactions from PeopleSoft Enterprise Student Financials to the general ledger. Item types are mapped to specific general ledger accounts, enabling transactions to move to the general ledger when the GL Interface process is run.

group

In PeopleSoft Billing and Receivables, a posting entity that comprises one or more transactions (items, deposits, payments, transfers, matches, or write-offs).

In PeopleSoft Human Resources Management and Supply Chain Management, any set of records that are associated under a single name or variable to run calculations in PeopleSoft business processes. In PeopleSoft Time and Labor, for example, employees are placed in groups for time reporting purposes.

incentive object

In PeopleSoft Enterprise Incentive Management, the incentive-related objects that define and support the PeopleSoft Enterprise Incentive Management calculation process and results, such as plan templates, plans, results data, user interaction objects, and so on.

incentive rule

In PeopleSoft Sales Incentive Management, the commands that act on transactions and turn them into compensation. A rule is one part in the process of turning a transaction into compensation.

incur

In PeopleSoft Promotions Management, to become liable for a promotional payment. In other words, you owe that amount to a customer for promotional activities.

initiative

In PeopleSoft Enterprise Campus Solutions, the basis from which all advancement plans are executed. It is an organized effort targeting a specific constituency, and it can occur over a specified period of time with specific purposes and goals. An initiative can be a campaign, an event, an organized volunteer effort, a membership drive, or any other type of effort defined by the institution. Initiatives can be multipart, and they can be related to other initiatives. This enables you to track individual parts of an initiative, as well as entire initiatives.

inquiry access

In PeopleSoft Enterprise Campus Solutions, a type of security access that permits the user only to view data.

See also *update access*.

institution

In PeopleSoft Enterprise Campus Solutions, an entity (such as a university or college) that is independent of other similar entities and that has its own set of rules and business processes.

item

In PeopleSoft Inventory, a tangible commodity that is stored in a business unit (shipped from a warehouse).

In PeopleSoft Demand Planning, Inventory Policy Planning, and Supply Planning, a noninventory item that is designated as being used for planning purposes only. It can represent a family or group of inventory items. It can have a planning bill of material (BOM) or planning routing, and it can exist as a component on a planning BOM. A planning item cannot be specified on a production or engineering BOM or routing, and it cannot be used as a component in a production. The quantity on hand will never be maintained.

In PeopleSoft Receivables, an individual receivable. An item can be an invoice, a credit memo, a debit memo, a write-off, or an adjustment.

item shuffle

In PeopleSoft Enterprise Campus Solutions, a process that enables you to change a payment allocation without having to reverse the payment.

joint communication	In PeopleSoft Enterprise Campus Solutions, one letter that is addressed jointly to two people. For example, a letter might be addressed to both Mr. Sudhir Awat and Ms. Samantha Mortelli. A relationship must be established between the two individuals in the database, and at least one of the individuals must have an ID in the database.
keyword	In PeopleSoft Enterprise Campus Solutions, a term that you link to particular elements within PeopleSoft Student Financials, Financial Aid, and Contributor Relations. You can use keywords as search criteria that enable you to locate specific records in a search dialog box.
KPI	An abbreviation for <i>key performance indicator</i> . A high-level measurement of how well an organization is doing in achieving critical success factors. This defines the data value or calculation upon which an assessment is determined.
LDIF file	Abbreviation for <i>Lightweight Directory Access Protocol (LDAP) Data Interchange Format file</i> . Contains discrepancies between PeopleSoft data and directory data.
learner group	In PeopleSoft Enterprise Learning Management, a group of learners who are linked to the same learning environment. Members of the learner group can share the same attributes, such as the same department or job code. Learner groups are used to control access to and enrollment in learning activities and programs. They are also used to perform group enrollments and mass enrollments in the back office.
learning components	In PeopleSoft Enterprise Learning Management, the foundational building blocks of learning activities. PeopleSoft Enterprise Learning Management supports six basic types of learning components: web-based, session, webcast, test, survey, and assignment. One or more of these learning component types compose a single learning activity.
learning environment	In PeopleSoft Enterprise Learning Management, identifies a set of categories and catalog items that can be made available to learner groups. Also defines the default values that are assigned to the learning activities and programs that are created within a particular learning environment. Learning environments provide a way to partition the catalog so that learners see only those items that are relevant to them.
learning history	In PeopleSoft Enterprise Learning Management, a self-service repository for all of a learner's completed learning activities and programs.
ledger mapping	You use ledger mapping to relate expense data from general ledger accounts to resource objects. Multiple ledger line items can be mapped to one or more resource IDs. You can also use ledger mapping to map dollar amounts (referred to as <i>rates</i>) to business units. You can map the amounts in two different ways: an actual amount that represents actual costs of the accounting period, or a budgeted amount that can be used to calculate the capacity rates as well as budgeted model results. In PeopleSoft Enterprise Warehouse, you can map general ledger accounts to the EW Ledger table.
library section	In PeopleSoft Enterprise Incentive Management, a section that is defined in a plan (or template) and that is available for other plans to share. Changes to a library section are reflected in all plans that use it.
linked section	In PeopleSoft Enterprise Incentive Management, a section that is defined in a plan template but appears in a plan. Changes to linked sections propagate to plans using that section.
linked variable	In PeopleSoft Enterprise Incentive Management, a variable that is defined and maintained in a plan template and that also appears in a plan. Changes to linked variables propagate to plans using that variable.
LMS	Abbreviation for <i>learning management system</i> . In PeopleSoft Enterprise Campus Solutions, LMS is a PeopleSoft Student Records feature that provides a common set of interoperability standards that enable the sharing of instructional content and data between learning and administrative environments.

load	In PeopleSoft Inventory, identifies a group of goods that are shipped together. Load management is a feature of PeopleSoft Inventory that is used to track the weight, the volume, and the destination of a shipment.
local functionality	In PeopleSoft HRMS, the set of information that is available for a specific country. You can access this information when you click the appropriate country flag in the global window, or when you access it by a local country menu.
location	Locations enable you to indicate the different types of addresses—for a company, for example, one address to receive bills, another for shipping, a third for postal deliveries, and a separate street address. Each address has a different location number. The primary location—indicated by a <i>1</i> —is the address you use most often and may be different from the main address.
logistical task	In PeopleSoft Services Procurement, an administrative task that is related to hiring a service provider. Logistical tasks are linked to the service type on the work order so that different types of services can have different logistical tasks. Logistical tasks include both preapproval tasks (such as assigning a new badge or ordering a new laptop) and postapproval tasks (such as scheduling orientation or setting up the service provider email). The logistical tasks can be mandatory or optional. Mandatory preapproval tasks must be completed before the work order is approved. Mandatory postapproval tasks, on the other hand, must be completed before a work order is released to a service provider.
market template	In PeopleSoft Enterprise Incentive Management, additional functionality that is specific to a given market or industry and is built on top of a product category.
mass change	In PeopleSoft Enterprise Campus Solutions, mass change is a SQL generator that can be used to create specialized functionality. Using mass change, you can set up a series of Insert, Update, or Delete SQL statements to perform business functions that are specific to the institution. See also <i>3C engine</i> .
match group	In PeopleSoft Receivables, a group of receivables items and matching offset items. The system creates match groups by using user-defined matching criteria for selected field values.
MCF server	Abbreviation for <i>PeopleSoft MultiChannel Framework server</i> . Comprises the universal queue server and the MCF log server. Both processes are started when <i>MCF Servers</i> is selected in an application server domain configuration.
merchandising activity	In PeopleSoft Promotions Management, a specific discount type that is associated with a trade promotion (such as off-invoice, billback or rebate, or lump-sum payment) that defines the performance that is required to receive the discount. In the industry, you may know this as an offer, a discount, a merchandising event, an event, or a tactic.
meta-SQL	Meta-SQL constructs expand into platform-specific Structured Query Language (SQL) substrings. They are used in functions that pass SQL strings, such as in SQL objects, the <i>SQLExec</i> function, and PeopleSoft Application Engine programs.
metastring	Metastrings are special expressions included in SQL string literals. The metastrings, prefixed with a percent (%) symbol, are included directly in the string literals. They expand at run time into an appropriate substring for the current database platform.
multibook	In PeopleSoft General Ledger, multiple ledgers having multiple-base currencies that are defined for a business unit, with the option to post a single transaction to all base currencies (all ledgers) or to only one of those base currencies (ledgers).
multicurrency	The ability to process transactions in a currency other than the business unit's base currency.

national allowance	In PeopleSoft Promotions Management, a promotion at the corporate level that is funded by nondiscretionary dollars. In the industry, you may know this as a national promotion, a corporate promotion, or a corporate discount.
need	In PeopleSoft Enterprise Campus Solutions, the difference between the cost of attendance (COA) and the expected family contribution (EFC). It is the gap between the cost of attending the school and the student's resources. The financial aid package is based on the amount of financial need. The process of determining a student's need is called <i>need analysis</i> .
node-oriented tree	A tree that is based on a detail structure, but the detail values are not used.
pagelet	Each block of content on the home page is called a pagelet. These pagelets display summary information within a small rectangular area on the page. The pagelet provide users with a snapshot of their most relevant PeopleSoft and non-PeopleSoft content.
participant	In PeopleSoft Enterprise Incentive Management, participants are recipients of the incentive compensation calculation process.
participant object	Each participant object may be related to one or more compensation objects. See also <i>compensation object</i> .
partner	A company that supplies products or services that are resold or purchased by the enterprise.
pay cycle	In PeopleSoft Payables, a set of rules that define the criteria by which it should select scheduled payments for payment creation.
payment shuffle	In PeopleSoft Enterprise Campus Solutions, a process allowing payments that have been previously posted to a student's account to be automatically reapplied when a higher priority payment is posted or the payment allocation definition is changed.
pending item	In PeopleSoft Receivables, an individual receivable (such as an invoice, a credit memo, or a write-off) that has been entered in or created by the system, but hasn't been posted.
PeopleCode	PeopleCode is a proprietary language, executed by the PeopleSoft application processor. PeopleCode generates results based upon existing data or user actions. By using business interlink objects, external services are available to all PeopleSoft applications wherever PeopleCode can be executed.
PeopleCode event	An action that a user takes upon an object, usually a record field, that is referenced within a PeopleSoft page.
PeopleSoft Internet Architecture	The fundamental architecture on which PeopleSoft 8 applications are constructed, consisting of a relational database management system (RDBMS), an application server, a web server, and a browser.
performance measurement	In PeopleSoft Enterprise Incentive Management, a variable used to store data (similar to an aggregator, but without a predefined formula) within the scope of an incentive plan. Performance measures are associated with a plan calendar, territory, and participant. Performance measurements are used for quota calculation and reporting.
period context	In PeopleSoft Enterprise Incentive Management, because a participant typically uses the same compensation plan for multiple periods, the period context associates a plan context with a specific calendar period and fiscal year. The period context references the associated plan context, thus forming a chain. Each plan context has a corresponding set of period contexts.
person of interest	A person about whom the organization maintains information but who is not part of the workforce.

personal portfolio	In PeopleSoft Enterprise Campus Solutions, the user-accessible menu item that contains an individual's name, address, telephone number, and other personal information.
plan	In PeopleSoft Sales Incentive Management, a collection of allocation rules, variables, steps, sections, and incentive rules that instruct the PeopleSoft Enterprise Incentive Management engine in how to process transactions.
plan context	In PeopleSoft Enterprise Incentive Management, correlates a participant with the compensation plan and node to which the participant is assigned, enabling the PeopleSoft Enterprise Incentive Management system to find anything that is associated with the node and that is required to perform compensation processing. Each participant, node, and plan combination represents a unique plan context—if three participants are on a compensation structure, each has a different plan context. Configuration plans are identified by plan contexts and are associated with the participants that refer to them.
plan template	In PeopleSoft Enterprise Incentive Management, the base from which a plan is created. A plan template contains common sections and variables that are inherited by all plans that are created from the template. A template may contain steps and sections that are not visible in the plan definition.
planned learning	In PeopleSoft Enterprise Learning Management, a self-service repository for all of a learner's planned learning activities and programs.
planning instance	In PeopleSoft Supply Planning, a set of data (business units, items, supplies, and demands) constituting the inputs and outputs of a supply plan.
population	In PeopleSoft Enterprise Campus Solutions, the middle level of the three-level classification structure that you define in PeopleSoft Enterprise Recruiting and Admissions for enrollment management. You can define a population level, link it to other levels, and set enrollment target numbers for it. See also <i>division</i> and <i>cohort</i> .
portal registry	In PeopleSoft applications, the portal registry is a tree-like structure in which content references are organized, classified, and registered. It is a central repository that defines both the structure and content of a portal through a hierarchical, tree-like structure of folders useful for organizing and securing content references.
price list	In PeopleSoft Enterprise Pricer, enables you to select products and conditions for which the price list applies to a transaction. During a transaction, the system either determines the product price based on the predefined search hierarchy for the transaction or uses the product's lowest price on any associated, active price lists. This price is used as the basis for any further discounts and surcharges.
price rule	In PeopleSoft Enterprise Pricer, defines the conditions that must be met for adjustments to be applied to the base price. Multiple rules can apply when conditions of each rule are met.
price rule condition	In PeopleSoft Enterprise Pricer, selects the price-by fields, the values for the price-by fields, and the operator that determines how the price-by fields are related to the transaction.
price rule key	In PeopleSoft Enterprise Pricer, defines the fields that are available to define price rule conditions (which are used to match a transaction) on the price rule.
primacy number	In PeopleSoft Enterprise Campus Solutions, a number that the system uses to prioritize financial aid applications when students are enrolled in multiple academic careers and academic programs at the same time. The Consolidate Academic Statistics process uses the primacy number indicated for both the career and program at the institutional level to determine a student's primary career and program. The system also uses the

	number to determine the primary student attribute value that is used when you extract data to report on cohorts. The lowest number takes precedence.
primary name type	In PeopleSoft Enterprise Campus Solutions, the name type that is used to link the name stored at the highest level within the system to the lower-level set of names that an individual provides.
process category	In PeopleSoft Process Scheduler, processes that are grouped for server load balancing and prioritization.
process group	In PeopleSoft Financials, a group of application processes (performed in a defined order) that users can initiate in real time, directly from a transaction entry page.
process definition	Process definitions define each run request.
process instance	A unique number that identifies each process request. This value is automatically incremented and assigned to each requested process when the process is submitted to run.
process job	You can link process definitions into a job request and process each request serially or in parallel. You can also initiate subsequent processes based on the return code from each prior request.
process request	A single run request, such as a Structured Query Report (SQR), a COBOL or Application Engine program, or a Crystal report that you run through PeopleSoft Process Scheduler.
process run control	A PeopleTools variable used to retain PeopleSoft Process Scheduler values needed at runtime for all requests that reference a run control ID. Do not confuse these with application run controls, which may be defined with the same run control ID, but only contain information specific to a given application process request.
product category	In PeopleSoft Enterprise Incentive Management, indicates an application in the Enterprise Incentive Management suite of products. Each transaction in the PeopleSoft Enterprise Incentive Management system is associated with a product category.
programs	In PeopleSoft Enterprise Learning Management, a high-level grouping that guides the learner along a specific learning path through sections of catalog items. PeopleSoft Enterprise Learning Systems provides two types of programs—curricula and certifications.
progress log	In PeopleSoft Services Procurement, tracks deliverable-based projects. This is similar to the time sheet in function and process. The service provider contact uses the progress log to record and submit progress on deliverables. The progress can be logged by the activity that is performed, by the percentage of work that is completed, or by the completion of milestone activities that are defined for the project.
project transaction	In PeopleSoft Project Costing, an individual transaction line that represents a cost, time, budget, or other transaction row.
promotion	In PeopleSoft Promotions Management, a trade promotion, which is typically funded from trade dollars and used by consumer products manufacturers to increase sales volume.
prospects	In PeopleSoft Enterprise Campus Solutions, students who are interested in applying to the institution. In PeopleSoft Enterprise Contributor Relations, individuals and organizations that are most likely to make substantial financial commitments or other types of commitments to the institution.
publishing	In PeopleSoft Enterprise Incentive Management, a stage in processing that makes incentive-related results available to participants.

rating components	In PeopleSoft Enterprise Campus Solutions, variables used with the Equation Editor to retrieve specified populations.
record group	A set of logically and functionally related control tables and views. Record groups help enable TableSet sharing, which eliminates redundant data entry. Record groups ensure that TableSet sharing is applied consistently across all related tables and views.
record input VAT flag	Abbreviation for <i>record input value-added tax flag</i> . Within PeopleSoft Purchasing, Payables, and General Ledger, this flag indicates that you are recording input VAT on the transaction. This flag, in conjunction with the record output VAT flag, is used to determine the accounting entries created for a transaction and to determine how a transaction is reported on the VAT return. For all cases within Purchasing and Payables where VAT information is tracked on a transaction, this flag is set to Yes. This flag is not used in PeopleSoft Order Management, Billing, or Receivables, where it is assumed that you are always recording only output VAT, or in PeopleSoft Expenses, where it is assumed that you are always recording only input VAT.
record output VAT flag	Abbreviation for <i>record output value-added tax flag</i> . See <i>record input VAT flag</i> .
recname	The name of a record that is used to determine the associated field to match a value or set of values.
recognition	In PeopleSoft Enterprise Campus Solutions, the recognition type indicates whether the PeopleSoft Enterprise Contributor Relations donor is the primary donor of a commitment or shares the credit for a donation. Primary donors receive hard credit that must total 100 percent. Donors that share the credit are given soft credit. Institutions can also define other share recognition-type values such as memo credit or vehicle credit.
reference data	In PeopleSoft Sales Incentive Management, system objects that represent the sales organization, such as territories, participants, products, customers, channels, and so on.
reference object	In PeopleSoft Enterprise Incentive Management, this dimension-type object further defines the business. Reference objects can have their own hierarchy (for example, product tree, customer tree, industry tree, and geography tree).
reference transaction	In commitment control, a reference transaction is a source transaction that is referenced by a higher-level (and usually later) source transaction, in order to automatically reverse all or part of the referenced transaction's budget-checked amount. This avoids duplicate postings during the sequential entry of the transaction at different commitment levels. For example, the amount of an encumbrance transaction (such as a purchase order) will, when checked and recorded against a budget, cause the system to concurrently reference and relieve all or part of the amount of a corresponding pre-encumbrance transaction, such as a purchase requisition.
regional sourcing	In PeopleSoft Purchasing, provides the infrastructure to maintain, display, and select an appropriate vendor and vendor pricing structure that is based on a regional sourcing model where the multiple ship to locations are grouped. Sourcing may occur at a level higher than the ship to location.
relationship object	In PeopleSoft Enterprise Incentive Management, these objects further define a compensation structure to resolve transactions by establishing associations between compensation objects and business objects.
remote data source data	Data that is extracted from a separate database and migrated into the local database.
REN server	Abbreviation for <i>real-time event notification server</i> in PeopleSoft MultiChannel Framework.
requester	In PeopleSoft eSettlements, an individual who requests goods or services and whose ID appears on the various procurement pages that reference purchase orders.

reversal indicator	In PeopleSoft Enterprise Campus Solutions, an indicator that denotes when a particular payment has been reversed, usually because of insufficient funds.
role	Describes how people fit into PeopleSoft Workflow. A role is a class of users who perform the same type of work, such as clerks or managers. Your business rules typically specify what user role needs to do an activity.
role user	A PeopleSoft Workflow user. A person's role user ID serves much the same purpose as a user ID does in other parts of the system. PeopleSoft Workflow uses role user IDs to determine how to route worklist items to users (through an email address, for example) and to track the roles that users play in the workflow. Role users do not need PeopleSoft user IDs.
roll up	In a tree, to roll up is to total sums based on the information hierarchy.
run control	A run control is a type of online page that is used to begin a process, such as the batch processing of a payroll run. Run control pages generally start a program that manipulates data.
run control ID	A unique ID to associate each user with his or her own run control table entries.
run-level context	In PeopleSoft Enterprise Incentive Management, associates a particular run (and batch ID) with a period context and plan context. Every plan context that participates in a run has a separate run-level context. Because a run cannot span periods, only one run-level context is associated with each plan context.
search query	You use this set of objects to pass a query string and operators to the search engine. The search index returns a set of matching results with keys to the source documents.
search/match	In PeopleSoft Enterprise Campus Solutions and PeopleSoft Enterprise Human Resources Management Solutions, a feature that enables you to search for and identify duplicate records in the database.
seasonal address	In PeopleSoft Enterprise Campus Solutions, an address that recurs for the same length of time at the same time of year each year until adjusted or deleted.
section	In PeopleSoft Enterprise Incentive Management, a collection of incentive rules that operate on transactions of a specific type. Sections enable plans to be segmented to process logical events in different sections.
security event	In commitment control, security events trigger security authorization checking, such as budget entries, transfers, and adjustments; exception overrides and notifications; and inquiries.
serial genealogy	In PeopleSoft Manufacturing, the ability to track the composition of a specific, serial-controlled item.
serial in production	In PeopleSoft Manufacturing, enables the tracing of serial information for manufactured items. This is maintained in the Item Master record.
service impact	In PeopleSoft Enterprise Campus Solutions, the resulting action triggered by a service indicator. For example, a service indicator that reflects nonpayment of account balances by a student might result in a service impact that prohibits registration for classes.
service indicator	In PeopleSoft Enterprise Campus Solutions, indicates services that may be either withheld or provided to an individual. Negative service indicators indicate holds that prevent the individual from receiving specified services, such as check-cashing privileges or registration for classes. Positive service indicators designate special services that are provided to the individual, such as front-of-line service or special services for disabled students.

session	<p>In PeopleSoft Enterprise Campus Solutions, time elements that subdivide a term into multiple time periods during which classes are offered. In PeopleSoft Contributor Relations, a session is the means of validating gift, pledge, membership, or adjustment data entry. It controls access to the data entered by a specific user ID. Sessions are balanced, queued, and then posted to the institution's financial system. Sessions must be posted to enter a matching gift or pledge payment, to make an adjustment, or to process giving clubs or acknowledgements.</p> <p>In PeopleSoft Enterprise Learning Management, a single meeting day of an activity (that is, the period of time between start and finish times within a day). The session stores the specific date, location, meeting time, and instructor. Sessions are used for scheduled training.</p>
session template	In PeopleSoft Enterprise Learning Management, enables you to set up common activity characteristics that may be reused while scheduling a PeopleSoft Enterprise Learning Management activity—characteristics such as days of the week, start and end times, facility and room assignments, instructors, and equipment. A session pattern template can be attached to an activity that is being scheduled. Attaching a template to an activity causes all of the default template information to populate the activity session pattern.
setup relationship	In PeopleSoft Enterprise Incentive Management, a relationship object type that associates a configuration plan with any structure node.
share driver expression	In PeopleSoft Business Planning, a named planning method similar to a driver expression, but which you can set up globally for shared use within a single planning application or to be shared between multiple planning applications through PeopleSoft Enterprise Warehouse.
single signon	With single signon, users can, after being authenticated by a PeopleSoft application server, access a second PeopleSoft application server without entering a user ID or password.
source key process	In PeopleSoft Enterprise Campus Solutions, a process that relates a particular transaction to the source of the charge or financial aid. On selected pages, you can drill down into particular charges.
source transaction	In commitment control, any transaction generated in a PeopleSoft or third-party application that is integrated with commitment control and which can be checked against commitment control budgets. For example, a pre-encumbrance, encumbrance, expenditure, recognized revenue, or collected revenue transaction.
speed key	See <i>communication key</i> .
SpeedChart	A user-defined shorthand key that designates several ChartKeys to be used for voucher entry. Percentages can optionally be related to each ChartKey in a SpeedChart definition.
SpeedType	A code representing a combination of ChartField values. SpeedTypes simplify the entry of ChartFields commonly used together.
staging	A method of consolidating selected partner offerings with the offerings from the enterprise's other partners.
standard letter code	In PeopleSoft Enterprise Campus Solutions, a standard letter code used to identify each letter template available for use in mail merge functions. Every letter generated in the system must have a standard letter code identification.
statutory account	Account required by a regulatory authority for recording and reporting financial results. In PeopleSoft, this is equivalent to the Alternate Account (ALTACCT) ChartField.

step	In PeopleSoft Sales Incentive Management, a collection of sections in a plan. Each step corresponds to a step in the job run.
storage level	In PeopleSoft Inventory, identifies the level of a material storage location. Material storage locations are made up of a business unit, a storage area, and a storage level. You can set up to four storage levels.
subcustomer qualifier	A value that groups customers into a division for which you can generate detailed history, aging, events, and profiles.
Summary ChartField	You use summary ChartFields to create summary ledgers that roll up detail amounts based on specific detail values or on selected tree nodes. When detail values are summarized using tree nodes, summary ChartFields must be used in the summary ledger data record to accommodate the maximum length of a node name (20 characters).
summary ledger	An accounting feature used primarily in allocations, inquiries, and PS/nVision reporting to store combined account balances from detail ledgers. Summary ledgers increase speed and efficiency of reporting by eliminating the need to summarize detail ledger balances each time a report is requested. Instead, detail balances are summarized in a background process according to user-specified criteria and stored on summary ledgers. The summary ledgers are then accessed directly for reporting.
summary time period	In PeopleSoft Business Planning, any time period (other than a base time period) that is an aggregate of other time periods, including other summary time periods and base time periods, such as quarter and year total.
summary tree	A tree used to roll up accounts for each type of report in summary ledgers. Summary trees enable you to define trees on trees. In a summary tree, the detail values are really nodes on a detail tree or another summary tree (known as the <i>basis</i> tree). A summary tree structure specifies the details on which the summary trees are to be built.
syndicate	To distribute a production version of the enterprise catalog to partners.
system function	In PeopleSoft Receivables, an activity that defines how the system generates accounting entries for the general ledger.
TableSet	A means of sharing similar sets of values in control tables, where the actual data values are different but the structure of the tables is the same.
TableSet sharing	Shared data that is stored in many tables that are based on the same TableSets. Tables that use TableSet sharing contain the SETID field as an additional key or unique identifier.
target currency	The value of the entry currency or currencies converted to a single currency for budget viewing and inquiry purposes.
tax authority	In PeopleSoft Enterprise Campus Solutions, a user-defined element that combines a description and percentage of a tax with an account type, an item type, and a service impact.
template	A template is HTML code associated with a web page. It defines the layout of the page and also where to get HTML for each part of the page. In PeopleSoft, you use templates to build a page by combining HTML from a number of sources. For a PeopleSoft portal, all templates must be registered in the portal registry, and each content reference must be assigned a template.
territory	In PeopleSoft Sales Incentive Management, hierarchical relationships of business objects, including regions, products, customers, industries, and participants.
3C engine	Abbreviation for <i>Communications, Checklists, and Comments engine</i> . In PeopleSoft Enterprise Campus Solutions, the 3C engine enables you to automate business processes that involve additions, deletions, and updates to communications, checklists,

and comments. You define events and triggers to engage the engine, which runs the mass change and processes the 3C records (for individuals or organizations) immediately and automatically from within business processes.

3C group	Abbreviation for <i>Communications, Checklists, and Comments group</i> . In PeopleSoft Enterprise Campus Solutions, a method of assigning or restricting access privileges. A 3C group enables you to group specific communication categories, checklist codes, and comment categories. You can then assign the group inquiry-only access or update access, as appropriate.
TimeSpan	A relative period, such as year-to-date or current period, that can be used in various PeopleSoft General Ledger functions and reports when a rolling time frame, rather than a specific date, is required. TimeSpans can also be used with flexible formulas in PeopleSoft Projects.
trace usage	In PeopleSoft Manufacturing, enables the control of which components will be traced during the manufacturing process. Serial- and lot-controlled components can be traced. This is maintained in the Item Master record.
transaction allocation	In PeopleSoft Enterprise Incentive Management, the process of identifying the owner of a transaction. When a raw transaction from a batch is allocated to a plan context, the transaction is duplicated in the PeopleSoft Enterprise Incentive Management transaction tables.
transaction state	In PeopleSoft Enterprise Incentive Management, a value assigned by an incentive rule to a transaction. Transaction states enable sections to process only transactions that are at a specific stage in system processing. After being successfully processed, transactions may be promoted to the next transaction state and “picked up” by a different section for further processing.
Translate table	A system edit table that stores codes and translate values for the miscellaneous fields in the database that do not warrant individual edit tables of their own.
tree	The graphical hierarchy in PeopleSoft systems that displays the relationship between all accounting units (for example, corporate divisions, projects, reporting groups, account numbers) and determines roll-up hierarchies.
tuition lock	In PeopleSoft Enterprise Campus Solutions, a feature in the Tuition Calculation process that enables you to specify a point in a term after which students are charged a minimum (or <i>locked</i>) fee amount. Students are charged the locked fee amount even if they later drop classes and take less than the normal load level for that tuition charge.
unclaimed transaction	In PeopleSoft Enterprise Incentive Management, a transaction that is not claimed by a node or participant after the allocation process has completed, usually due to missing or incomplete data. Unclaimed transactions may be manually assigned to the appropriate node or participant by a compensation administrator.
universal navigation header	Every PeopleSoft portal includes the universal navigation header, intended to appear at the top of every page as long as the user is signed on to the portal. In addition to providing access to the standard navigation buttons (like Home, Favorites, and signoff) the universal navigation header can also display a welcome message for each user.
update access	In PeopleSoft Enterprise Campus Solutions, a type of security access that permits the user to edit and update data. See also <i>inquiry access</i> .
user interaction object	In PeopleSoft Sales Incentive Management, used to define the reporting components and reports that a participant can access in his or her context. All Sales Incentive Management user interface objects and reports are registered as user interaction objects. User interaction objects can be linked to a compensation structure node through a compensation relationship object (individually or as groups).

variable	In PeopleSoft Sales Incentive Management, the intermediate results of calculations. Variables hold the calculation results and are then inputs to other calculations. Variables can be plan variables that persist beyond the run of an engine or local variables that exist only during the processing of a section.
VAT exception	Abbreviation for <i>value-added tax exception</i> . A temporary or permanent exemption from paying VAT that is granted to an organization. This terms refers to both VAT exoneration and VAT suspension.
VAT exempt	Abbreviation for <i>value-added tax exempt</i> . Describes goods and services that are not subject to VAT. Organizations that supply exempt goods or services are unable to recover the related input VAT. This is also referred to as exempt without recovery.
VAT exoneration	Abbreviation for <i>value-added tax exoneration</i> . An organization that has been granted a permanent exemption from paying VAT due to the nature of that organization.
VAT suspension	Abbreviation for <i>value-added tax suspension</i> . An organization that has been granted a temporary exemption from paying VAT.
warehouse	A PeopleSoft data warehouse that consists of predefined ETL maps, data warehouse tools, and DataMart definitions.
work order	In PeopleSoft Services Procurement, enables an enterprise to create resource-based and deliverable-based transactions that specify the basic terms and conditions for hiring a specific service provider. When a service provider is hired, the service provider logs time or progress against the work order.
worker	A person who is part of the workforce; an employee or a contingent worker.
workset	A group of people and organizations that are linked together as a set. You can use worksets to simultaneously retrieve the data for a group of people and organizations and work with the information on a single page.
worksheet	A way of presenting data through a PeopleSoft Business Analysis Modeler interface that enables users to do in-depth analysis using pivoting tables, charts, notes, and history information.
worklist	The automated to-do list that PeopleSoft Workflow creates. From the worklist, you can directly access the pages you need to perform the next action, and then return to the worklist for another item.
XML schema	An XML definition that standardizes the representation of application messages, component interfaces, or business interlinks.
yield by operation	In PeopleSoft Manufacturing, the ability to plan the loss of a manufactured item on an operation-by-operation basis.
zero-rated VAT	Abbreviation for <i>zero-rated value-added tax</i> . A VAT transaction with a VAT code that has a tax percent of zero. Used to track taxable VAT activity where no actual VAT amount is charged. Organizations that supply zero-rated goods and services can still recover the related input VAT. This is also referred to as exempt with recovery.

Index

A

- Access Media Object system function 186
- Activate Item system function 186
- Add Action system function 202
- Add Activity Button Clicked event 107
- Add Calendar Activity system function 107, 108, 109
- Add Item system function 122
- Add Last Entry Row to Grid event 147
- Add mode
 - designating a default cursor field 21
 - preventing form clear 24
- additional documentation xx
- Allow Image Items control property 19
- Allow in Saved Query control property 232
- Allow OLE Items control property 19
- Allow RTF Text control property 19
- Allow Text Items control property 19
- Allowed in Saved Query control property 19, 232
- Alternate Grid Format control property 144
- Alternate Grid Row Format String control property 19
- Always Hidden control property 19
- application fundamentals xix
- Application Tree View 11
- Attach Path To Segment system function 203
- Automatic Scroll Horizontal control property 19
- Automatic Scroll Vertical control property 19
- Automatically Find on Entry control property 20, 68

B

- business view
 - filtering on 231
 - querying on 231
- Business View Columns Browser 12
- Business View Name control property 20
- Business View Name form property 5
- Button Type control property 20

C

- calendar controls
 - adding activities 106, 107, 108, 109, 110
 - deleting activities 108, 111
 - hiding and showing views 106, 109
 - modifying activities 108, 112, 113
 - refreshing 108
 - selecting views 114
- Calendar Day View Visible control property 20
- Calendar Month View Visible control property 20
- Calendar Week View Visible control property 20
- Call Function system function 242, 243
- Change Row Selection system function 162, 203
- Checked Value control property 20, 115
- Clear Characterization Cache system function 187
- Clear Grid Buffer grid control system function 162, 163
- Clear Grid Buffer parent child control system function 204
- Clear Grid Cell Error grid control system function 162, 163
- Clear Grid Cell Error parent child control system function 204
- Clear QBE Column grid control system function 161, 163
- Clear QBE Column parent child control system function 204
- Clear Selection system function 164
- Clear Sequencing system function 162, 164
- Clickable control property 20, 233
- Client Edge control property 20
- Collapsible control property 20, 236
- Collapse All control property 195
- Collapse Subform system function 247
- colors, changing for grids 160, 173
- Column Header One control property 20
- Column Header Two control property 21

- Column Moved to Tree control
 - property 21
- Column Order control property 21
- Column Selection Changed event 148
- Column Sort Order control property 21
- combo boxes
 - entering values with no
 - descriptions 118
 - requiring user entries 119
 - translating descriptions 118
- comments, submitting xxiv
- common elements xxiv
- contact information xxiv
- Contact Tree Node system function 205
- Continue Custom Data Fetch system
 - function 147
- Continue Custom Fetch system
 - function 153
- Control ID control property 21
- control properties 18
- controls
 - disabling 23
 - hiding and showing 19, 30, 38
 - positioning on forms 28, 35
 - renaming 33
- Copy Grid Row to Grid Buffer grid control
 - system function 162, 164
- Copy Grid Row To Grid Buffer parent child
 - control system function 205
- cross-references xxiii
- Customer Connection website xx

D

- Data Dictionary Browser 11
- data dictionary overrides 41
- data dictionary overrides, setting for a
 - control 30, 172
- Data Item Information control property 21
- Data Structure control property 21
- Data Structure form property 5
- database commits 9
- Default cursor on add mode control
 - property 21
- Default cursor on update mode control
 - property 21
- Default Pushbutton control property 227
- Delete All Actions system function 205
- Delete All Tree Nodes system
 - function 206

- Delete Calendar Activity system
 - function 106, 108, 109, 111
- Delete Grid Row grid control system
 - function 162, 165
- Delete Grid Row parent child control
 - system function 206
- Delete Item system function 187
- dialog close 5
- Disable Characterization Cache system
 - function 187
- Disable Copy control property 21
- Disable Drag and Drop (Cut/Copy/Paste)
 - control property 22
- Disable Grid grid control system
 - function 161, 165
- Disable Grid parent child control system
 - function 206
- Disable Move (Cut) control property 22
- Disable Page-at-a-Time Process control
 - property 22
- Disable QBE control property 22
- Disable Subform system function 244
- Disabled control property 23
- Display Customize Grid Option system
 - function 161
- Display Customized Grid control
 - property 23
- Display Customized Grid Option system
 - function 166
- Display Export to Excel control
 - property 23
- Display Export to Excel Option system
 - function 161, 166
- Display Export to Word control
 - property 23
- Display Export to Word Option system
 - function 161, 166
- Display Import from Excel control
 - property 23
- Display Import from Excel Option system
 - function 161, 167
- Display Style control property 23
- Do Not Clear After Add control
 - property 24
- documentation
 - printed xx
 - related xx
 - updates xx
- Double Click on Row Header event 148
- Drill Into Calendar Activity event 107

Drill Into Time Span event 107

E

edit controls
 filtering on 137
 hiding password entries 30
 showing its data dictionary title on the form 41
 showing multiple lines 28
 type ahead feature in 137
 Editable control property 24
 Enable Grid grid control system
 function 161, 167
 Enable Grid parent child control system
 function 207
 Enable In-Your-Face-Error Display form
 property 5
 Enable Progress List control property 271
 Enable Re-entry Save control
 property 271
 Enable Subform system function 243
 End Form on Add form property 5
 Entry Point form property 6
 error messages 5
 Expand All control property 195
 Expand All/Collapse All control
 property 24
 Expand Subform system function 246
 Expand Tree Node system function 207
 exporting data 161

F

FDA best practices 11
 Fetch on Business View form property 47, 51
 Fetch on Form Business View form
 property 6
 Fetch on Form Businessview control
 property 24, 65, 68
 Fetch on Grid Business View form
 property 6
 Fetch on Grid Businessview control
 property 65, 68, 71
 fields 6, 24, 65
 File Name control property 24
 Filter Criteria - Checked control
 property 24
 Filter Criteria - Unchecked control
 property 25

Filter Criteria control property 24, 229
 find/browse forms, establishing modeless
 interconnections 4
 Flat control property 25
 fonts, changing for grids 160, 174
 form backgrounds 8
 form close, *See* dialog close
 form flow
 closing forms automatically 5
 showing a form on application
 initialization 6
 Form Guide form property 7
 form interconnections 4, 12, 13
 Form Name control property 25
 Form Name form property 7
 form names 8
 form properties 4
 Form Type form property 7
 forms, closing, *See* dialog close
 Full File Name control property 25

G

Get Current Wizard Page ID system
 function 285
 Get Custom Grid Row event 153
 Get Custom Grid Row system
 function 147
 Get Error Count system function 245
 Get Grid Row grid control system
 function 162, 167
 Get Grid Row parent child control system
 function 208
 Get Max Grid Rows grid control system
 function 161, 168
 Get Max Grid Rows parent child control
 system function 208
 Get Next Selected Row system
 function 168, 208
 Get Node ID system function 201, 209
 Get Node Level system function 210
 Get OLE Item system function 188
 Get Related Node ID system function 201, 202, 210
 Get Row Number system function 201, 211
 Get Selected Context Action system
 function 211
 Get Selected Grid Row Count grid control
 system function 161, 169

- Get Selected Grid Row Number grid control system function 162, 169
- Get Selected Grid Row Number system function 212
- Get Subform ID 245
- Get Tree Node Handle system function 213
- Get Warning Count system function 245
- Get Wizard Page Index system function 285
- glossary 311
- grid buffers
 - clearing 162, 163, 204
 - inserting rows 162, 171
 - populating from grid 162, 164
 - updating rows 162, 180
- Grid Column Clicked event 144, 148
- grid columns
 - arranging 145
 - changing the heading 144, 161, 172, 173
 - changing the sort order 145
 - clearing and setting errors 162, 163
 - disabling and enabling 146, 161, 165, 167
 - hiding and setting errors 172
 - hiding and showing 144, 161, 170, 179
 - making clickable 144
 - wrapping text 145
- grid controls
 - changing the appearance of 160, 173, 174, 175
 - clearing and setting errors 162, 163
 - customizing 144, 161, 166
 - disabling and enabling 161, 165, 167
 - fetching data automatically 6, 20, 65
 - hiding and setting errors 172
 - hiding and showing
 - HTML row selector 26
 - QBE line 27
 - loading 145, 146, 162
 - manipulating the grid buffer 162
- Grid Row Count control property 25
- grid rows
 - clearing and setting errors 162, 163
 - counting 161, 168, 169
 - deleting 162, 165
 - disabling and enabling 161, 165, 167
 - hiding and setting errors 172
 - hiding and showing 144, 161, 170, 180

- write to grid buffer (GC) 162, 164

H

- Height control property 26
- Height form property 7
- Hide Grid Column grid control system function 161, 170
- Hide Grid Column parent child control system function 213
- Hide Grid Row system function 161, 170
- Hide HTML Row Selector control property 26
- Hide in Grid control property 26
- Hide Query By Example control property 27
- Hide Subform system function 244
- Hide the Viewer Icon Panel system function 190

I

- importing data 161
- Indent and Outdent control property 27
- Insert Controls tool bar 11
- Insert Grid Buffer Row By Node ID system function 202, 214
- Insert Grid Buffer Row system function 162, 171, 194, 213
- Insert OLE Object 188
- Insert Text system function 189
- Insert URL system function 190

J

- Justification control property 27

K

- Key Relations control property 27
- Kill Focus on Grid event 148

L

- Layout tool bar 11
- Left control property 28
- Lines control property 28
- Load Calendar Activity event 106, 107
- Load Calendar Activity system function 107, 108
- Load from Cache system function 123, 124
- Load Text by Instance control property 28

Location Indicator Feature control
property 29, 195

M

Main Toolbar 11
Maintain Aspect Ratio control
property 29
Mapping Links control property 29, 86,
91, 236
Mapping Links form property 8
Menubar Separator control property 29
message forms 20
MMA Partners xx
modal form interconnections 4, 12
Modal Frame control property 29
modeless form interconnections 4, 13
Modify Calendar Activity system
function 108, 112
Move Up and Down control property 29
Multi-Line Edit control property 29
Multiple Select control property 29, 195

N

New Text Item on Open control
property 29
No Adds On Update Grid control
property 29
No display if currency is OFF control
property 30
node ID column 202
Node ID Column control property 30
Node Indent Verify Before event 198, 201
Node is Moved Down event 198
Node is Moved Up event 198
Node Move Down Verify Before
event 198, 201
Node Move Up Verify Before event 198,
201
Node Outdent Verify Before event 198,
201
notes xxiii
Notified by Child event 237
Notified by Parent event 237
Notify Child system function 246
Notify Parent system function 244
Number of columns joined to header control
property 30

O

Overrides Button control property 30

P

Page is Exited - After event 273
Page is Exited - Before event 273
page-at-a-time processing 22, 145, 146,
153, 193
parent child control
adding an entry row 202
preventing user changes to the tree
structure 201
parent child controls, clearing and setting
errors 204
Parent control property 30
Password control property 30
PeopleBooks
ordering xx
PeopleCode, typographical
conventions xxii
PeopleSoft application fundamentals xix
Personalization Applied event 48, 52
Personalization mode 48, 52
Portlet is Exited event 48, 52
Portlet is Initialized event 48, 52
Position of Saved Query links control
property 30, 232
Post Add Activity Button Clicked
event 107
Post Button Clicked - Asynch event 227
Post Visual Assistant Clicked event 148
Post Wizard is Initialized event 272
prerequisites xix
Prevent Resizing control property 30
printed documentation xx
Process All Rows in Grid control
property 30
Product Synch Mapping control
property 31
Product Synch Mode control property 31
Progress Indicator control property 31,
270
progress indicators 269, 270
properties, *See* control properties, form
properties
Property Browser 11

Q

QBE, *See* query-by-example (QBE)

query-by-example (QBE)
 clearing 161, 163
 configuring for a grid 141, 177
 disabling 22, 144
 hiding 27, 144

R

radio buttons 42
 Re-entry Save control property 31
 Read Only control property 31, 137
 Reclaim Whitespace control property 31
 related documentation xx
 Required control property 31
 Required entry field control property 31
 Reusable control property 31
 Row Exit & Changed - Inline event 148
 Row is Exited & Changed - Asynch event 148
 Row is Exited event 148
 Row is Selected (Web Only) event 148

S

Save Data control property 32
 scrolling 19
 Select Calendar View system
 function 107, 109, 114
 --Select One-- prompt 117
 Selection Changed event 115, 120, 229
 Set Action system function 215
 Set Add Button Text system function 109
 Set Add Button Visible system
 function 109
 Set Characterization Cache system
 function 191
 Set Control Text system function 233
 Set Cursor Position system function 191
 Set Data Dictionary Item Overrides system
 function 172
 Set Data Dictionary Item system
 function 161, 172, 216
 Set Data Dictionary Overrides system
 function 161, 216
 Set Day Type system function 109
 Set Drag Cursor system function 217
 Set Focus on Grid event 148
 Set Grid Cell Error system function 162,
 172, 217
 Set Grid Color system function 160, 173

Set Grid Column Heading system
 function 161, 173, 218
 Set Grid Font system function 160, 174,
 218
 Set Grid Row Bitmap system
 function 161, 174
 Set Grid Row Format system function 19,
 160, 175
 Set Grid Text Indicator system
 function 192
 Set Lower Limit system function 175
 Set QBE Column Compare Style system
 function 161, 177, 219
 Set Root Node ID system function 202
 Set Selected Wizard Page system
 function 272, 286
 Set Selection Append Flag system
 function 162, 178
 Set Selection system function 161, 162,
 177
 Set Sequencing system function 162, 179
 Set Text Color system function 192
 Set Tree Bitmap Scheme system
 function 220
 Set Tree Node Bitmap system
 function 194, 202, 220
 Set Tree Node Bold system function 221
 Set Tree Node Clickable Bitmap system
 function 199, 201, 221
 Set Tree Node Handle system
 function 222
 Set Tree Root Node ID system
 function 222
 Set View Visible system function 106,
 107, 109
 Set Wizard Form Mode system
 function 272, 273
 Set Wizard Page Index system
 function 272
 Set Wizard Page Status system
 function 270, 273, 287
 Set Work Day Hours system function 109
 Set Work Week system function 109
 Show details of all tree nodes control
 property 32
 Show Grid Column system function 161,
 179, 223
 Show Grid Row system function 161, 180
 Show Header control property 32, 236
 Show N Level system function 223

Show Subform system function 244
 SL_CalendarActivityID system variable 107
 SL_EndTime system variable 106, 107, 108
 SL_StartTime system variable 106, 107, 108
 Sort Direction control property 32
 Sortable by End User control property 32
 Sorted control property 32
 Static Edge control property 32
 Status Bar 11
 Subform Application Name control property 32
 suggestions, submitting xxiv
 Support Multiple Currencies control property 32
 Suppress Fetch on Node Expand system function 194, 223
 Suppress Grid Line system function 162, 180, 224
 Suppress Node Indent/Outdent system function 198, 201, 224
 Suppress Node Move Up/Down system function 198, 201, 224
 Suppress Validation and Save control property 32, 270, 271
 Suppress Wizard Page Validation and Save system function 270, 288

T

Tab Sequence Toolbar 12
 Tab Stop control property 33
 tables 9
 Task List control property 33
 terms 311
 Text Clicked event 20, 233
 Text is Overridden control property 33
 Tile Wallpaper form property 8
 Title control property 34
 Title form property 8
 Tool Tip control property 34
 Toolbar control property 34
 Toolbar Separator control property 35
 Top control property 35
 Total Controls on a Form form property 8
 Transaction control property 36, 86, 92, 237
 Transaction form property 9
 Tree - Begin Drag/Cut/Copy event 197

Tree - Cancel Drag Drop/Paste event 198
 Tree - Drag Over Node event 197
 Tree - End Drag Drop/Paste event 197
 Tree Node Bitmap is Clicked event 199, 201
 Tree Node Is Collapsing event 199
 Tree Node Is Expanding event 194, 199
 Tree Node is Indented event 198
 Tree Node is Outdented event 198
 Tree Node Selection Changed event 197
 Trigger Default Action system function 246
 Type Ahead feature in edit controls 137
 typographical conventions xxii

U

Unchecked Value control property 36, 115
 Update Grid Buffer Row system function 162, 180, 225
 Update Mapping Link control property 36
 Update Mode control property 36
 Update mode, designating a default cursor field 21
 Update on Business View form property 47, 51
 Update on Form Business View form property 9
 Update on Form Businessview control property 36
 Update on Grid Business View form property 9
 Update on Grid Businessview control property 65, 71
 Update Parent system function 244, 273, 275
 updates, *See* database commits
 Use Alternate Grid Row Format control property 36
 UTC Display Format control property 37

V

Value control property 37, 229
 Visible control property 38
 Visual Assist Button Clicked event 148
 visual cues xxiii

W

Wallpaper File form property 10

- Wallpaper form property 9
- Wallpaper Full File Name form property 10
- warnings xxiii
- Was Grid Cell Value Entered system function 161, 181, 225
- Width control property 39
- Width form property 7
- Wildcard control property 39
- wizard controls
 - assigning SI values 272, 274, 275
 - changing form mode 272, 273
 - changing page condition 272
 - changing page status 273
 - incorporating confirmation forms 285
 - reordering pages 272
 - setting filter and QBE values 272
 - setting transaction boundaries 270
 - setting up data structures 269
 - starting at a page other than 1 272
 - triggering interconnections 273
 - using subforms on 269
- Wizard is Exited event 273
- Wizard is Finished - After event 273
- Wizard is Finished - Before event 273
- Wizard is Initialized event 272, 273
- WIZARD:Post Subform is Entered event 272
- WIZARD:Save for Re-entry event 269, 273
- WIZARD:Subform is Entered event 272
- WIZARD:Subform is Exited event 273, 275
- WIZARD:Subform is Initialized event 272
- WIZARD:Validate Subform event 273
- Wrap Text control property 39