

G2 GUIDE

User's Guide

Version 2015



G2 GUIDE User's Guide, Version 2015

March 2016

The information in this publication is subject to change without notice and does not represent a commitment by Gensym Corporation.

Although this software has been extensively tested, Gensym cannot guarantee error-free performance in all applications. Accordingly, use of the software is at the customer's sole risk.

Copyright (c) 1985-2016 Gensym Corporation

All rights reserved. No part of this document may be reproduced, stored in a retrieval system, translated, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of Gensym Corporation.

Gensym®, G2®, Optegrity®, and ReThink® are registered trademarks of Gensym Corporation.

NeurOn-Line™, Dynamic Scheduling™, G2 Real-Time Expert System™, G2 ActiveXLink™, G2 BeanBuilder™, G2 CORBALink™, G2 Diagnostic Assistant™, G2 Gateway™, G2 GUIDE™, G2GL™, G2 JavaLink™, G2 ProTools™, GDA™, GFI™, GSI™, ICP™, Integrity™, and SymCure™ are trademarks of Gensym Corporation.

Telewindows is a trademark or registered trademark of Microsoft Corporation in the United States and/or other countries. Telewindows is used by Gensym Corporation under license from owner.

This software is based in part on the work of the Independent JPEG Group.

Copyright (c) 1998-2002 Daniel Veillard. All Rights Reserved.

SCOR® is a registered trademark of PRTM.

License for Scintilla and SciTE, Copyright 1998-2003 by Neil Hodgson, All Rights Reserved.

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (<http://www.openssl.org/>).

All other products or services mentioned in this document are identified by the trademarks or service marks of their respective companies or organizations, and Gensym Corporation disclaims any responsibility for specifying which marks are owned by which companies or organizations.

Gensym Corporation
52 Second Avenue
Burlington, MA 01803 USA
Telephone: (781) 265-7100
Fax: (781) 265-7101

Part Number: DOC022-1200

Contents Summary

Preface xvii

Part I Introduction 1

Chapter 1 Introduction to G2 GUIDE 3

Chapter 2 Getting Started 25

Part II Creating a User Interface 43

Chapter 3 Generating Master Dialogs 45

Chapter 4 Building Master Dialogs 63

Chapter 5 Using UIL Controls on a Workspace 93

Chapter 6 Customizing Dialogs 99

Chapter 7 Launching Dialogs 131

Chapter 8 System-Defined Dialogs 151

Part III Editing User Interface Components 159

Chapter 9 Push Buttons 161

Chapter 10 Radio Buttons 185

Chapter 11 Check Buttons 201

Chapter 12 Toggle Buttons 217

Chapter 13 Edit Boxes, Combo Boxes, and Spin Controls 227

Chapter 14	Scroll Areas and Message Objects	257
Chapter 15	Sliders	291
Chapter 16	Text Objects	299
Chapter 17	Title Bars, Borders, and Separators	307
Chapter 18	Navigation Buttons and Other Tools	319

Part IV Advanced Features 327

Chapter 19	Formats and Validation Criteria	329
Chapter 20	Specifying Source and Target Objects	349
Chapter 21	Creating Temporary Storage Objects	365
Chapter 22	Methods, Actions, and Callbacks	379
Chapter 23	Help Dialog	393
Chapter 24	Creating Custom UIL Subclasses	405
Chapter 25	Specifying the Colors of UIL Objects	421
Chapter 26	Upgrading GUIDE Applications	429

Glossary 439

Index 449

Contents

Preface xvii

About this Guide xvii

Audience xvii

Organization xviii

Conventions xxi

Related Documentation xxiii

Customer Support Services xxv

Part I Introduction 1

Chapter 1 Introduction to G2 GUIDE 3

Introduction 3

Dialogs for Viewing and Editing Attribute Values 4

User Interface Components on Workspaces 6

Online Examples and Tutorial: the Demo KB 6

Programmatic Support for GUIDE: GUIDE/UIL 7

Using a GUIDE User Interface 7

Launching Dialogs 8

Controlling Dialogs with Push Buttons 10

Selecting UIL Controls 10

Scrolling a Scroll Area 11

Creating a GUIDE User Interface 12

Generating Master Dialogs 12

Building Customized Master Dialogs 16

Editing UIL Controls 22

UIL Methods, Actions, and Callbacks 24

Chapter 2 Getting Started 25

Introduction 26

Installing GUIDE 26

Merging GUIDE into Your KB for the First Time 26

Merging GUIDE into a KB with an Earlier Version 27
Verifying Your Version of GUIDE 27
License Requirements 28

The GUIDE/UIL Module Hierarchy 28
Module Support for Navigation Buttons 31

Removing Unneeded GUIDE Modules from an Application 31

Setting G2 Minimum Scheduling Parameter 32

Starting GUIDE 32

Reinitializing GUIDE 32

Choosing a User Mode 32

Using the GUIDE Menu Bar 33
Removing g2cuidev.kb 33
Reusing the GUIDE Submenus 34

Resetting the GUIDE Editor 34

Enabling and Disabling GUIDE/UIL User Menu Choices 34
Enabling and Disabling User Menu Choices for All Objects 35
Enabling and Disabling User Menu Choices for Particular Modules 35

Using GFR Startup Objects 36

Making UIL Controls Permanent 36

Printing GUIDE Workspaces 37

Suggestions and Cautions 39

Part II Creating a User Interface 43

Chapter 3 Generating Master Dialogs 45

Introduction 45

Using the GUIDE Dialog Generator 46
Master Dialogs with Default UIL Controls 46
Master Dialogs with non-Default UIL Controls 47

Steps for Generating a Master Dialog 48
Generating a Master Dialog with Default UIL Controls 52
Generating a Master Dialog with Non-Default UIL Controls 54

Launching Generated Dialogs from Push Buttons 59
Updating UIL Controls from the Initiating Object When the Dialog is
Launched by a Push Button 60
Using an Action to Specify an Initiating Object 61

	Editing Generated Dialogs	61
Chapter 4	Building Master Dialogs	63
	Introduction	64
	Using the GUIDE Palette	64
	Tools Provided by the GUIDE Palette	65
	Steps for Building a Master Dialog	67
	Adding a Dialog Title	73
	Adding Radio Buttons and Check Buttons	73
	Adding Scroll Areas and Message Objects	75
	Changing the Size of a Dialog Subworkspace	77
	Editing a Tab Dialog	78
	Lifting a Tab Page to the Top of the Stack	78
	Adding UIL Controls to a Tab Page	78
	Changing the Size and Labels of Tab Buttons	79
	Changing the Placement of Tab Buttons	80
	Adding New Tab Pages	81
	Deleting a Tab Page	82
	Cloning a Tab Page	82
	Reordering Tab Pages	82
	Lifting and Dropping Tab Pages with non-UIL Objects	83
	Moving the Stack of Tab Pages	83
	Resizing the Stack of Tab Pages	83
	Transferring Tab Pages	83
	Manipulating UIL Controls through User Menu Choices	84
	Moving UIL Controls	85
	Moving UIL Controls by Dragging Them	85
	Moving UIL Controls with Labels	85
	Moving UIL Controls with Borders	85
	Using the Move Menu Choice	86
	Resizing UIL Controls	87
	Transferring UIL Controls	87
	Specifying Initial Contents of Text Objects, Message Objects, and Edit Boxes	87
	Specifying Initial Contents of an Array or List Attribute	88
	Specifying Source and Target Attributes of UIL Controls	90
	Closing a Finished Subworkspace	90
	Creating a Customized Dialog Programmatically	91

Chapter 5	Using UIL Controls on a Workspace	93
	Introduction	93
	Examples of UIL Controls Used on Workspaces	93
	Invoking a Procedure from a Push Button on a Workspace	94
	Using an Edit Box on a Workspace	94
	Using a Scroll Area on a Workspace	96
	Placing UIL Objects on Subworkspaces of G2 Items	98
Chapter 6	Customizing Dialogs	99
	Introduction	99
	Editing Master Dialogs	100
	Edit Dialog Dialog	102
	Dialog Options Dialog	108
	Editor Behaviors Dialog	111
	Controlling Dialogs with Actions	115
	Specifying the Actions Run by a Push Button	116
	Creating Systems of Cascaded Dialogs	118
	Specifying a Default Button for a Dialog	120
	Using Dialogs on Multiple Windows	120
	Creating and Deleting Permanent Dialog Copies	121
	Internationalization of Dialogs	121
	GFR Objects that Support Internationalization	121
	UIL Object Attributes that Support Internationalization	123
	How the Translation Works	124
	Creating GFR Objects to Support Internationalization	125
	Summary of Dialog Menu Choices	129
Chapter 7	Launching Dialogs	131
	Introduction	131
	Pooling Reusable Dialogs for Quick Retrieval	132
	Using Dialogs in the Dialog Bin	132
	Releasing and Returning Dialogs	133
	Creating and Deleting Permanent Copies	133
	Procedures that Launch Dialogs	133
	Launching a Dialog from an Action Button	140
	Launching a Dialog from a User-Defined Procedure	142
	Processing a Dialog Before Returning it to the Dialog Bin	144

- Launching a Dialog from a User Menu Choice 145
- Launching a Dialog from a Push Button 147
 - Creating Push Buttons to Launch Dialogs 148
 - Specifying Source and Target Objects for UIL Controls on a Dialog Launched from a Push Button 148
- Launching a Dialog from a Rule 148

Chapter 8 System-Defined Dialogs 151

- Introduction 151
- Message, Query, Confirmation, and Notification Dialogs 151
 - Using uil-post-generic-dialog to Post Dialogs 152
 - Message Dialog 154
 - Query Dialog 154
 - Confirmation Dialog 155
 - Notification Dialog 156
- Delay Notification 158

Part III Editing User Interface Components 159

Chapter 9 Push Buttons 161

- Introduction 161
 - Adding Push Buttons to a Master Dialog 162
 - Using Push Buttons to Perform Operations on Dialogs 162
 - System-Defined Actions for Dialog Processing 164
 - Creating Actions 167
 - Setting a Target Object for a Push Button 167
 - Specifying Labels for Push Buttons 168
- Editing Pushbuttons 169
 - Edit Pushbutton Dialog 170
 - Edit Dialog Actions Dialog 175
 - Customize Dialog Actions Dialog 177
 - Create New Action Dialog 180
- Summary of Push Button Menu Choices 181

Chapter 10 Radio Buttons 185

- Introduction 185
 - Selecting Motif or Windows Style Buttons 186
 - Adding Radio Buttons to a Master Dialog 186
 - Moving Radio Buttons 187
 - Resizing Radio Buttons 187

	Deleting Radio Buttons and Radio Boxes	187
	Updating and Concluding Radio Buttons	187
	Specifying Labels for Radio Buttons	188
	Editing Radio Boxes	189
	Editing Radio Buttons	192
	Summary of Radio Box Menu Choices	196
	Summary of Radio Button Menu Choices	198
Chapter 11	Check Buttons	201
	Introduction	201
	Selecting Motif or Windows Style Buttons	202
	Adding Check Buttons to a Master Dialog	202
	Moving Check Buttons	203
	Resizing Check Buttons	203
	Updating and Concluding Check Buttons	203
	Specifying Labels for Check Buttons	204
	Editing Check Boxes	205
	Editing Check Buttons	208
	Summary of Check Box Menu Choices	212
	Summary of Check Button Menu Choices	214
Chapter 12	Toggle Buttons	217
	Introduction	217
	Selecting Motif or Windows Style Buttons	217
	Adding Toggle Buttons to a Master Dialog	218
	Updating and Concluding Toggle Buttons	218
	Specifying Labels for Toggle Buttons	219
	Editing Toggle Buttons	220
	Edit Toggle Button Dialog	221
	Summary of Toggle Button Menu Choices	225
Chapter 13	Edit Boxes, Combo Boxes, and Spin Controls	227
	Introduction	227
	Setting the Initial Contents of Edit Boxes	228
	Edit Styles for Edit Boxes	229
	Validating the Contents of Edit Boxes	229
	Keyboard Navigation to Edit Boxes	231
	Disabling Keyboard Navigation to an Edit Box	231
	Customizing Before and After Method Processing (Optional)	231

	Updating and Concluding Edit Boxes	232
	Editing Edit Boxes	232
	Edit Edit Box Dialog	233
	Select Edit Style Dialog	240
	Creating and Editing an Edit Field Edit Style	241
	Specifying a Password-Style Block Font	246
	Background Color and Text Color Dialogs	247
	Combo Boxes	247
	Editing Combo Boxes	248
	Spin Control Boxes	250
	Creating Spin Control Boxes	250
	Editing Spin Control Boxes	250
	Summary of Edit Box, Combo Box, and Spin Control Menu Choices	252
	Summary of Spin Control Box Menu Choices	253
	Summary of Combo Box Menu Choices	255
Chapter 14	Scroll Areas and Message Objects	257
	Introduction	257
	Adding Scroll Areas and Message Objects	258
	Resizing Scroll Areas and Message Objects	259
	Moving Scroll Areas and Message Objects	259
	Updating and Concluding Scroll Areas	259
	Specifying Formats for Message Objects	260
	Specifying Configurations for Scroll Areas and Message Objects	260
	Specifying Selection and Unselection Methods for Scroll Areas	262
	Managing Message Size in Scroll Areas	262
	Specifying User-Defined Methods for Message Objects	262
	Appending Items to Message Objects	263
	Editing Scroll Areas	264
	Edit Scroll Area Dialog	265
	Scroll Area Options Dialog	270
	Edit Message Dialog	273
	Multiple Column Scroll Areas	276
	Creating a Multiple-Column Scroll Area	277
	Creating Methods Required by Multiple-Column Scroll Areas	280
	Summary of Scroll Area Menu Choices	288
	Summary of Message Object Menu Choices	289

Chapter 15	Sliders	291
	Introduction	291
	Using Sliders	292
	Creating Sliders	293
	Editing Sliders	294
	Summary of Slider Menu Choices	296
Chapter 16	Text Objects	299
	Introduction	299
	Setting the Initial Contents of Text Objects	300
	Updating the Contents of Text Objects	300
	Specifying Formats for Text Objects	300
	Editing Text Objects	301
	Edit Text Dialog	301
	Summary of Text Object Menu Choices	304
Chapter 17	Title Bars, Borders, and Separators	307
	Introduction	307
	Title Bars	307
	Using the Hide Button on Title Bars	308
	Borders	309
	Adding Borders	309
	Deleting Borders	310
	Moving Objects with Borders	311
	Resizing Borders	311
	Edit Border Dialog	311
	Edit Border Margins Dialog	312
	Separators	313
	Summary of Title Bar Menu Choices	314
	Summary of Border Menu Choices	315
	Summary of Separator Menu Choices	317
Chapter 18	Navigation Buttons and Other Tools	319
	Introduction	320
	Navigation Buttons	320
	Classes of Navigation Buttons	321
	Modules Supporting Navigation Buttons	321

Edit Navigation Button Dialog	322
The Print Workspace Button	324
The GUIDE Garbage Pail	324
Summary of Navigation Button Menu Choices	325

Part IV Advanced Features 327

Chapter 19 Formats and Validation Criteria 329

Introduction	329
Formatting Rules for Edit Boxes, Message Objects, and Text Objects	329
Validation Criteria for Edit Boxes	330
Creating Customized Validation Procedures or Functions	330
Creating Formats	331
Applying and Editing Formats	333
Select Format Dialog	333
Edit Format Specification Dialog	335
Float Formatting Options Dialog	342
Text Formatting Options Dialog	343
Edit Legal Values Dialog	345
Date & Time Options Dialog	346

Chapter 20 Specifying Source and Target Objects 349

Introduction	349
Specifying Source and Target Objects	350
Source and Target Objects of Different UIL Controls	351
Edit Source Object & Attribute Dialog	352
Edit Target Object & Attribute Dialog	355
Updating from and Concluding to Embedded Objects	359

Chapter 21 Creating Temporary Storage Objects 365

Introduction	365
How Temporary Storage Objects Work	366
How Temporary Storage Objects Are Created	367
Steps for Defining a Temporary Storage Object for a Dialog	367
Creating this Example	369

Chapter 22	Methods, Actions, and Callbacks	379
	Introduction	379
	UIL Methods	380
	UIL Methods for Application Development	381
	UIL Methods for Runtime Operations	381
	How UIL Methods Work	382
	Object Attributes that Reference UIL Methods	384
	UIL Actions	386
	UIL Callbacks	386
	Callbacks on Push Buttons and Other Kinds of Buttons	386
	Callbacks in GUIDE 3.0 and GUIDE 4.0	387
	Creating Methods, Actions, and Callbacks	387
	Creating UIL Methods Using the Edit Method Dialog	387
	Creating Callbacks, Methods, Procedures, Functions, and Actions Using the GUIDE Method Help Dialog	390
Chapter 23	Help Dialog	393
	Introduction	393
	Opening the GUIDE Help Dialog	394
	Displaying Argument Signatures of UIL Methods, Callbacks, and Actions	395
	Displaying Help for UIL Methods	397
	Finding the UIL Help System File	398
	Generating Master Dialogs	399
	Using UIL Examples	400
	Using the GUIDE Online Tutorial	402
	Using the GUIDE Debugging Utility	402
Chapter 24	Creating Custom UIL Subclasses	405
	Introduction	405
	Creating and Using Customized Subclasses	406
	Choosing a Parent Class for a Customized Subclass	406
	Customizing the Behavior and Appearance of Subclasses of uil-grobj or uil-grmes	410
	Creating a Customized Object Definition	411
	Creating a Customized Message Definition	414

Creating Instances of Customized Subclasses and Adding them to Master Dialogs 417

Creating Subclasses of uil-object and uil-message 417

Deciding What Attributes to Add to a Subclass of uil-object or uil-message 418

Chapter 25 Specifying the Colors of UIL Objects 421

Introduction 421

Creating Configurations 422

Using The GUIDE Configuration Editor 422

Deleting Configurations 424

Copying Configurations 424

Editing Configurations 426

Applying Configuration Edits to All Buttons in a Group 428

Chapter 26 Upgrading GUIDE Applications 429

Introduction 429

Upgrading 5.0 KBs 430

4.0 and 5.0 Conversion Tools 430

Editing the Label Text of Generic Dialogs 430

Extending Context-Sensitive Help 436

How to Extend Context-Sensitive Help for Dialogs and Items on Dialogs 436

Steps for Extending Help 436

Glossary 439

Index 449

Preface

Describes this document and the conventions that it uses.

About this Guide **xvii**

Audience **xvii**

Organization **xviii**

Conventions **xxi**

Related Documentation **xxiii**

Customer Support Services **xxv**



About this Guide

This guide describes the G2 Graphical User Interface Development Environment (GUIDE), a development tool that enables you to create graphical user interfaces for G2 applications.

Users of this guide also need the *G2 GUIDE/UIL Procedures Reference Manual*, which describes the G2 GUIDE User Interface Library (GUIDE/UIL). GUIDE/UIL provides an application programmer's interface (API) to procedures that control dialogs and other elements of a graphical user interface.

Audience

This guide is written for GUIDE application developers. It addresses application developers as "you," and refers to end users of GUIDE applications as "the user" or "users."

Organization

This guide is divided into five parts and 26 chapters:

	Title	Description
Part I	Introduction	
1	Introduction to G2 GUIDE	Summarizes how to create and use a GUIDE user interface for a G2 application.
2	Getting Started	Describes preliminary steps for starting and running GUIDE, and explains how to avoid problems that can prevent you from running or saving your GUIDE application.
Part II	Creating a User Interface	
3	Generating Master Dialogs	Describes how to use the GUIDE Dialog Generator to generate a master dialog for viewing and editing attributes of a particular user-defined class.
4	Building Master Dialogs	Describes how to use the G2 GUIDE Palette and other GUIDE tools to build customized dialogs for viewing and editing the class-specific attributes of user-defined classes.
5	Using UIL Controls on a Workspace	Illustrates several ways to use UIL controls on a workspace, without incorporating them into a dialog.
6	Customizing Dialogs	Describes how to edit and customize GUIDE dialogs.
7	Launching Dialogs	Describes how to launch a dialog from an action button, a user-menu choice, a user-defined procedure, a push button, or a rule.

	Title	Description
8	Building Master Dialogs	Describes how to use the G2 GUIDE Palette and other GUIDE tools to build customized dialogs for viewing and editing the class-specific attributes of user-defined classes.
Part III Editing User Interface Components		
9	Push Buttons	Describes how you can create and edit push buttons to run specific sets of actions on dialogs, such as opening and closing them, or updating or concluding their values.
10	Radio Buttons	Describes how to create and edit groups of radio buttons to provide users with sets of mutually exclusive choices.
11	Check Buttons	Describes how to create and edit groups of check buttons, in which users can select any number of choices.
12	Toggle Buttons	Describes how to create and edit toggle buttons, which represent two mutually exclusive choices.
13	Edit Boxes, Combo Boxes, and Spin Controls	Describes how to create and customize edit boxes, combo boxes, and spin controls.
14	Scroll Areas and Message Objects	Describes how to create and edit scroll areas and message objects.
15	Sliders	Describes how to use and edit Sliders.
16	Text Objects	Describes how to create and edit text objects, which display read-only text.

	Title	Description
17	Title Bars, Borders, and Separators	Describes how to use title bars, border, and separators to provide your user interface with visual definition.
18	Navigation Buttons and Other Tools	Describes how to add navigation buttons, help buttons, print workspaces, and the GUIDE garbage pail to workspaces.
<hr/>		
Part IV	Advanced Features	
19	Formats and Validation Criteria	Describes how to create and edit reusable formats for edit boxes, message objects, and text objects.
20	Specifying Source and Target Objects	Describes how to specify the objects from which the graphical objects on a dialog are updated and to which the graphical objects conclude their values.
21	Creating Temporary Storage Objects	Describes how to create and use temporary storage objects, which you can use when you process data while it is being updated into or concluded from a dialog.
22	Methods, Actions, and Callbacks	Describes how to create and use UIL methods, actions, and callbacks.
23	Help Dialog	Describes the GUIDE help facility.
24	Creating Custom UIL Subclasses	Describes how to create customized subclasses of system-defined UIL classes provided with GUIDE.

	Title	Description
25	Specifying the Colors of UIL Objects	Describes how to create reusable objects called configurations, which specify the colors of the different regions of the graphical components in your user interface.
26	Upgrading GUIDE Applications	Describes how to modify dialogs and other components of a user interface created with earlier versions of GUIDE, to take advantages of the features introduced in newer versions.

Conventions

This guide uses the following typographic conventions and conventions for defining system procedures.

Typographic

Convention Examples	Description
g2-window, g2-window-1, ws-top-level, sys-mod	User-defined and system-defined G2 class names, instance names, workspace names, and module names
history-keeping-spec, temperature	User-defined and system-defined G2 attribute names
true, 1.234, ok, "Burlington, MA"	G2 attribute values and values specified or viewed through dialogs
Main Menu > Start KB Workspace > New Object create subworkspace Start Procedure	G2 menu choices and button labels

Convention Examples	Description
conclude that the x of y ...	Text of G2 procedures, methods, functions, formulas, and expressions
<i>new-argument</i>	User-specified values in syntax descriptions
<u><i>text-string</i></u>	Return values of G2 procedures and methods in syntax descriptions
File Name, OK, Apply, Cancel, General, Edit Scroll Area	GUIDE and native dialog fields, button labels, tabs, and titles
File > Save	GMS and native menu choices
Properties	
workspace	Glossary terms
<i>c:\Program Files\Gensym\</i>	Windows pathnames
<i>/usr/gensym/g2/kbs</i>	UNIX pathnames
<i>spreadsh.kb</i>	File names
<i>g2 -kb top.kb</i>	Operating system commands
<i>public void main() gsi_start</i>	Java, C and all other external code

Note Syntax conventions are fully described in the *G2 Reference Manual*.

Procedure Signatures

A procedure signature is a complete syntactic summary of a procedure or method. A procedure signature shows values supplied by the user in *italics*, and the value (if any) returned by the procedure *underlined*. Each value is followed by its type:

```
g2-clone-and-transfer-objects
  (list: class item-list, to-workspace: class kb-workspace,
   delta-x: integer, delta-y: integer)
  -> transferred-items: g2-list
```

Related Documentation

G2 Core Technology

- *G2 Bundle Release Notes*
- *Getting Started with G2 Tutorials*
- *G2 Reference Manual*
- *G2 Language Reference Card*
- *G2 Developer's Guide*
- *G2 System Procedures Reference Manual*
- *G2 System Procedures Reference Card*
- *G2 Class Reference Manual*
- *Telewindows User's Guide*
- *G2 Gateway Bridge Developer's Guide*

G2 Utilities

- *G2 ProTools User's Guide*
- *G2 Foundation Resources User's Guide*
- *G2 Menu System User's Guide*
- *G2 XL Spreadsheet User's Guide*
- *G2 Dynamic Displays User's Guide*
- *G2 Developer's Interface User's Guide*
- *G2 OnLine Documentation Developer's Guide*
- *G2 OnLine Documentation User's Guide*
- *G2 GUIDE User's Guide*
- *G2 GUIDE/UIIL Procedures Reference Manual*

G2 Developers' Utilities

- *Business Process Management System Users' Guide*
- *Business Rules Management System User's Guide*
- *G2 Reporting Engine User's Guide*
- *G2 Web User's Guide*
- *G2 Event and Data Processing User's Guide*

- *G2 Run-Time Library User's Guide*
- *G2 Event Manager User's Guide*
- *G2 Dialog Utility User's Guide*
- *G2 Data Source Manager User's Guide*
- *G2 Data Point Manager User's Guide*
- *G2 Engineering Unit Conversion User's Guide*
- *G2 Error Handling Foundation User's Guide*
- *G2 Relation Browser User's Guide*

Bridges and External Systems

- *G2 ActiveXLink User's Guide*
- *G2 CORBALink User's Guide*
- *G2 Database Bridge User's Guide*
- *G2-ODBC Bridge Release Notes*
- *G2-Oracle Bridge Release Notes*
- *G2-Sybase Bridge Release Notes*
- *G2 JMail Bridge User's Guide*
- *G2 Java Socket Manager User's Guide*
- *G2 JMSLink User's Guide*
- *G2 OPCLink User's Guide*
- *G2 PI Bridge User's Guide*
- *G2-SNMP Bridge User's Guide*
- *G2 CORBALink User's Guide*
- *G2 WebLink User's Guide*

G2 JavaLink

- *G2 JavaLink User's Guide*
- *G2 DownloadInterfaces User's Guide*
- *G2 Bean Builder User's Guide*

G2 Diagnostic Assistant

- *GDA User's Guide*
- *GDA Reference Manual*
- *GDA API Reference*

Customer Support Services

You can obtain help with this or any Gensym product from Gensym Customer Support. Help is available online, by telephone, by fax, and by email.

To obtain customer support online:

➔ Access G2 HelpLink at www.gensym-support.com.

You will be asked to log in to an existing account or create a new account if necessary. G2 HelpLink allows you to:

- Register your question with Customer Support by creating an Issue.
- Query, link to, and review existing issues.
- Share issues with other users in your group.
- Query for Bugs, Suggestions, and Resolutions.

To obtain customer support by telephone, fax, or email:

➔ Use the following numbers and addresses:

	Americas	Europe, Middle-East, Africa (EMEA)
Phone	(781) 265-7301	+31-71-5682622
Fax	(781) 265-7255	+31-71-5682621
Email	service@gensym.com	service-ema@gensym.com

Introduction

Chapter 1: Introduction to G2 GUIDE

Summarizes how to create and use a GUIDE user interface for a G2 application.

Chapter 2: Getting Started

Describes preliminary steps for starting and running GUIDE, and explains how to avoid problems that can prevent you from running or saving your GUIDE application.

Introduction to G2 GUIDE

Summarizes how to create and use a GUIDE user interface for a G2 application.

Introduction	3
Using a GUIDE User Interface	7
Creating a GUIDE User Interface	12
UIL Methods, Actions, and Callbacks	24



Introduction

The G2 Graphical User Interface Development Environment (GUIDE) enables you to create graphical user interfaces (GUI's) for G2 applications.

You construct a GUIDE user interface using graphical components called **UIL controls**.

GUIDE supports different classes of UIL controls for different purposes:

- Some classes of UIL controls, such as edit boxes, buttons, and scroll areas, enable users to view and edit the data stored in object attributes. The different classes are suitable for viewing and editing different types of data.
- Other classes of UIL controls, such as borders and separators, enable you to organize a user interface visually.

You can add UIL controls to interactive dialogs, which you design using GUIDE. You can also add UIL controls to workspaces of your G2 application.

Dialogs for Viewing and Editing Attribute Values

GUIDE interactive dialogs can add structure and flexibility to your user interface. Many of the more powerful features of GUIDE are available only through dialogs.

Useful Features of Dialogs Created with GUIDE

G2 applications store information that is valuable to users in class-specific attributes of G2 objects. Users can edit these attribute values directly in the attribute tables of G2 objects.

Through the user interface that you create using GUIDE, users can view and edit the information stored in these attributes.

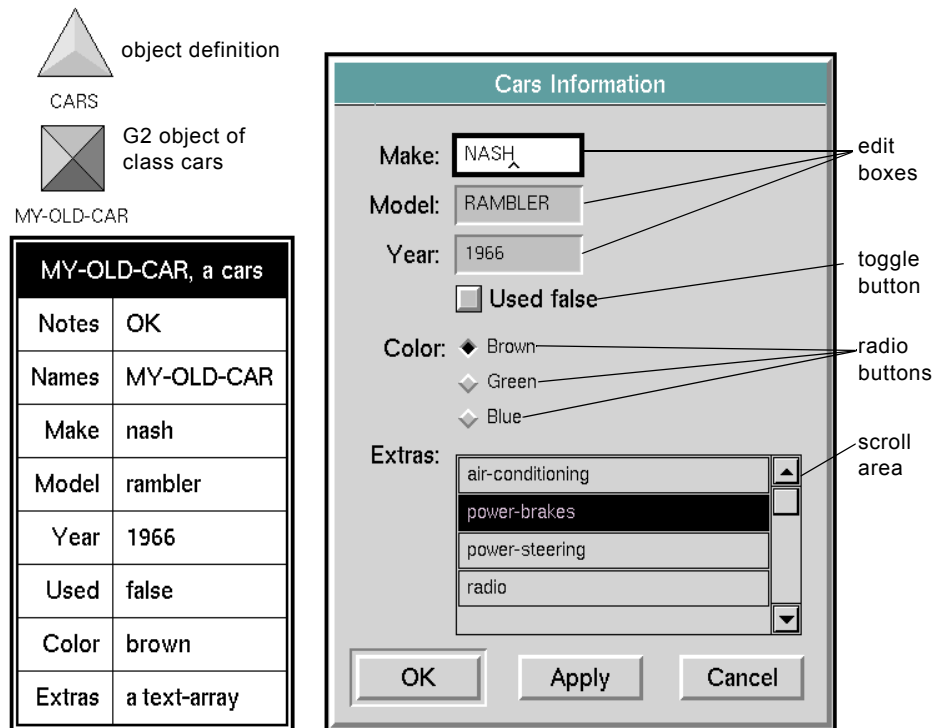
However, enabling users to view and edit attribute values through dialogs has significant advantages over requiring them to view and edit attributes directly in attribute tables:

- You can create dialogs that enable users to view and edit attribute values from several different G2 objects.
- You can validate the changes that users make to attribute values, using criteria that you specify.
- You can add push buttons to a dialog that enable users to perform operations such as:
 - Updating the display in the dialog with current attribute values
 - Modifying attribute values using the contents of the dialog
 - Opening or closing the dialog
 - Opening another dialog
- You can specify the format used to display attribute values.
- You can process attribute values before you display them in a dialog. For example, you can find the average value of a given attribute in several different G2 objects and display this average in a dialog.
- You can take advantage of GUIDE/UII features that enable you to specify any number of different natural language versions of the text in labels of UII controls on dialogs.

Representing Different Types of Data

UII provides different classes of UII controls that you can use to represent attribute values in different ways. Each class is appropriate to a particular type of data. For example, edit boxes are suitable for displaying and editing text values, and scroll areas are suitable for displaying lists and arrays of values.

The following figure illustrates a G2 object and the UIL controls used on a dialog to view and edit the object's attributes.



The dialog named Cars Information in the figure above illustrates how different classes of UIL controls can be used to represent class-specific attributes of a G2 object:

- The attributes Make, Model, and Year contain text. They are represented on the dialog by edit boxes. Each edit box displays the current value of the corresponding attribute. A user can change the attribute value by editing the contents of the edit box.
- The Used attribute can have one of two mutually exclusive values, **true** or **false**. It is represented by a toggle button. The current state of the toggle button indicates the current value of the **used** attribute. A user can change the value of the **used** attribute by toggling the **used** button on or off.
- The Color attribute can have one of three mutually exclusive values, **Brown**, **Blue**, or **Green**. It is represented by a group of three radio buttons. Each button in the group represents one of the possible values of the Color attribute. The current value of the Color attribute is indicated by the radio button that is selected; only one button in a radio button group can be selected at one time. A user can change the value of the Color attribute by selecting a different radio button.

- The Extras attribute references an array of text elements. The text elements can be used to specify features such as power steering, power brakes, air conditioning, or radio.

On the dialog, the Extras attribute is represented by a scroll area. The scroll area contains message objects representing the elements of the text array. The figure above illustrates how the scroll area looks when the text array contains the elements air-conditioning, power-steering, and radio.

User Interface Components on Workspaces

You can add any UIL control directly to a workspace of your application, where you use it without the support of a GUIDE interactive dialog. However, you give up much of the structure and flexibility provided by dialogs when you use UIL controls on workspaces, for example, you cannot use push buttons to perform operations on UIL controls that are not on a dialog.

For examples of how to use UIL controls on workspaces, see [Using UIL Controls on a Workspace](#).

Online Examples and Tutorial: the Demo KB

GUIDE is shipped with a demo knowledge base, *guidemo.kb*, that contains:

- Working examples of GUIDE dialogs, and other important features of a GUIDE user interface.

You can look at these examples as illustrations of how a GUIDE user interface can be designed. You can also copy and modify examples to use in your own applications.
- An online tutorial that shows you how to use GUIDE to create a user interface.

To open the main workspace of the demo knowledge base:

- 1 Click the Help System navigation button in the GUIDE palette.
This button has a question mark (?) on it. The GUIDE Help dialog appears.
- 2 Click the navigation button labeled UIL Examples to display the main workspace of the demo knowledge base.

Note If you have not loaded *guidemo.kb*, you see a message informing you that *guidemo.kb* is not loaded. If you see this message, pause your KB, merge in *guidemo.kb* (select the **automatically resolve conflicts** option when you merge), and restart your knowledge base.

Programmatic Support for GUIDE: GUIDE/UIL

GUIDE is supported by the G2 GUIDE User Interface Library (GUIDE/UIL). GUIDE/UIL provides an application programmer's interface (API) to procedures that perform basic operations on dialogs and UIL controls.

You can create working dialogs without writing code or using UIL procedures explicitly.

However, you can extend the basic capabilities of GUIDE by using UIL procedures in customized methods, actions, callbacks, and procedures. For information about how to create customized versions of these procedures, see [Methods, Actions, and Callbacks](#).

This guide contains examples of the use of UIL procedures in the following chapters or sections:

- [Launching Dialogs](#)
- [Multiple Column Scroll Areas](#)
- [System-Defined Dialogs](#)
- [Creating Temporary Storage Objects](#)

The demo knowledge base also contains an extensive set of examples of the use of UIL procedures. For information about how to access the demo knowledge base, see [Online Examples and Tutorial: the Demo KB](#).

For detailed information about GUIDE/UIL, see the *G2 GUIDE/UIL Procedures Reference Manual*.

Using a GUIDE User Interface

Users of a GUIDE user interface need to be able to do the following things:

- Launch dialogs
- Update the values displayed in a dialog with current information
- Edit the values in a dialog
- Send the values in a dialog back to the application
- Close dialogs

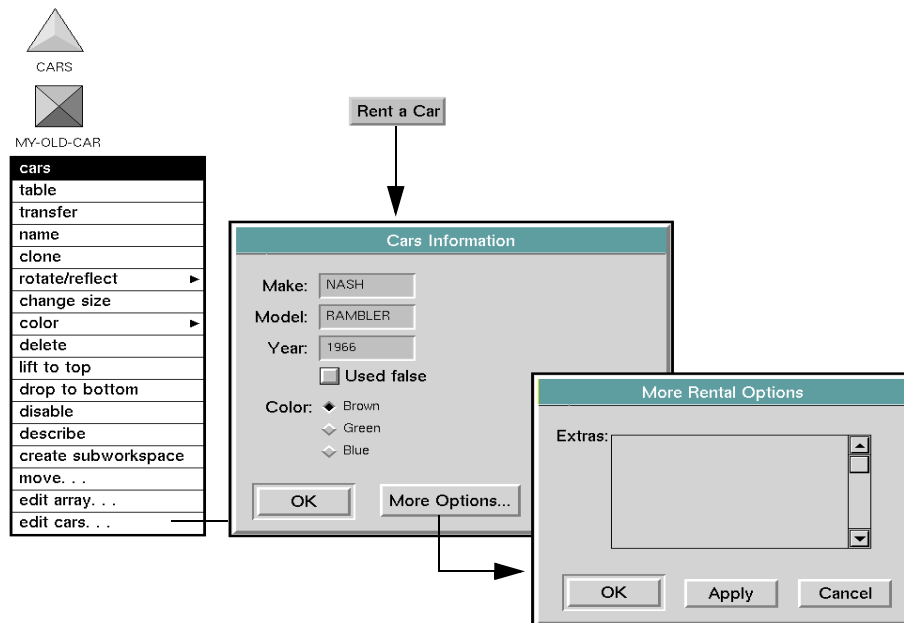
GUIDE enables you to incorporate all of these features into the user interface that you construct using GUIDE.

Launching Dialogs

Users of a GUIDE user interface can launch dialogs in the following ways:

- By selecting a user-menu choice from the menu of a G2 object whose attributes they want to view or edit
- By clicking a G2 action button
- By clicking a UIL push button in a dialog that is already open
- By executing a procedure, action, or rule

The following figure illustrates some of the ways that users can launch dialogs in a GUIDE user interface:



In the figure above:

- The user menu choice **edit cars** on the menu of a G2 object named **rambler** launches the dialog **Cars Information**.

The **edit cars** user menu choice starts a UIL procedure that launches the dialog, as shown in the following table:

an user-menu-choice	
Notes	USER-MENU-CHOICE-XXX-510: OK
Authors	none
Names	none
Label	edit-cars.-.-.
Applicable class	cars
Condition	none
Action	start uil-start-or-refocus-dialog ("Cars Information", the item, this window)
Action priority	2

- The G2 action button labeled **Rent a Car** launches the dialog **Cars Information**.

The **Rent a Car** button starts a UIL procedure that launches the dialog, as shown in the following table:

RENT-CAR, an action-button	
Notes	OK
Names	RENT-CAR
Label	"Rent a Car"
Action	start uil-start-dialog ("Cars-information", "rent a car", this window)
Action priority	2

- The **More Options** push button in the **Cars Information** dialog launches the **More Rental Options** dialog.

Launching Dialogs from within Other Dialogs

As the example in the figure above indicates, users can open dialogs from within other dialogs. Users can click a push button in one dialog, the **parent dialog**, to open another dialog, the **child dialog**. Parent and child dialogs that you associate in this way are called **cascaded dialogs**. For information about how to create cascaded dialogs, see [Creating Systems of Cascaded Dialogs](#).

Controlling Dialogs with Push Buttons

Push buttons enable users to perform a variety of operations on dialogs, such as launching and closing them, or updating or concluding the values of the UIL controls on the dialogs.

These operations are performed by procedures known as **actions**. When a user clicks on a push button, a set of actions associated with that push button is run on a specified dialog. This can be the dialog that contains the push button, or another dialog. You can edit a push button to specify the actions that are run when a user clicks on it, and the dialog on which the actions are run.

By default, each dialog that you create contains OK, Apply, and Cancel buttons that run appropriate actions on the dialog:

- The OK button applies the changes that the user has made in the dialog and closes the dialog.
- The Apply button applies the changes but leaves the dialog open.
- The Cancel button closes the dialog without applying the changes.

GUIDE provides an extensive set of system-defined actions that perform common operations, such as opening and closing dialogs, and updating or concluding their contents. You can also create customized actions to perform customized operations.

You can edit any push button to make it run any system-defined or user-defined action.

Selecting UIL Controls

Users can select any UIL control by clicking on it. Clicking on different UIL controls has different effects:

- Clicking on a button selects that button. Clicking on a radio button also deselects any other radio button in the same group.
- Clicking on a message object in a scroll area selects the message object.
- Clicking on a push button activates the push button, causing the callback procedure associated with the button to run. The callback procedure can run a

set of actions on a specified dialog, or it can run a procedure or procedures to perform other operations required by your application.

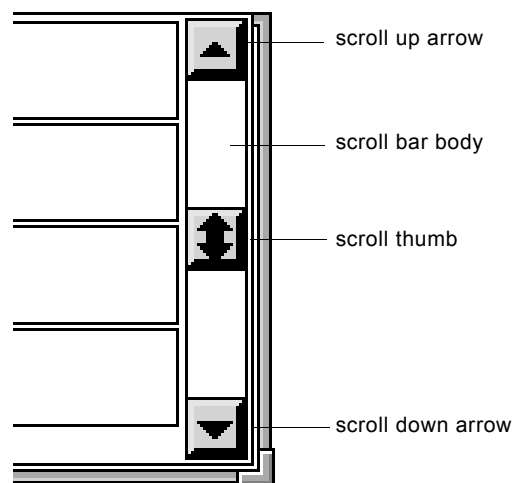
- Clicking on an edit box starts an edit session on the contents of the edit box and ends the edit session on any other edit box in the same dialog.

Users can navigate through the editable text fields (edit boxes) in a GUIDE dialog using the Tab key or other key. You can edit the sequence in which the Tab key navigates through edit boxes.

Scrolling a Scroll Area

You use the scroll bar on a scroll area to scroll its contents up or down. To do this, you can click the following parts of a scroll bar: the scroll down arrow, the scroll bar body, the scroll up arrow, or the scroll thumb.

The following figure illustrates the parts of a scroll bar:



Clicking on the parts of the scroll bar has the follow effects:

- Clicking on the scroll up arrow scrolls message objects down one line.
- Clicking on the scroll down arrow scrolls messages objects up one line.
- Clicking on the scroll bar body above the scroll thumb scrolls message objects down one page.
- Clicking on the scroll bar body below the scroll thumb scrolls message objects up one page.
- Dragging the scroll thumb up moves message objects down. Dragging the scroll thumb down moves message objects up.

Creating a GUIDE User Interface

GUIDE provides all the tools you need to create a user interface that meets the requirements of your G2 application.

Using GUIDE, you can:

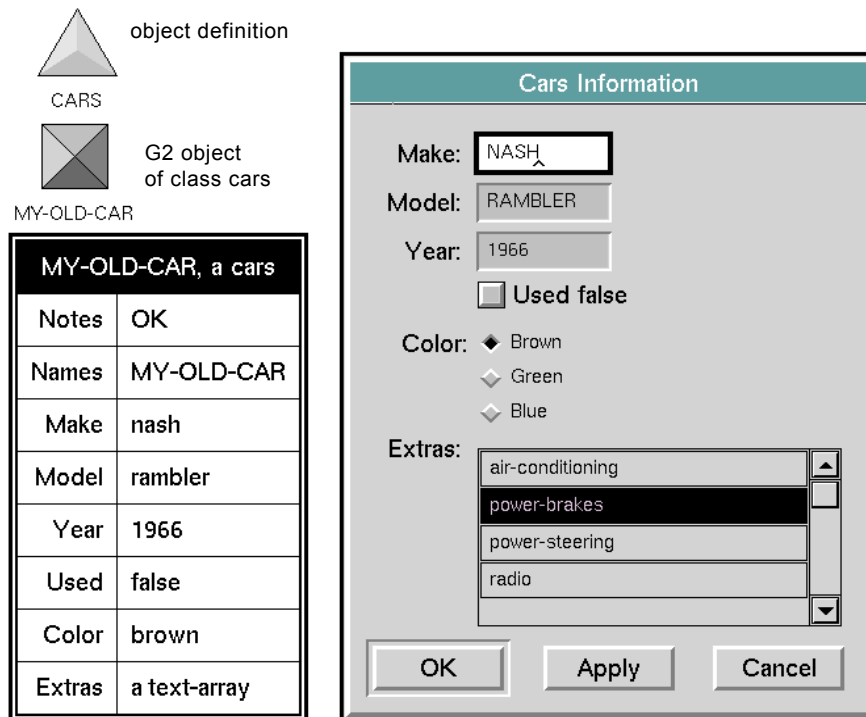
- Generate master dialogs automatically.
A **master dialog** is a template for the dialogs that users see and use when they run your G2 application. The dialogs that users see and use are called **copy dialogs**. GUIDE creates copy dialogs by cloning master dialogs that you create.
- Build customized master dialogs.
- Add UIL controls to workspaces.
- Edit the appearance and behavior of UIL controls.
- Write customized procedures for performing common operations such as opening and closing dialogs, or updating and concluding the values that they contain. You use your customized procedures in place of system-defined UIL procedures that perform these operations by default.
- Access useful information about UIL procedures.

Generating Master Dialogs

The GUIDE Dialog Generator enables you to generate master dialogs for viewing and editing the class-specific attributes of particular user-defined classes.

In the GUIDE Dialog Generator, you select the class-specific attributes that you want users to be able to view and edit. In the generated master dialog, each of the attributes that you select is represented by a UIL control through which users can view and edit the value of that attribute.

For example, the following figure illustrates an automatically generated GUIDE dialog that enables users to view and edit the class-specific attributes of an instance of the user-defined class `cars`:



The dialog titled `Cars Information` contains a UIL control for each class-specific attribute of objects of the class `cars` such as `my-old-car`.

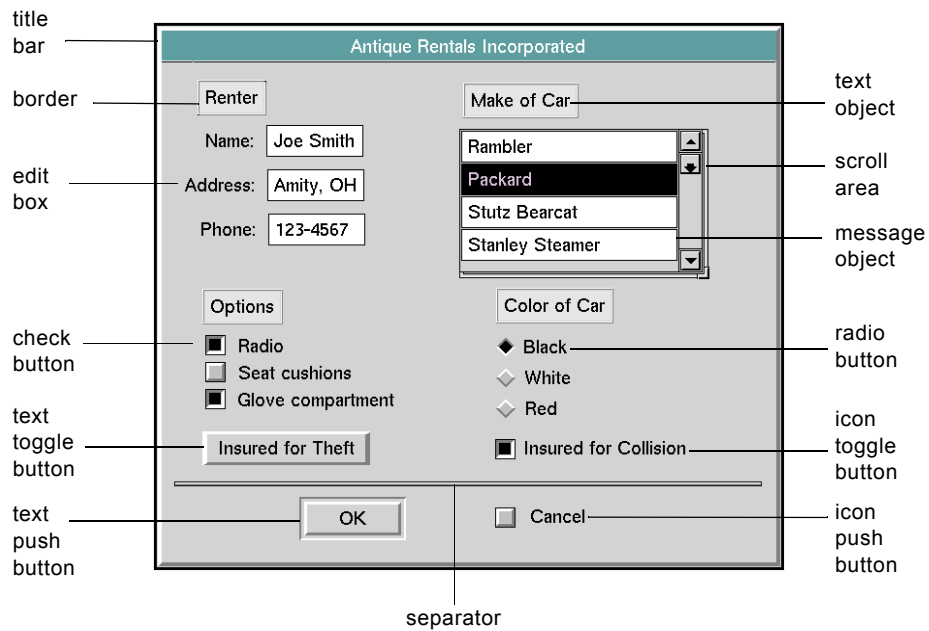
To view attribute values of `my-old-car`, a user opens the dialog. When the dialog is displayed, the edit boxes are updated with current values of the corresponding attributes.

To edit attribute values of `my-old-car`, a user enters values into the edit boxes associated with the attributes and clicks the `OK` or `Apply` button. This writes the values in the edit boxes to the corresponding attributes.

Users can view and edit the attributes of any other G2 object of the class `cars`, using the `Cars Information` dialog.

By default, the GUIDE Dialog Generator chooses an appropriate kind of UIL control to represent each attribute, depending on the data type of the attribute. For example, the GUIDE Dialog Generator uses edit boxes to represent text attributes and scroll areas to represent attributes that reference lists or arrays. You can override these defaults and choose other kinds of UIL controls to represent particular attributes.

The following figure illustrates a GUIDE dialog that includes most UIL controls provided by GUIDE:



The following table describes the UIL controls that you can use to create a user interface:

UIL Controls

UIL Control	Description
Title bar	A label for the dialog that appears at the top of the dialog subworkspace.
Push button	Provides users with control over dialogs. Push buttons can perform operations such as updating or concluding the values in a dialog, closing a dialog, or opening another dialog.
Edit Box, Spin Control Entry Box, Combo Box	Displays values that users can edit. Spin control entry boxes and combo boxes were introduced in GUIDE 6.0 and are not shown above.

UI Controls

UI Control	Description
Check button	Used in groups called check boxes. Each check button has one value when selected and a different value when unselected. Users can select any number of check buttons in a check box.
Radio button	Used in groups called radio boxes. Each radio button has one value when selected and a different value when unselected. Users must select one and only one radio button in a radio box.
Toggle button	Used individually, rather than in groups. Each toggle button has one value when selected and another value when unselected.
Text object	Displays text. The text can be updated by your application, but users cannot edit it.
Scroll Area and Message Object	A scrollable field containing one or more message objects. Each message object displays text. Your application can update message objects. Users can select message objects but cannot edit their contents.
Sliders	A graphical object with a pointer that moves along a horizontal or vertical track. The position of the pointer indicates the current value of the slider. Users can change the value by dragging the pointer.
Border	A box that surrounds a workspace, edit box, or text object, providing visual definition.

UIL Controls

UIL Control	Description
Separator	A vertical or horizontal line, providing visual definition.
Navigation button	Buttons that users can click to move from one workspace to another. Note: Do not attempt to use navigation buttons in dialogs that you create using GUIDE.

Building Customized Master Dialogs

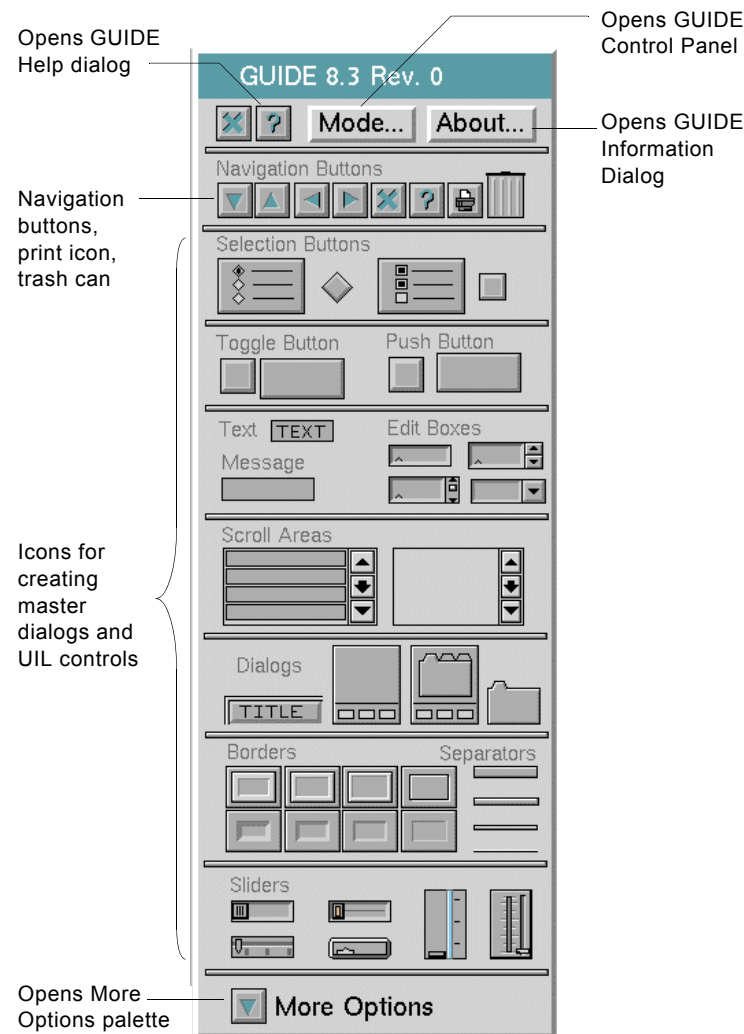
For some purposes an application can require dialogs that are more fully customized than dialogs that you create using the GUIDE Dialog Generator. The GUIDE palette is the tool that enables you to create fully customized master dialogs. For detailed information about how to build a customized dialog, see [Building Master Dialogs](#).

Using the GUIDE palette, you can:

- Create master dialogs.
- Add UIL controls to existing master dialogs.
- Change the layout and appearance of master dialogs.
- Add UIL controls directly to a workspace, without incorporating them into a dialog.

Features of the GUIDE Palette

The GUIDE palette is a workspace that includes basic tools for developing a GUIDE user interface. The GUIDE palette looks like this:



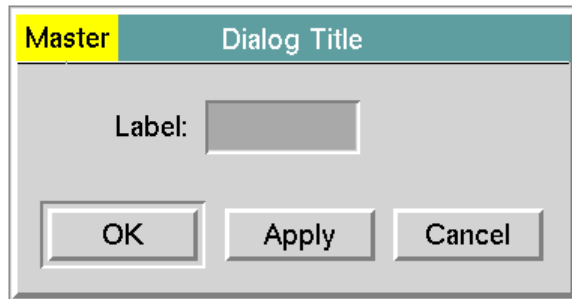
Using Icons for Dialogs and UIL Controls

To create a master dialog:

- 1 Click the dialog icon in the GUIDE palette.
Clicking on the dialog icon clones a master dialog. You then click the workspace where you want to put the master dialog. Clicking on the workspace transfers the cloned dialog to that workspace.
- 2 Add UIL controls to the new master dialog. To do this, you first open the subworkspace of the dialog. Then you add UIL controls to the subworkspace

by clicking on their icons on the GUIDE palette and then clicking on the subworkspace.

The following figure illustrates how to create a master dialog and add a UIL control to it using the GUIDE palette:



subworkspace of master dialog



Note The title bar of every master dialog contains the word Master. If you do not see Master in the title bar of a dialog subworkspace, that dialog is not a master and you cannot edit it.

Using the GUIDE Control Panel

The GUIDE Control Panel enables you to select a user mode for running GUIDE, and to specify the default window style and size of the UIL controls that you create using the GUIDE palette.

You open the GUIDE Control Panel by clicking on the Mode button in the GUIDE palette.

You can also open the GUIDE Control Panel by selecting the following choice from the GUIDE menu bar:

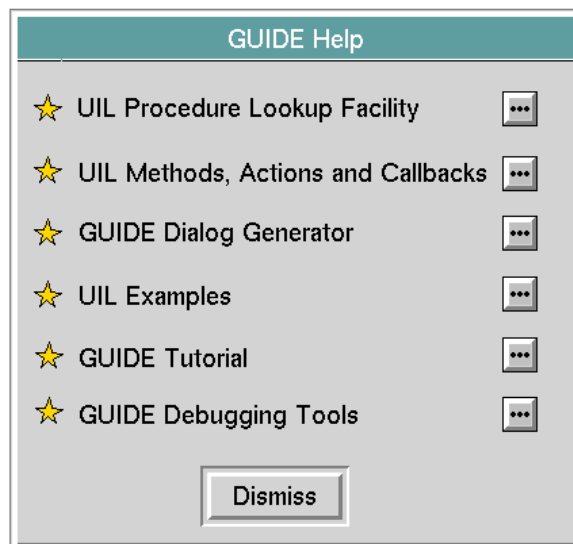
Tools > GUIDE Control Panel

The GUIDE Control Panel looks like this:



Using the GUIDE Help Dialog

You access the GUIDE Help dialog by clicking on the question mark (?) icon in the GUIDE palette. The main dialog of the GUIDE Help dialog looks like this:

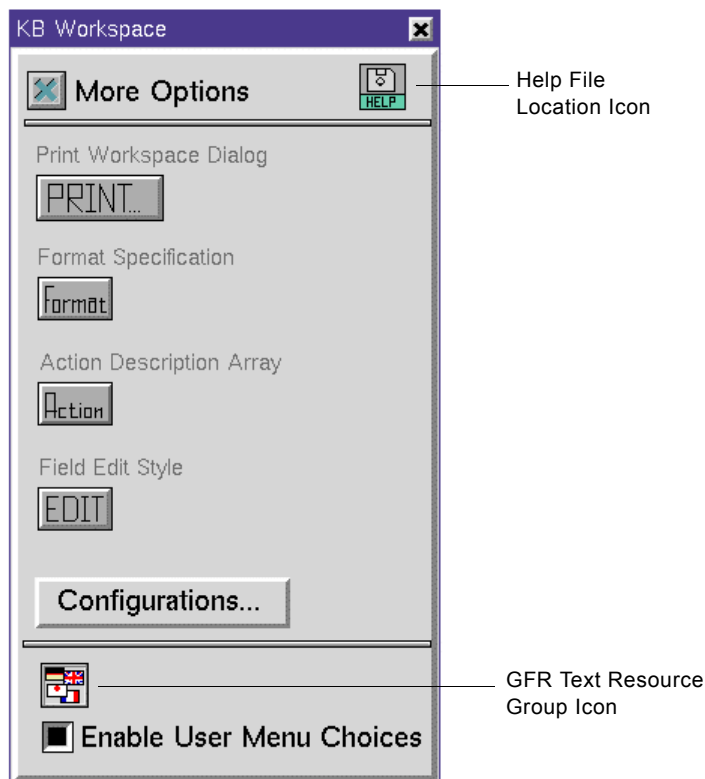


You can click the buttons in the GUIDE Help dialog to:

- Display information about each system-defined UIL procedure. The information includes a synopsis of the procedure's arguments and return values.
- Display information about UIL methods and invoke editors that help you create customized methods and actions to use in place of the system defined methods, actions, and callbacks.
- Invoke the GUIDE Dialog Generator.
- Open a workspace that contains working examples of dialogs and UIL controls. This workspace contains the examples in *guidemo.kb*.
- Run the online GUIDE tutorial, which leads you through the basic steps of creating a user interface with GUIDE.
- Start the GUIDE Debugging Utility, which enables you to display messages that help you debug your GUIDE application.

Using More Options

Clicking on the More Options navigation button in the GUIDE palette opens the More Options palette:



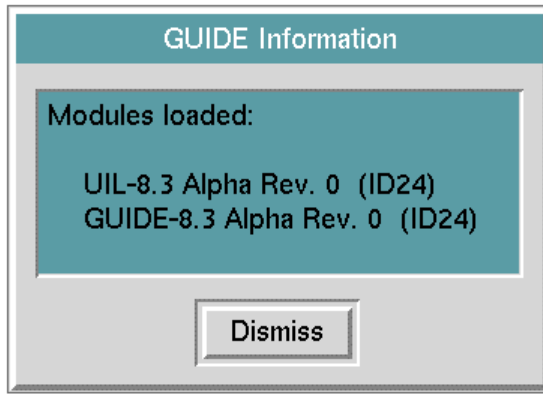
The More Options palette contains icons that you can select and drop on a workspace to create:

- A help file location icon that you can click to open a table listing the file name and current pathname of the help system file, which lists the procedures in the public API to UIL. For information about this file, see [Finding the UIL Help System File](#).
- A Print button. Users can click the Print button to open the Printer Setup Dialog. In this dialog, users can specify options for printing the workspace that contains the Print button.
- A reusable Format to specify aspects of format such as the use of quotation marks, capitalization, and hyphens. You can apply formats to edit boxes, text objects, and message objects. Formats can also specify validation criteria for edit boxes. For information about formats, see [Formats and Validation Criteria](#).
- An action description array. An action description array is an array of operations that can be run on a dialog. You can invoke an action description array on a dialog by invoking the procedure `uil-control-dialog-callback`. For information about action description arrays, see [Controlling Dialogs with Actions](#).
- A reusable field edit style that specifies the behavior of the editor for edit boxes. You can apply field edit styles to individual edit boxes. For information about field edit styles, see [Edit Styles for Edit Boxes](#).
- A Configurations button that you can click to open the Create Configuration dialog. For information about how to create and use configurations, see [Specifying the Colors of UIL Objects](#).
- A GFR Text Resource Group icon. The subworkspace of this icon contains a default English GFR Local Text Resource. Generic dialogs derive the text of their labels from GFR (Gensym Foundation Resources) local text resources. For information about how to edit GFR Local Text Resources to specify GUIDE label text, see [Editing the Label Text of Generic Dialogs](#).
- An Enable User Menu Choices button, which you can click to enabled GUIDE/UIL user menu choices.

GUIDE/UIL user menu choices end in a period (.), to distinguish them from G2 user menu choices. Always use the GUIDE/UIL user menu choice, rather than its G2 equivalent, to perform an operation on a UIL object. For example, always use the menu choice `clone.` (rather than `clone`) to clone a dialog.

How to Find What Version of GUIDE You Are Using

To find out what version of GUIDE you are using, open the GUIDE Information dialog:



You can open this dialog in either of two ways:

- Click the About navigation button in the GUIDE palette.
- Select Help/About GUIDE from the GUIDE menu bar.

Editing UIL Controls

GUIDE provides graphical editors that enable you to edit the appearance and behavior of dialogs and of UIL controls, such as edit boxes, push buttons, and scroll areas.

Note You can also edit the attributes of UIL controls directly, in their attribute tables. You may find it convenient to edit UIL controls through their attribute tables when you are debugging an application. However, using the graphical editors to edit UIL controls reduces the possibility of error and is the recommended method for most purposes.

Editing the Behavior and Appearance of Dialogs and UIL Controls

You can edit the appearance and run-time behavior of dialogs and UIL controls, using specialized graphical editors. Each class of UIL control has its own graphical editor, which is described in a separate chapter in this guide. For example, the object editor for push buttons is described in [Push Buttons](#).

Specifying Source and Target Attributes of UIL Controls

When you add UIL controls to a dialog by means of the GUIDE palette, you must edit each UIL control to specify its **source attribute** and **target attribute**.

- The source attribute is the class-specific attribute of a G2 object (the **source object**) whose value is reflected in the UIL control.
- The target attribute is the class-specific attribute of a G2 object (the **target object**) to which the UIL control concludes its value.

GUIDE provides two graphical editors that enable you to specify source attributes and target attributes: Edit Source Object & Attribute and Edit Target Object & Attribute.

When you generate a dialog automatically, GUIDE automatically sets the source and target attributes of the UIL controls on the dialog to the corresponding attributes in the G2 class for which you are creating the dialog. If necessary, you can change the defaults source and target attribute, using the Edit Source Object & Attribute and Edit Target Object & Attribute editors.

Creating UIL Controls with Customized Appearance and Behavior

For some purposes, you may want to create UIL controls with a different appearance or behavior from the UIL controls that you can create using the GUIDE Dialog Generator or the GUIDE palette. For example, you may want to customize the following aspects of UIL controls:

- Color
- Size
- Behavior
- User restrictions

You customize UIL controls by creating customized subclasses of controls, or by creating customized configurations.

Creating Subclasses of UIL Controls

Every UIL control is an instance of a system-defined UIL class. For example, edit boxes are instances of the class `uil-edit-box`, and scroll areas are instances of the class `uil-scroll-area`.

To create a large number of UIL controls with a customized appearance and behavior, you can:

- Create a subclass of the system-defined UIL class for that kind of control.
- Edit this subclass to specify the appearance and behavior that you want.
- Create UIL controls from this subclass. Each of these UIL controls will have the customized appearance.

For information about how to create customized subclasses, see [Creating Custom UIL Subclasses](#).

Applying Customized Configurations to UIL Controls

The color or colors of a UIL control is specified by a G2 object known as a configuration. Each class of UIL controls has a default configuration.

You can customize the colors of all UIL controls of a particular class by editing the configuration that is applied to those controls. You can also create customized configurations and apply them to individual UIL controls.

For information about how to create customized UIL controls, see [Creating Custom UIL Subclasses](#).

UIL Methods, Actions, and Callbacks

GUIDE uses three specialized kinds of procedures to perform operations required for creating and using a GUIDE user interface. These procedures are called **UIL methods**, **actions**, and **callbacks**. GUIDE/UIL provides an extensive set of system-defined methods, actions, and callbacks.

When you create a user interface, GUIDE automatically ensures that the interface references an appropriate system-defined procedure for every common operation. You can create and use a GUIDE user interface without modifying the set of system-defined methods, actions, and callbacks that the interface references by default.

However, for some purposes, your application may need to use customized methods, actions, or callbacks to perform specialized processing. For detailed information about how to create and use customized methods, actions, and callbacks, see [Methods, Actions, and Callbacks](#).

Getting Started

Describes preliminary steps for starting and running GUIDE, and explains how to avoid problems that can prevent you from running or saving your GUIDE application.

Introduction	26
Installing GUIDE	26
The GUIDE/UII Module Hierarchy	28
Removing Unneeded GUIDE Modules from an Application	31
Setting G2 Minimum Scheduling Parameter	32
Starting GUIDE	32
Reinitializing GUIDE	32
Choosing a User Mode	32
Using the GUIDE Menu Bar	33
Resetting the GUIDE Editor	34
Enabling and Disabling GUIDE/UII User Menu Choices	34
Using GFR Startup Objects	36
Making UII Controls Permanent	37
Printing GUIDE Workspaces	37
Suggestions and Cautions	39



Introduction

The G2 User Interface Developer's Environment (GUIDE) is a knowledge base (KB) module, whose name is `guide`. All components of GUIDE are identified by either the public `gui-` prefix or private `_gui-` prefix.

Installing GUIDE

You install GUIDE by merging it into any modularized knowledge base. When you merge GUIDE, its required modules are automatically loaded into G2.

The filename of GUIDE is `guide.kb`. The default location of this KB is the `utils` subdirectory of the `kbs` directory under the `g2` directory.

How you merge GUIDE into a KB depends on whether your KB already has a version of GUIDE.

Merging GUIDE into Your KB for the First Time

To merge GUIDE into a knowledge base (KB) that has never included GUIDE:

- 1 Pause your KB.
- 2 Choose Main Menu > Merge KB to display the Load KB workspace.
The merge in this KB option is enabled on the workspace.
- 3 Specify the location of the `guide.kb` file and click End.

Tip When merging GXL, let G2 resolve conflicts by enabling the **automatically resolve conflicts** option.

When you merge GUIDE into your KB, it is not a required module unless it is specified in the Module Information table of your KB.

To make GUIDE a required module:

- 1 Choose Main Menu > System Tables > Module Information.
- 2 Specify GUIDE as a directly required module of either:
 - The module for which you are creating a GUIDE interface, or
 - The top-level module of your application.

For more information on merging KBs and making a KB a required module, see the *G2 Reference Manual*.

Merging GUIDE into a KB with an Earlier Version

To merge GUIDE into a KB that has a previous version of GUIDE:

- 1 Move your current GUIDE modules from the directory that contains your application into a separate directory.
- 2 Place this version of the GUIDE modules into the directory that contains your application.
- 3 Pause your KB.
- 4 Choose Main Menu > Merge KB to display the Load KB workspace.
The merge in this KB option is enabled on the workspace.
- 5 Specify the location of the guide.kb file and click End.

Tip When merging GXL, let G2 resolve conflicts by enabling the **automatically resolve conflicts** option.

- 6 Restart the KB.
- 7 Choose Main Menu > Save KB to save your application modules.

The next time you load these modules, it will not be necessary to resolve conflicts.

For information on upgrading GUIDE applications, see [Upgrading GUIDE Applications](#).

Verifying Your Version of GUIDE

To verify that the version of GUIDE/UIIL that you have merged is the one that you want, you can open the Guide Version Information bin, which contains a version information object specifying the version number of the currently loaded GUIDE/UIIL.

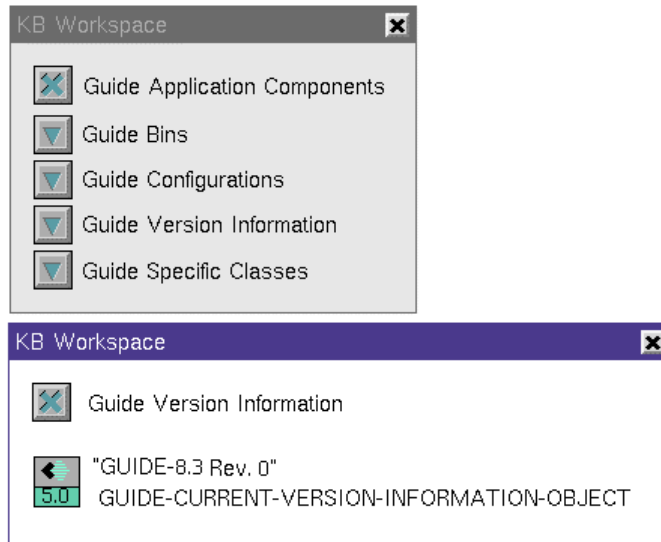
To open the Guide Version Information bin:

- 1 Choose Main Menu > Inspect and enter:

```
go to guide-storage-bins
```

The Guide Application Components bin appears.

- Click the Guide Version Information button to display the Guide Version Information bin. For example:



License Requirements

You can build and run a GUIDE user interface under any G2 license. GUIDE/UIL also works with all licenses.

The GUIDE/UIL Module Hierarchy

When you merge GUIDE/UIL into your KB, its required modules are automatically loaded into G2. The following table describes these modules:

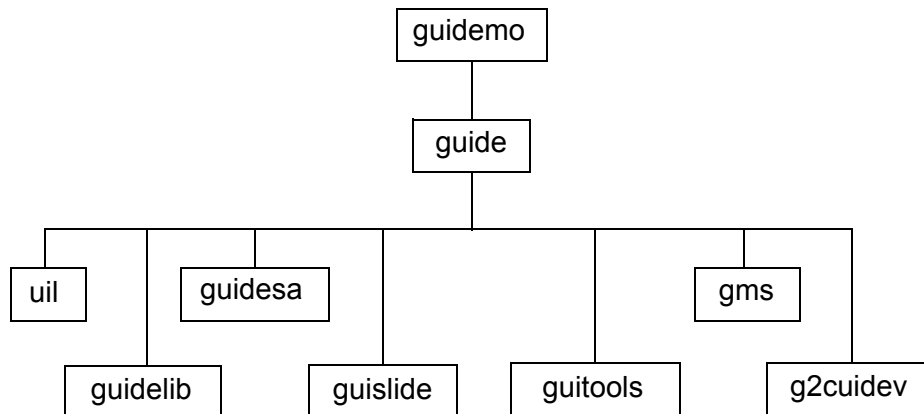
Module	File Name	Contents
g2cuidev	<i>g2cuidev.kb</i>	Definitions and API support for the G2 Developer's Interface tools. For information about these tools, see the <i>G2 Developer's Interface User's Guide</i> .
g2uiprnt	<i>g2uiprnt.kb</i>	The print workspace dialog. You can customize printer selection, papersize, orientation, margins, and other print options. For information on using its features, see Printing GUIDE Workspaces .

Module	File Name	Contents
gfr	<i>gfr.kb</i>	Definitions and API support for the G2 Foundation Resources (GFR) utility.
gms	<i>gms.kb</i>	The G2 Menu System. For information about GMS, see the <i>G2 Menu System User's Guide</i> .
gold	<i>gold.kb</i>	The G2 Online Documentation (GOLD) system. For information about GOLD, see <i>G2 OnLine Documentation User's Guide</i> .
guicolor	<i>guicolor.kb</i>	The color selection dialog.
guidata	<i>guidata.kb</i>	Support for editing attributes configured as lists or arrays.
guide	<i>guide.kb</i>	A top-level module that requires modules containing all GUIDE editors and the front-end to all UIL objects.
guidelib	<i>guidlib.kb</i>	The standard dialogs for editing attributes of UIL controls.
guidemo	<i>guidemo.kb</i>	Online examples and tutorial.
guidesa	<i>guidesa.kb</i>	class definitions for Scroll Areas.
guigfr	<i>guigfr.kb</i>	A graphical user interface and supporting API for the internationalization of dialogs.
guimove	<i>guimove.kb</i>	A move dialog, in which you can make adjustments to the X and Y positions of any G2 object.
guislide	<i>guidslide.kb</i>	Class definitions for Slider objects (<i>uilslide</i>) and an API to these classes.
guitools	<i>guitools.kb</i>	Includes the modules <i>guicolor</i> , <i>guimove</i> , <i>guigfr</i> , and <i>guidata</i> .
sys-mod	<i>sys-mod.kb</i>	The library of G2 system procedures.
uil	<i>uil.kb</i>	General API to all UIL objects. The UIL module is required for use of a GUIDE user interface.

Module	File Name	Contents
uilcombo	<i>uilcombo.kb</i>	Class definitions for combo boxes and an API to these classes.
uildefs	<i>uildefs.kb</i>	Definitions for UIL objects.
uilib	<i>uilib.kb</i>	The UIL machinery not contained in <i>uilsa</i> or <i>uilslide</i> . uilib depends upon the GFR KB for its internationalization. For information about internationalization, see the <i>G2 Foundation Resources Reference Manual</i> .
uilsa	<i>uilsa.kb</i>	Internal support for Scroll Areas.
uilslide	<i>uilslide.kb</i>	Internal support for Slider objects.
uiltdlg	<i>uiltdlg.kb</i>	Internal support for tab dialogs.
uilroot	<i>uilroot.kb</i>	Definitions and API support for navigation buttons.

Note These module dependencies are subject to change in future versions of GUIDE/UIL.

This is the module hierarchy:



Module Support for Navigation Buttons

Support for navigation buttons is provided by the modules `uil` and `uilroot`.

The `uil` module supports full navigation button functionality.

The `uilroot` module supports the creation, configuration, activation and deletion of navigation buttons, but does not support other features, such as labels. Your G2 application can use only `uilroot` if it does not require that navigation buttons have labels.

When GUIDE is loaded, you can create navigation buttons by cloning them from the G2 GUIDE palette. When GUIDE is not loaded, you can create navigation buttons by selecting **New Object** from the **KB Workspace** menu.

Removing Unneeded GUIDE Modules from an Application

GUIDE/UIL is highly modularized, enabling you to deliver your deployed applications with only those parts of GUIDE/UIL that your applications require.

After you develop a user interface using GUIDE/UIL, you can remove the GUIDE modules that your KB no longer needs by selecting the following from the GUIDE menu bar:

Tools > Remove GUIDE Application

This command removes all modules used by GUIDE that are not directly required by other modules in your KB. Thus, it is important that you correctly modularize your KB before you select this menu choice.

For example, if your application makes use of dialogs and most of the other UIL components, your module must list `uil` as one of its directly required modules. However, if you application uses only scroll areas, then `uilsa` is the only module that it requires.

Choosing **Remove GUIDE Application** invokes the procedure `uil-remove-guide-application()`. This procedure is public, and you can invoke it directly from your GUIDE/UIL application. For information about this procedure, see the *G2 GUIDE/UIL Procedures Reference Manual*.

You can also remove unneeded GUIDE modules individually, using the **G2 Delete Module** command. However, if you change the directly required module list of any GUIDE or UIL module, be sure to save that module before you load your application again. If you do not save the module before you reload your application, the module will reload the same modules that you removed from its list of directly required modules.

You can merge GUIDE development modules back into an application whenever you need to modify or add features to the GUIDE user interface.

Setting G2 Minimum Scheduling Parameter

For best performance of GUIDE/UIL, set the minimum-scheduling-interval in the Timing Parameters system table to 0.25 or less.

For information about how to set this parameter, see the *G2 Reference Manual*.

Starting GUIDE

GUIDE is started when you start your knowledge base. You can start your knowledge base by choosing **Start** or **Restart** from the G2 Main Menu.

When GUIDE is started, the G2 GUIDE/UIL initialization panel appears in the lower left corner of the screen. The initialization panel includes the version numbers for GUIDE and UIL and a changing display of status information about the initialization process.

When the G2 GUIDE/UIL initialization panel disappears, initialization is complete and you can use GUIDE.

Reinitializing GUIDE

You can reinitialize GUIDE by selecting the following from the GUIDe menu bar:

Tools > Reinitialize GUIDE

You can also use the UIL procedure `uil-initialize-guide` to initialize GUIDE.

Choosing a User Mode

Before you begin to use GUIDE to develop a user interface, change your user mode to Build mode. You can do this in either of two ways:

- Choose G2 > Change Mode > `uil-build` from the GUIDE menu bar.
- Select Build Mode in the GUIDE Control Panel. For information about how to do this, see [Using the GUIDE Control Panel](#).

When you use GUIDE in Build mode, the run-time behavior of dialogs and UIL controls is suspended so that you can edit them. In Build mode, you can access menu choices on dialogs and UIL controls by clicking on them. These menu choices enable you to perform operations on the dialogs and UIL controls such as moving, cloning, and deleting them, and accessing graphical editors through which you can edit their attributes.

You can also create a user interface in Administrator mode, which gives you the same access to the menus of dialogs and UIL controls that Build mode gives you. However, Administrator mode lacks restrictions that allow you to use the G2

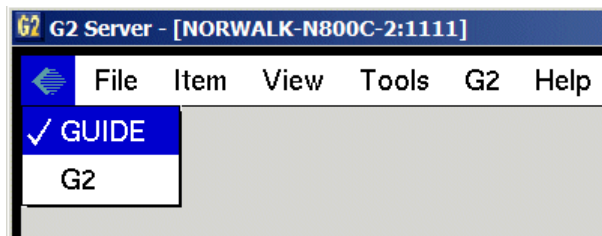
GUIDE palette normally. Use Build mode when you create a user interface unless you have a specific reason to use Administrator mode.

You can simulate the run-time behavior of UIL controls by changing your user mode to User mode. In User mode, you enter text into edit boxes, selected and deselect buttons, and scroll up and down in scroll areas. You cannot edit dialogs or UIL controls in User mode.

Using the GUIDE Menu Bar

You can perform many GUIDE operations, such as creating objects and displaying dialogs, through the GUIDE menu bar displayed across the top of the window where you are running G2.

To include all GUIDE-specific menu options in the GUIDE menu bar, click the Gensym logo in the upper-left corner of the window and choose GUIDE from the menu:



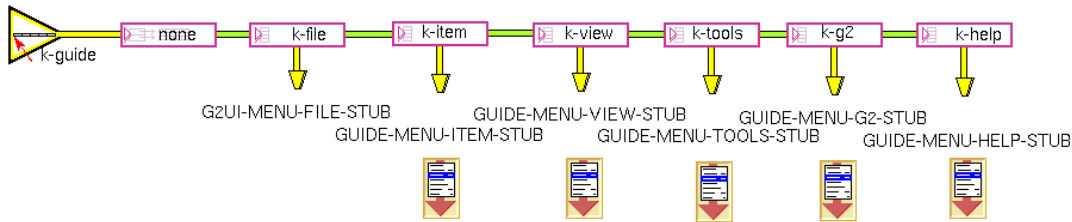
The menu bar and some new dialogs reached through the menu bar, depend on the following G2 Developer's Interface modules: `g2uimenu`, `g2uifile`, `g2uiprint`, and `g2uitree`. these modules are required by the module `g2cuidev`. You may find that you need to remove these KBs from GUIDE when you develop your application.

Removing `g2cuidev.kb`

If you edit GUIDE's module hierarchy not to include `g2cuidev.kb`, you should also disable GUIDE's menu bar to prevent startup errors. You can reach that menu bar with this Inspect command:

```
go to guide-menu-resource
```

On this workspace, you will see a triangular template labeled **k-guide**. Disable that item, and GUIDE will not attempt to show its menu bar upon startup.



Reusing the GUIDE Submenus

All of the GUIDE submenus are usable by other menus. The GUIDE menu bar template can be reached by using this Inspect command:

```
go to guide-menu-resource
```

The resulting workspace contains a copy of the G2 menu bar, with new, extended submenus in place of the ones used by the G2 menu bar (except for the File menu, which is untouched). All these GUIDE-specific menus can be reused by customers' menu bars, just as the GUIDE menu bar itself reuses the File submenu. For more information about copying and reusing menus, see the *G2 Developer's Interface User's Guide*.

Resetting the GUIDE Editor

You may want to reset GUIDE if you encounter problems during an editing session. Resetting GUIDE returns it to its initial state at the beginning of your editing session.

To reset GUIDE, select the following choice from the GUIDE menu bar:

```
Tools > Reset Editor
```

Enabling and Disabling GUIDE/UIL User Menu Choices

For certain basic operations, such as cloning or deleting, UIL objects have separate versions of the G2 user menu choices.

The UIL user menu choices end in a period (.), to distinguish them from the corresponding G2 user menu choices. For example, the UIL user menu choice for cloning is **clone.**, which except for the period is identical to the **clone** G2 user menu choice for this operation.

If both the G2 and the UIL versions of a user menu choice are present in the menu of a UIL object, always use the UIL version to perform the operation on the object.

Enabling and Disabling User Menu Choices for All Objects

To enable user menu choices on UIL objects, toggle on the **Enable User Menu Choices** toggle button in the **More Options** palette, which you can access through the G2 GUIDE palette.

Enabling and Disabling User Menu Choices for Particular Modules

You can enable or disable user menu choices for particular GUIDE/UIL modules by setting the following logical parameters to **true** or **false**:

```
guide-user-menu-choices-active
g2uiprnt-user-menu-choices-active
guigfr-user-menu-choices-active
g2uifile-user-menu-choices-active
guidata-user-menu-choices-active
guimove-user-menu-choices-active
guislide-user-menu-choices-active
guidesa-user-menu-choices-active
uil-user-menu-choices-active
uilcombo-user-menu-choices-active
uilslide-user-menu-choices-active
uilsa-user-menu-choices-active
uiltldg-user-menu-choices-active
uilib-user-menu-choices-for-non-uil-objects-active
uilib-user-menu-choices-for-uil-objects-active
```

By default, all these logical parameters are set to **true**. You disable user menu choices for the objects in a particular module by setting the corresponding logical parameter to **false**. For example, to disable user menu choices on Scroll Areas, set `uilsa-user-menu-choices-active` to **false**.

Note that there are two logical parameters affecting user menu choices of the `uilib` module:

- `uilib-user-menu-choices-for-non-uil-objects-active` **true** enables `uilib` user menu choices on all objects; **false** disables `uilib` user menu choices for non-UIL objects only.
- `uilib-user-menu-choices-for-uil-objects-active` **true** enables `uilib` user menu choices on all objects; **false** disables `uilib` user menu choices for all objects.

You can set the values of these logical parameters in an initially rule, a startup procedure, or any other method that is suitable for your requirements. For information about how to set the values of parameters, see the *G2 Reference Manual*.

Using GFR Startup Objects

Versions of GUIDE/UIL earlier than Version 4.1 depended upon initially rules to initialize. This worked without problem unless other KBs loaded with GUIDE/UIL had their own initialization procedures that could not run until GUIDE/UIL finished initializing.

Because higher-level modules depend on the services of lower-level modules, startup precedence becomes an issue in a multiple-module system. To ensure correct operation, lower-level modules should be activated before higher-level modules. GFR provides a startup facility that ensures that module initialization activities proceed from low-level modules in an organized fashion up the module hierarchy to higher-level modules. GFR also provides for organized dispatch to multiple warmboot procedures following the loading of your KB from a snapshot file.

The GFR KB provides a `gfr-startup-object` class to solve problems such as these. Startup objects are assigned to modules (no more than one per module, though some modules do not have any), and they contain references to procedures that should be run when G2 is started. The order in which these procedures are run, if there are multiple startup objects loaded at one time, is determined by the module hierarchy: the lowest-level modules' startup procedures are run first, on up the hierarchy to the highest-level modules' startup procedures last.

Startup objects can make it easier for KB developers to integrate GUIDE/UIL into their KBs. A startup object can be created for the modules that depend on GUIDE/UIL; the startup procedures specified in those startup objects are guaranteed to run after the GUIDE/UIL initialization has finished, with no need to check for its completion. For more information on GFR startup objects, see the *G2 Foundation Resources Reference Manual*.

Note GUIDE's startup object is in the `uilib` module, not in the `guide` module. It handles initialization for all of GUIDE if the `guide` module is loaded.

Making UIL Controls Permanent

By default, UIL controls are transient, and do not persist if you reset or restart your KB.

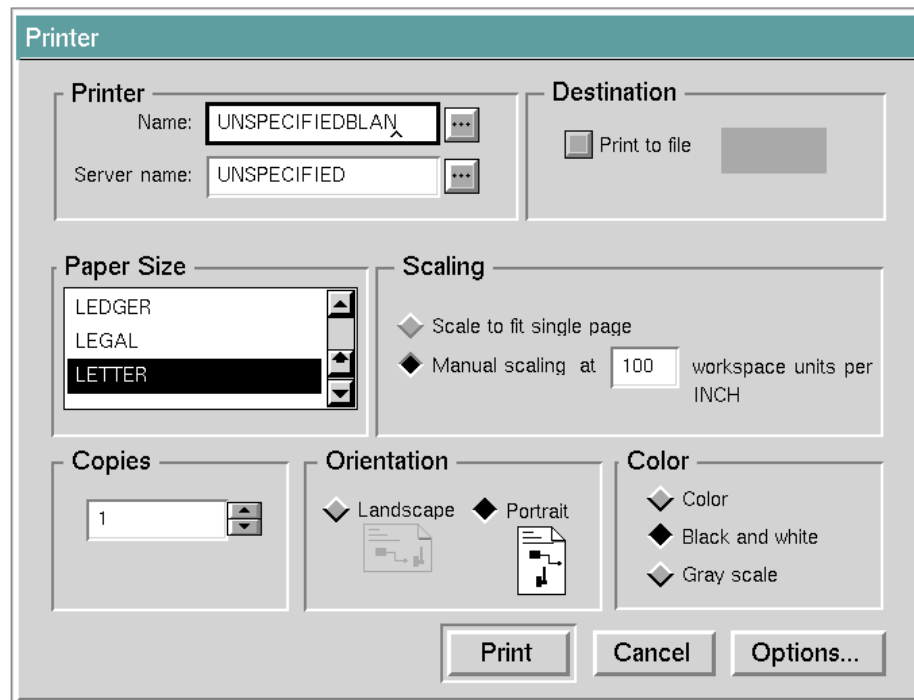
If you need to make a UIL control permanent, use the UIL procedure `uil-make-grobj-permanent()`. For information about this procedure, see the *G2 GUIDE/UIL*

Procedures Reference Manual.

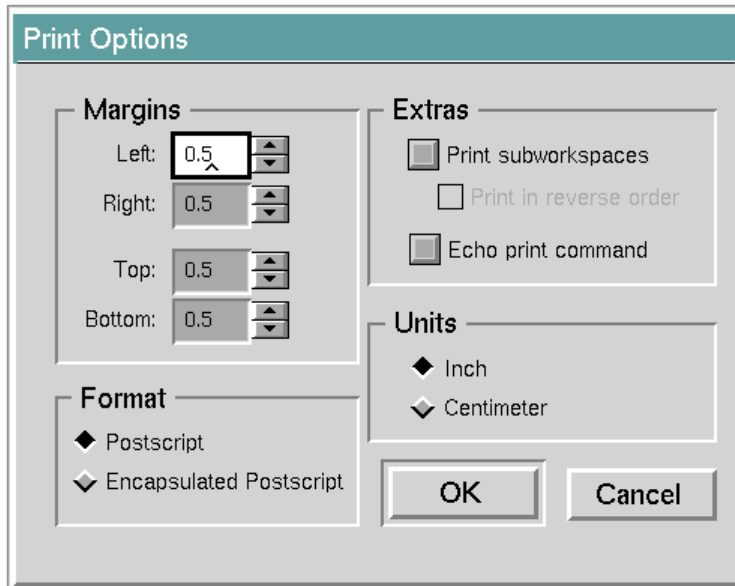
Caution Do not attempt to use the G2 action make permanent to make UIL controls permanent.

Printing GUIDE Workspaces

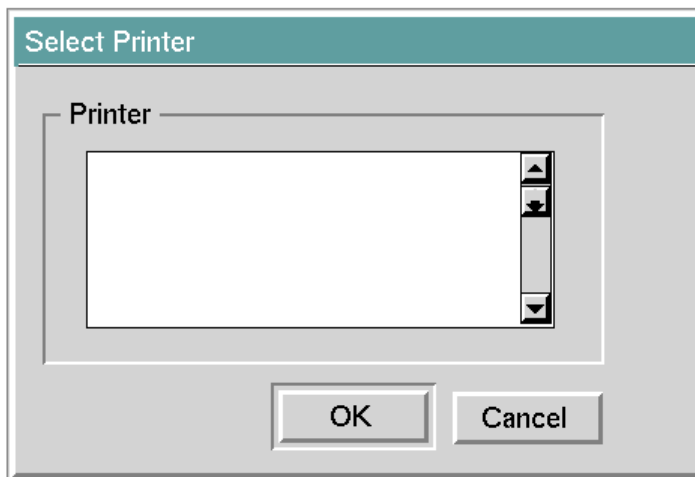
You can print GUIDE workspaces by choosing KB Workspace > Print from the menu of the workspace that you want to print. The following dialog appears:



To display additional print options, click the Options button in the Printer dialog. The Print Options dialog appears:

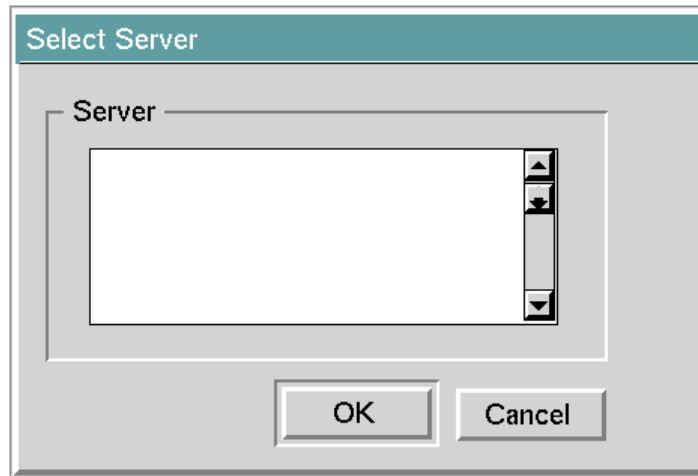


To select the printer on which you want to print your workspace, click the button to the right of the Name field in the Printer dialog. The Select Printer dialog appears:



In the Select Printer dialog, a list of the currently configured printers appears in the scroll area. Select the printer that you want to use and click the OK button.

To select the server that you want to use, click the button to the right of the Server Name field in the Printer dialog. The Select Server dialog appears:



In the Select Server dialog, a list of the currently connected servers appears in the scroll area. Select the server that you want to use and click the OK button.

Suggestions and Cautions

This section lists easy ways to avoid problems that can prevent you from running or saving your GUIDE application.

- **Always use GUIDE in a modularized KB.**

The GUIDE dialog system works only in modularized KBs. Do not attempt to use GUIDE in an unmodularized KB. For information about modularizing KBs, see the *G2 Reference Manual*.

- **Always name the top-level module of the application that uses GUIDE.**

GUIDE incorporates the name of the top-level module into class definitions and IDs that it generates automatically for internal use. GUIDE cannot generate these class names and IDs if the top-level module has no name.

- **Always merge GUIDE.**

Always Merge GUIDE or any module that requires GUIDE or UIL. Do not Load these modules.

- **Restart after pause to merge in GUIDE.**

Always restart your application after you pause to merge in the GUIDE module or any module requiring GUIDE or UIL.

Do not reset your application after you pause G2 if dialogs are displayed. This causes you to lose the dialogs. You can recover the dialogs by merging your GUIDE application into your KB again.

- **Use GUIDE graphical editors to edit object attributes.**

For most purposes, it is good practice to use GUIDE graphical editors to edit the attribute values of dialogs or UIL controls.

It is possible to edit attribute values directly in the attribute tables. However, if you edit the IDs or other attributes of dialogs and GUIDE controls directly in attribute tables, you can introduce inconsistencies that prevent your application from running properly.

- **Use GUIDE clone. menu choice to copy workspaces.**

Always use the GUIDE clone. menu choice to copy workspaces. Problems occur when you use the G2 clone menu choice to clone workspaces, especially workspaces that contain a combination of UIL objects and non-UIL objects.

- **Do not use Operate On Area menu choice to clone or delete UIL objects.**

Many UIL objects are composed of pieces held together by named relations or internal lists. The Operate On Area menu choice is not designed to operate on these features. Therefore, Do not use the Operate On Area menu choice to clone or delete UIL objects.

Using Operate On Area to clone or delete objects produces UIL objects that do not work properly. However, you can use Operate On Area to move and align UIL objects.

- **Always use the user menu choices of UIL objects.**

User menu choices end with a period (.). For example, always clone and delete UIL objects using the clone. and delete. menu choices.

- **Close subworkspaces of dialogs before resetting GUIDE application.**

Copy dialogs are lost if their subworkspaces are displayed when you reset your application. To keep the copy dialogs, close them by clicking OK or Cancel before you reset your application.

- **Remove all UIL objects from a workspace before cloning or deleting the workspace.**

GUIDE does not properly clone or delete the UIL objects on a workspace when you clone or delete the workspace.

Before you delete dialogs, delete copies of the dialogs from the dialog bin by choosing **delete copies** from the dialog menus. If you do not delete the copies, the copies remain in the bins after you delete the dialogs from the workspace.

- **Do not make master dialogs proprietary.**

Master dialogs are templates that GUIDE clones to create the dialogs that users see and use. Do not make master dialogs proprietary. Proprietary dialogs cannot be cloned.

- **Do not make workspaces with master dialogs proprietary.**

Do not make workspaces proprietary if they contain master dialogs or other UIL objects. Proprietary workspaces are restricted to prevent cloning, deleting, and certain kinds of text access. GUIDE dialogs fail to function when placed on proprietary workspaces, because they make extensive use of clone and delete operations.

However, you can place navigation buttons on proprietary workspaces.

- **Do not use clone workspace or delete workspace menu choices of a dialog subworkspace.**

Instead, use the menu choices for cloning or deleting the master dialog, not the subworkspace of the master dialog.

Be careful not to delete the subworkspaces of master dialogs. Doing so makes a master dialog unusable, because the subworkspace contains the dialog that users see and use.

- **Check the value of uil-initialization-status parameter.**

When the initialization of GUIDE is complete, GUIDE sets the value of the symbolic parameter `uil-initialization-status` to `initialized`. The initial value of this symbolic parameter is `uninitialized`.

Applications build on GUIDE should verify that the value of `uil-initialization-status` is set to `initialized` before attempting to use GUIDE.

- **Do not use the string scroll-area in module names.**

Module names that include the string `scroll-area` can conflict with names that GUIDE generates automatically for internal use.

- **Limit number of named UIL controls as required to conserve memory.**

Names of UIL controls are stored as symbols, and G2 does not reclaim symbols. For this reason, you may want to limit number of named UIL controls that you add to your user interface. Names are optional for all UIL

controls except radio buttons and check buttons, which are automatically named by the selection boxes that contain them. Thus, you may want to limit the number of UIL controls to which you give optional names, or the number of selection boxes that you add to your user interface.

Creating a User Interface

Chapter 3: Generating Master Dialogs

Describes how to use the GUIDE Dialog Generator to generate a master dialog for viewing and editing attributes of a particular user-defined class.

Chapter 4: Building Master Dialogs

Describes how to use the G2 GUIDE Palette and other GUIDE tools to build customized dialogs for viewing and editing the class-specific attributes of user-defined classes.

Chapter 5: Using UIL Controls on a Workspace

Illustrates several ways to use UIL controls on a workspace, without incorporating them into a dialog.

Chapter 6: Customizing Dialogs

Describes how to edit and customize GUIDE dialogs.

Chapter 7: Launching Dialogs

Describes how to launch a dialog from an action button, a user-menu choice, a user-defined procedure, a push button, or a rule.

Chapter 8: System-Defined Dialogs

Describes the predefined message, confirmation, query, and notification dialogs and delay notification icon.

Generating Master Dialogs

Describes how to use the GUIDE Dialog Generator to generate a master dialog for viewing and editing attributes of a particular user-defined class.

Introduction **45**

Using the GUIDE Dialog Generator **46**

Steps for Generating a Master Dialog **48**

Launching Generated Dialogs from Push Buttons **59**

Editing Generated Dialogs **61**



Introduction

Master dialogs serve as templates for dialogs that users see and use when they run a G2 application.

For information about how to build more fully customized master dialogs, using the G2 GUIDE Palette, see [Building Master Dialogs](#).

Note You cannot use the push buttons and other UIL controls on a master dialog. To test a master dialog and the UIL controls that it contains, you must launch a copy dialog of the master dialog. For information about how to do this, see [Launching Dialogs](#).

Using the GUIDE Dialog Generator

You can start the GUIDE Dialog Generator by choosing **generate dialog** from the menu of the user-defined class for which you want to generate the dialog.

You can also start the GUIDE Dialog Generator by selecting **Tools > Dialog Generator** from the GUIDE menu bar.

You can also start the GUIDE Dialog Generator by clicking on the question mark (?) icon on the G2 GUIDE Palette. This opens the GUIDE Help dialog. In the GUIDE Help dialog, click the button labeled **GUIDE Dialog Generator**.

When you generate a dialog, you can either allow GUIDE to choose, by default, the classes of UIL controls that best represent the different class-specific attributes, or you can override GUIDE's default choices and use non-default classes of UIL controls to represent some or all of the attributes.

Master Dialogs with Default UIL Controls

When you generate a master dialog with default UIL controls, you select the class-specific attributes for which you want to add UIL controls to the dialog. To represent each attribute, GUIDE uses a UIL control of the class that is best suited for representing data of the type associated with that attribute.

The following table summarizes the kinds of UIL controls that GUIDE uses by default to represent attributes with different data types.

Default UIL Controls for Attribute Data Types

Data Type of Class-specific Attribute	Example of Class-specific Attribute Definition	Default UIL Control for an Attribute of this Data Type
untyped	color color initially is red	Edit box
truth value	color is a truth-value, initially is true	Icon toggle button. The button's default toggle state reflects initially is value in attribute declaration.
symbol (with two possible values)	color is a symbol, has values red or white, initially is red	Icon toggle button. The button's default toggle state reflects initially is value in attribute declaration.

Default UIL Controls for Attribute Data Types

Data Type of Class-specific Attribute	Example of Class-specific Attribute Definition	Default UIL Control for an Attribute of this Data Type
symbol (with more than two possible values)	color is a symbol, has values red, white, or blue, initially is red	Radio box. By default, the radio button representing the initially is value in an attribute declaration is selected.
list, array	color is an instance of a text-array	Scroll area. Elements of the list or array are represented by message objects in the scroll area.
embedded object	color is an instance of a my-user-defined-class	Push button. You can edit this push button to make it open a dialog for editing the attributes of the embedded object, if such a dialog exists.
other data types	color is a text, initially is "red"	Edit box

The GUIDE Dialog Generator also automatically generates a user menu choice for launching the master dialog. The user menu choice is added to every object of the class for which you generated the master dialog.

Master Dialogs with non-Default UIL Controls

Generating a master dialog with non-default UIL controls is similar to generating a dialog with default UIL controls. You select the class-specific attributes for which you want to add UIL controls to the dialog, and GUIDE adds UIL controls to represent these attributes.

However, you can override GUIDE's default choices of UIL classes and use UIL controls of other classes to represent some or all of the attributes. When you generate a master dialog with non-default UIL controls, you can also:

- Generate user menu choices and push buttons for launching the dialog.
- Generate rules for updating the UIL controls on the dialog. You can generate two kinds of rules:
 - Rules for updating the display of an individual UIL control on the dialog when its corresponding attribute receives a value.
 - A rule for updating all UIL controls on a dialog when a user moves the initiating object of the dialog. This rule can be useful for tracking the X and Y coordinates of the initiating object.

Steps for Generating a Master Dialog

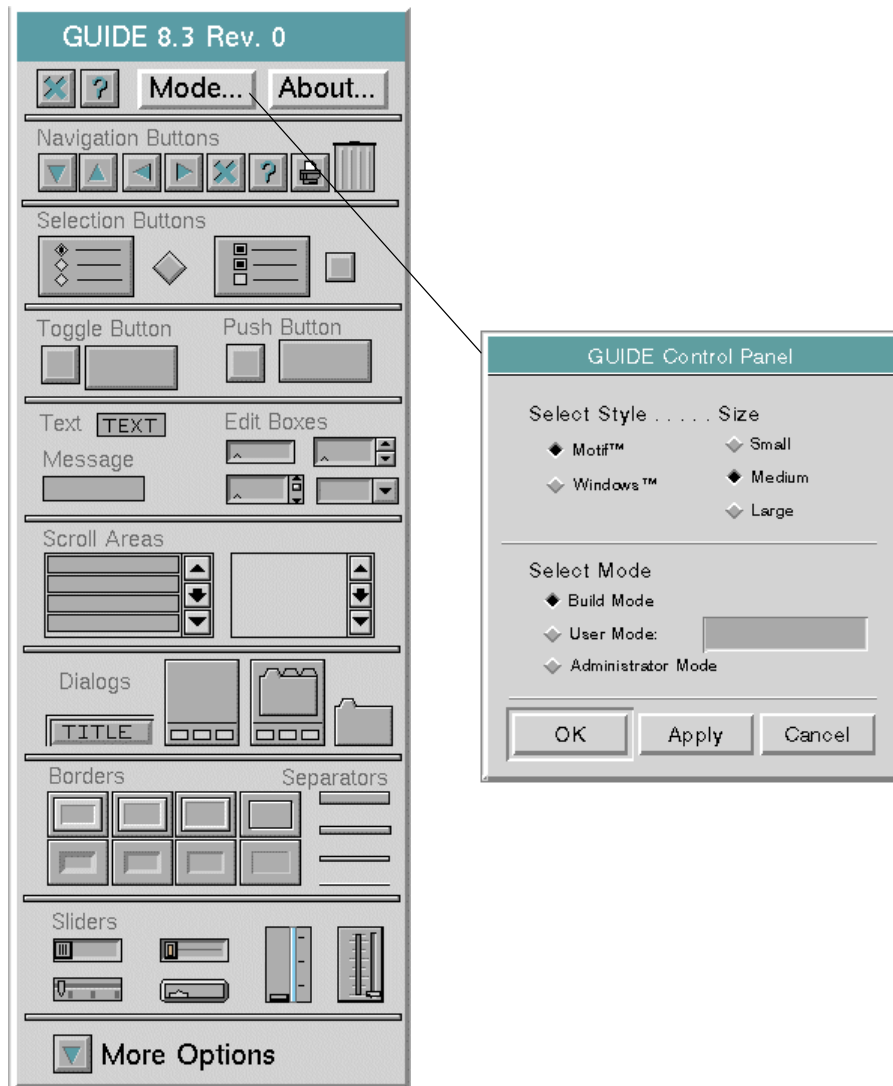
To generate a master dialog (with default or non-default UIL controls):

- 1 Change your user mode to either Build mode or Administrator mode.

Build mode is the recommended mode for constructing a user interface. However, you can use all the features of GUIDE in either Build mode or Administrator mode.

Note You may find it necessary to change to Administrator mode if certain modules in your application can be accessed only by users who are running GUIDE in Administrator mode.

To change your mode, click the Mode button in the G2 GUIDE Palette. Clicking the Mode button opens the GUIDE Control Panel:



In the GUIDE Control Panel, select Build Mode or Administrator Mode, and click OK.

Note If you are currently running GUIDE in Administrator mode, you can open the GUIDE Control Panel dialog by clicking the Mode button and choosing select from the Mode button's menu.

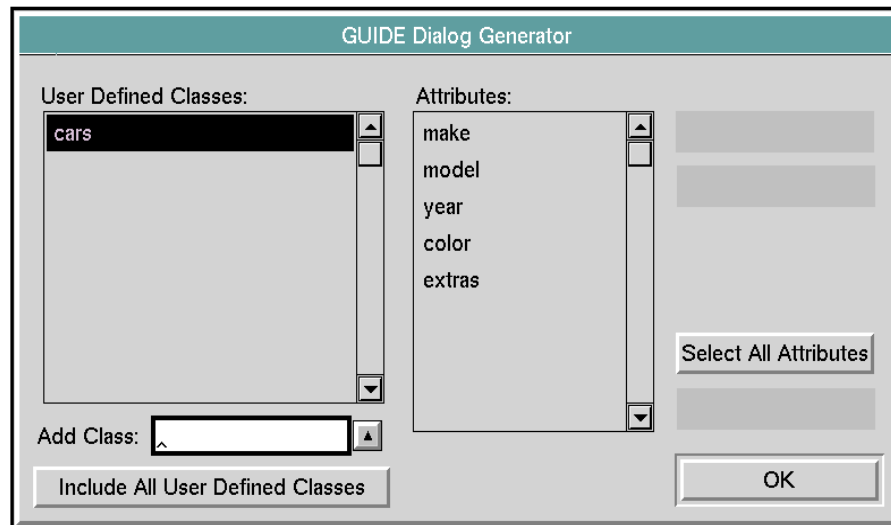
- 2 Open the attribute table of the user-defined class to make sure that it has values for:
 - Class-name
 - Direct-superior-classes
 - Class-specific-attributes (This slot defines the attributes for which GUIDE can add a UIL control to the dialog.)

If any of these slots are empty, fill them with appropriate values.

- 3 Choose **generate dialog** from the menu of the user-defined class.

The GUIDE Dialog Generator appears.

For example, suppose that you choose **generate dialog** from the menu of a user-defined class named **cars**, which has the class-specific attributes **make**, **model**, **year**, **color**, and **extras**. The GUIDE Dialog Generator looks like this:



- 4 In the User Defined Classes column, select the name of the user-defined class for which you want to generate a dialog.

If you start the GUIDE Dialog Generator by choosing **generate dialog** from the menu of a user-defined class, the name of the user-defined class appears selected in the User Defined Classes column, as in the figure above.

If you start the GUIDE Dialog Generator from the GUIDE Help dialog, the User Defined Classes column appears empty. You can select the class for which you want to generate a dialog by entering its name in the Add Class field and then clicking the up arrow. The name of the class appears selected in the User Defined Classes column.

You can also click the Include All User Defined Classes button to display the names of all user-defined classes in the User Defined Classes column. You can then select the name of the class for which you want to generate a dialog.

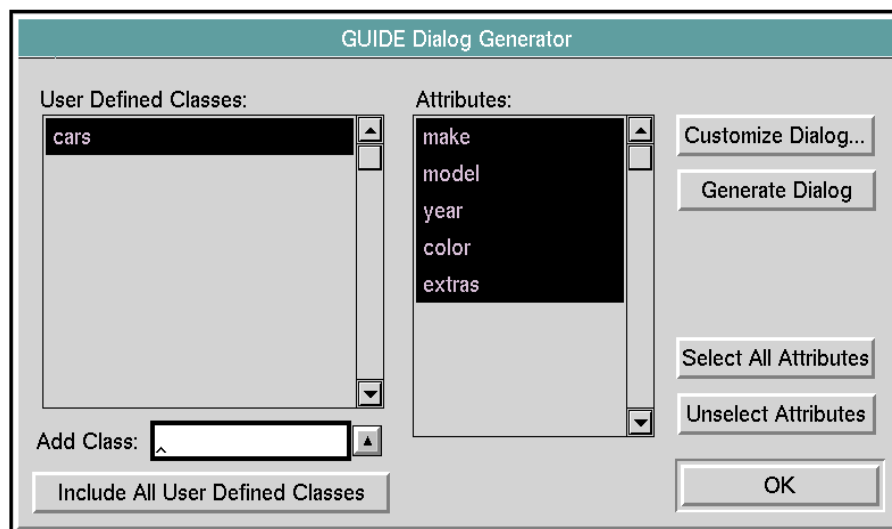
When a class in the User Defined Classes column is selected, the class-specific attributes of that class are displayed in the Attributes column.

- 5 In the Attributes column, select the class-specific attributes that you want users to be able to view and edit through the generated dialog. GUIDE adds a UIL control to the dialog for each attribute that you select.

You can select individual attributes by clicking on them. You can select all attributes by clicking on the Select All Attributes button.

The order of the selected attributes in the Attributes column determines the order in which the UIL controls appear in the generated dialog. You can change the order of the attribute in the Attributes column by dragging them up or down.

When you select attributes, buttons labeled Customize Dialog and Generate Dialog become enabled:



At this point, you choose to generate either a master dialog with default UIL controls or with non-default UIL controls:

- Click Customize Dialog to generate a master dialog with non-default UIL controls.
- Click Generate Dialog to generate a master dialog with default UIL controls.

The following sections explain how to generate each kind of dialog.

Generating a Master Dialog with Default UIL Controls

To generate a master dialog with default UIL controls:

- 1 Click the Generate Dialog button in the GUIDE Dialog Generator.

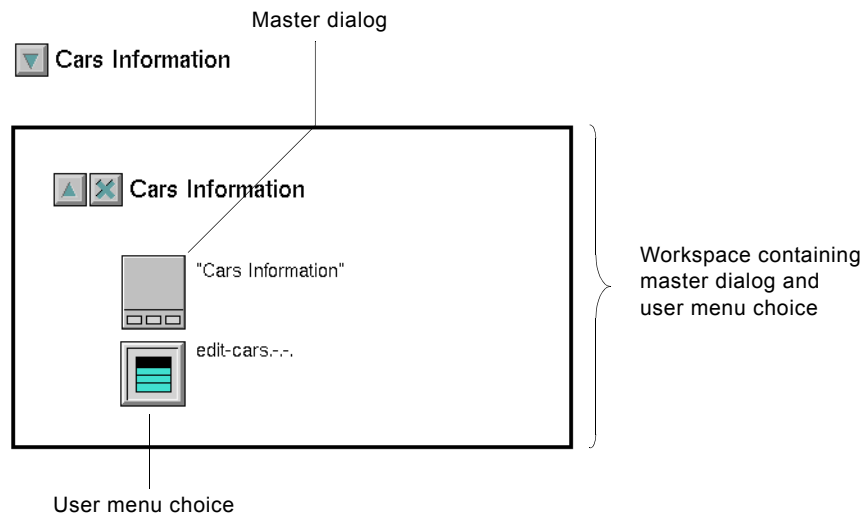
GUIDE creates a dialog and user menu choice, and places them on a workspace. GUIDE creates a navigation button that you can click to go to this workspace.

The navigation button is labeled **class information**. The dialog is named **class Information**. The user menu choice is named **edit class**. In each of these names, **class** is the name of the user-defined class for which you generated the dialog.

- 2 To see the dialog and user menu choice, open the workspace.

To open the workspace, click the **class information** navigation button.

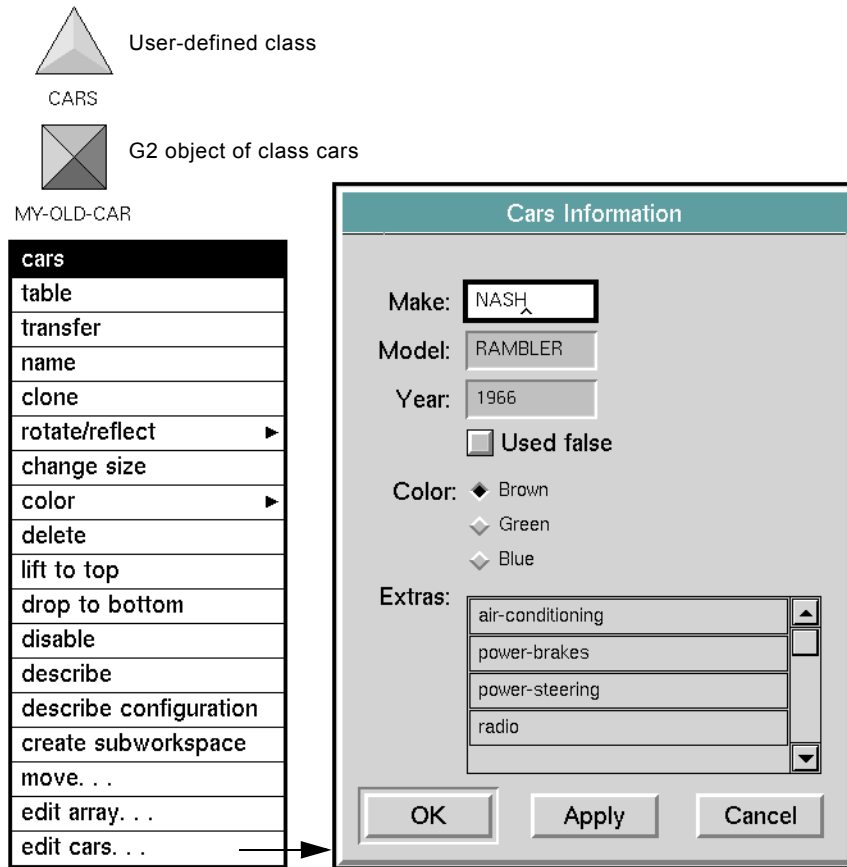
The following figure illustrates the navigation button, dialog, and user menu choice generated for the user-defined **cars** class:



- 3 Launch the dialog to verify that it has the features you want.

To launch the dialog, select **edit class** on an object of the user-defined class for which you generated the dialog.

For example, if you select **edit cars** from the menu of an object of the **cars** class, you launch the Cars Information dialog:



When the dialog is launched, each UIL control on the dialog is updated with the current value of the corresponding attribute of the G2 object for which the dialog is displayed.

Users can edit the values of the UIL controls and use standard OK, Apply, and Cancel buttons on the dialog to apply or discard their edits. When a conclude method is run on the dialog, each UIL control concludes its value to its corresponding attribute of that G2 object.

Generating a Master Dialog with Non-Default UIL Controls

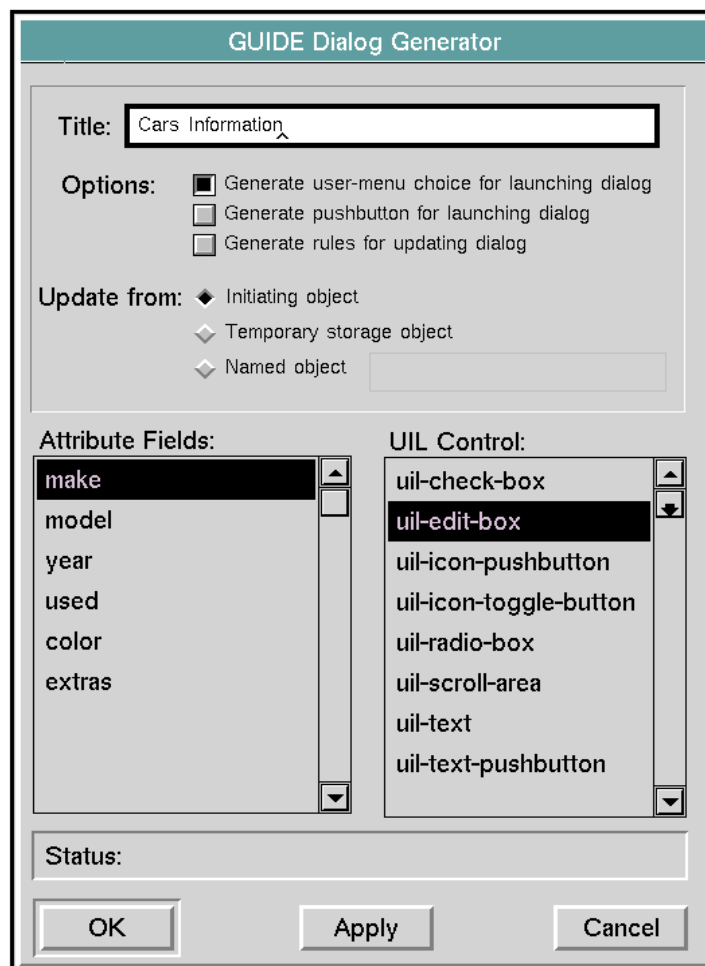
You can create a master dialog with non-default UIL controls. You can optionally create a user menu choice push button, and/or rule for launching the dialog.

To create a master dialog with non-default UIL controls:

- 1 Click the Customize Dialog button in the GUIDE Dialog Generator.

A second GUIDE Dialog Generator window appears. You specify options for the customized dialog in this window.

The following figure illustrates the GUIDE Dialog Generator when it has been opened to create a customized dialog for the user-defined cars class:



- 2 In the Title field, specify a name for the new dialog.

By default, the name of the dialog is *class information*, where *class* is the name of the user-defined class for which you are generating the dialog.

- 3 Choose the optional features that you want to generate.

To do this, select any combination of the following check buttons in the Options group:

- Generate user-menu choice for launching dialog
- Generate push button for launching dialog

If you choose to launch a generated dialog from a push button, you must specify the source and target objects and attributes for the UIL controls on the dialog. For information about how to do this, see [Launching Generated Dialogs from Push Buttons](#).

- Generate rules for updating dialog

If you select **Generate rules for updating dialog**, a dialog named **Generate Update Rules** appears when you click OK in the GUIDE Dialog Generator dialog. The **Generate Update Rules** dialog enables you to specify the particular rules that you want to generate. For information about how to use the **Generate Update Rules** dialog, see step [7](#).

- 4 Specify the source object whose attribute values are updated into the UIL controls on this dialog. To do this, select one of the following radio buttons in the Update From group:

- Initiating object

The initiating object is a G2 object that launches the dialog.

Specify **Initiating object** as the source object when you use a user-menu choice to launch the dialog. In this case, the source object is the object from whose menu a user selects **edit class-name** to launch the dialog, where *class-name* is the name of the user-defined class.

It is possible to specify **Initiating object** as the source object when you use a push button to launch the dialog. For information about how to do this, see [Launching Generated Dialogs from Push Buttons](#).

- Temporary storage object

For information about how to use temporary storage objects, see [Creating Temporary Storage Objects](#).

- Named object

This object can be any named G2 object.

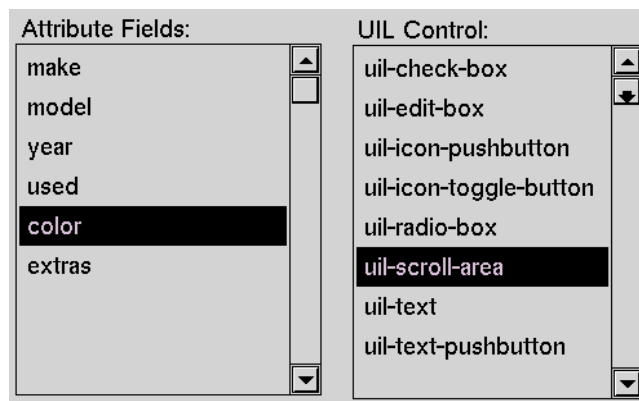
If you select **Initiating object**, the initiating object is also, by default, the target object to which the UIL controls on the dialog conclude their values. You can edit the UIL controls to specify different target objects.

If you select **Temporary storage object** or **Named object**, you must edit each UIL control to specify its target object.

For information about how to specify the source and target objects of UIL controls, see [Specifying Source and Target Objects](#).

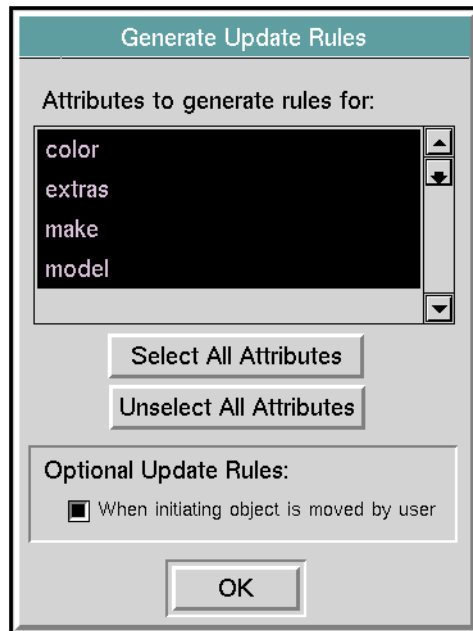
- 5 Select the kind of UIL control that you want to use in the dialog to represent each class specific attribute of the G2 object, as follows:
 - a Select an attribute in the column labeled **Attribute Fields**.
 - b Select a UIL control in the column labeled **UIL Control**. A UIL control of this class will be added to the dialog to represent the selected attribute.

For example, the following figure shows how to specify that the **color** attribute is represented in the generated dialog by a scroll area:



- c To change the kind of UIL control used to represent an attribute, first select that attribute in the **Attribute Fields** column. The UIL control currently associated with that attribute is automatically selected in the **UIL Control** column. Then, in the **UIL control** column, select the UIL control that you want to use in place of the currently selected one.
- 6 When you finish selecting options in the **GUIDE Dialog Generator** dialog, click **OK**.

- 7 If you chose to generate update rules, the Generate Update Rules dialog appears when you click OK in the GUIDE Dialog Generator:



The Generate Update Rules dialog enables you to generate two kinds of rules:

- Rules that update the state or value of individual UIL controls on the dialog whenever the corresponding attributes in the source object receive values.
- A rule that updates the state or value of every UIL control on the dialog whenever a user moves the initiating object of the dialog.

To specify which rules you want to generate, follow these steps:

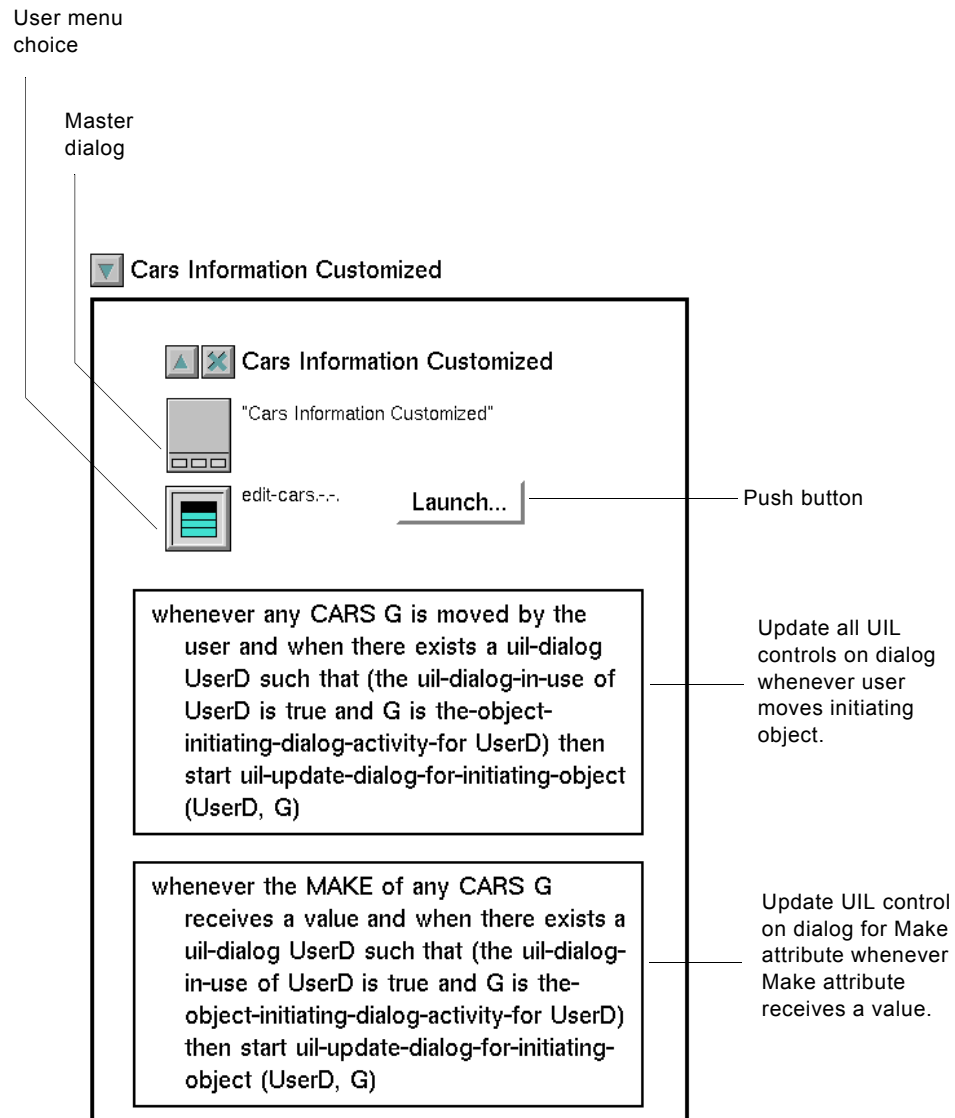
- a In the scroll area labeled Attribute to Generate Rules For, select each attribute for which you want to generate a rule. To select all attributes, click the Select All Attributes button.
 - b Select the toggle button labeled When initiating object is moved by user to generate a rule that updates all UIL controls on the dialog whenever a user moves the initiating object.
 - c Click the OK button.
- 8 Close the GUIDE Dialog Generator dialog by clicking on the OK button.

GUIDE generates the customized dialog and (optionally) a user menu choice, push button, and update rules. It places these objects on a workspace.

GUIDE creates a navigation button that you can click to go to the workspace.

- Look at the workspace to make sure that it contains the dialog and any optional features that you chose to generate.

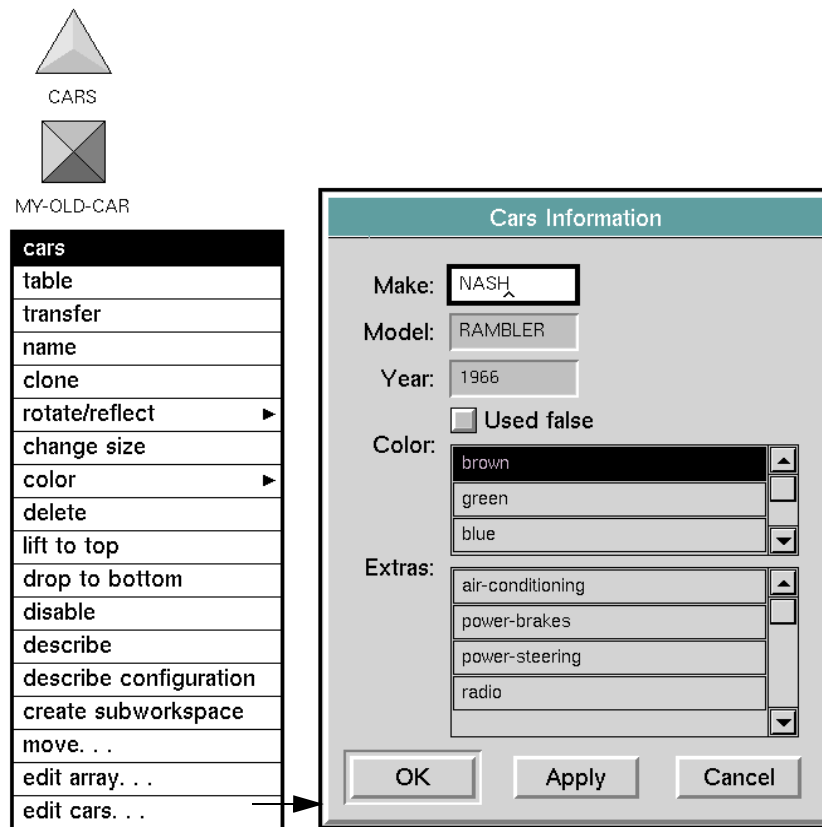
For example, the following figure illustrates a workspace containing a customized dialog, user menu choice, push button, and update rules, and the navigation button generated to open this workspace.



- Launch the customized dialog to make sure that it contains all the features that you want.

To do this, select the user menu choice `edit class` from a G2 object of the class for which you generated the customized dialog.

The following figure illustrates the customized dialog for the cars class:



Launching Generated Dialogs from Push Buttons

When you generate a dialog using the GUIDE Dialog Generator, you can also generate a push button to launch the dialog.

When you use a push button to launch a dialog, you must designate specific objects as the source objects for UIL controls on the dialog. In the GUIDE Dialog Generator, you designate a default source object for all UIL controls on the generated dialog. You do this by selecting Temporary storage object, Named object, or Initiating object under the label Update from.

The object used as the source object must have class-specific attributes from which the UIL controls on the dialog can be updated.

Updating UIL Controls from the Initiating Object When the Dialog is Launched by a Push Button

When you launch a dialog by a push button, that push button is the default initiating object of the dialog.

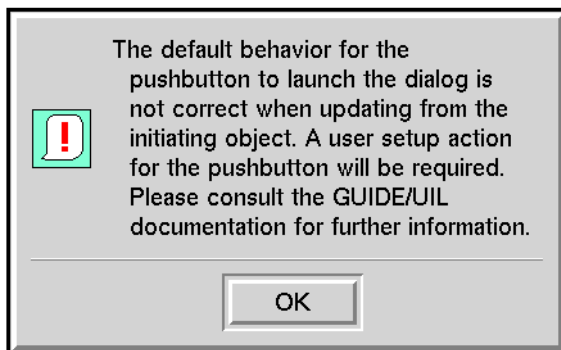
When you use to a push button to launch a dialog, updating the UIL controls on the dialog from a Temporary storage object or Named object presents no special problems. However, updating the UIL controls from the Initiating object, in this case, the push button itself, is not possible. A push button does not have class-specific attributes from which the UIL controls can be updated.

For example, if you choose Initiating object under Update from when you generate the Cars Information dialog in the example above, the source object and source attribute of the Make edit box are defined as follows:

Uil event source object	initiating-object
Uil event source attribute	make

A push button cannot be used as the source object of the Make edit box because it does not have a `make` attribute (or any other class-specific attribute) from which the Make edit box can be updated.

The GUIDE Dialog Generator reminds you that you cannot use a push button as a source object. If you select the Generate pushbutton for launching dialog option and you also specify Initiating object under Update From, you see the following message when you click OK in the GUIDE Dialog Generator:



In order to use the push button to launch a generated dialog that is updated from its initiating object, you must specify an object other than the push button itself to use as the initiating object. The object that you specify as the initiating object must have class-specific attributes from which UIL controls on the generated dialog can be updated.

To designate a object other than the push button as the initiating object, you can create a customized action that designates a new initiating object.

Using an Action to Specify an Initiating Object

You can create an action that designates an object other than the push button itself as the initiating object. You then add this action to the list of actions that are run when a user clicks on the push button.

A user-defined action can designate an initiating object in two ways:

- The action can call the UIL procedure `uil-override-initiating-object-for-dialog` to specify some object other than the push button as the initiating object of the dialog. In the action, specify:

`call uil-override-initiating-object-for-dialog(object, dialog)`

where *object* is the object to use as the initiating object, and *dialog* is the generated dialog that the push button launches.

- The action can use the following `conclude` statement to specify some object other than the push button as the initiating object of the dialog:

`conclude that the-object-initiating-dialog-activity-for dialog is object`

where *dialog* is the generated dialog that the push button launches and *object* is the object to use as the initiating object.

For information about how to add an action to the list of actions run by a push button, see [Push Buttons](#).

Note In a push button's list of actions, you must include the action that designates a new initiating object before `uil-call-update-method` action or any other action that you use to update the dialog.

Editing Generated Dialogs

For many purposes, your application can use generated dialogs without modification. However, GUIDE enables you to edit generated dialogs to change their appearance, contents, and behavior. The following table lists the chapters in this guide that describe how to edit dialogs and UIL controls:

References to Information about Editing Dialogs

Editing Operation	Reference
Allow more than one copy of the generated dialog to be used on the same G2 window.	Dialog Options Dialog
Change the contents, layout, or appearance of a dialog.	Building Master Dialogs

References to Information about Editing Dialogs

Editing Operation	Reference
Edit the appearance or behavior of a UIL control.	The chapter that describes UIL controls of that class. For example, see Push Buttons , for information about how to edit a push button.
Edit the value of a class-specific attribute that is declared as an array or a list.	Specifying Initial Contents of an Array or List Attribute
Specify the G2 object from which a UIL control is updated or to which it concludes its value.	Specifying Source and Target Objects
Specify formats and validation criteria for UIL controls.	Formats and Validation Criteria
Specify edit styles for edit boxes.	Edit Boxes, Combo Boxes, and Spin Controls

Building Master Dialogs

Describes how to use the G2 GUIDE Palette and other GUIDE tools to build customized dialogs for viewing and editing the class-specific attributes of user-defined classes.

- Introduction **64**
- Using the GUIDE Palette **64**
- Steps for Building a Master Dialog **67**
- Editing a Tab Dialog **77**
- Manipulating UIL Controls through User Menu Choices **82**
- Moving UIL Controls **84**
- Resizing UIL Controls **86**
- Transferring UIL Controls **86**
- Specifying Initial Contents of Text Objects, Message Objects, and Edit Boxes **86**
- Specifying Initial Contents of an Array or List Attribute **87**
- Specifying Source and Target Attributes of UIL Controls **89**
- Closing a Finished Subworkspace **89**
- Creating a Customized Dialog Programmatically **90**



Introduction

You can use the GUIDE palette and other GUIDE tools to build customized master dialogs for viewing and editing the class-specific attributes of user-defined classes.

The tools described in this chapter enable you to:

- Create a master dialog and add UIL controls to it.
- Create and edit a master tab dialog.
- Add UIL controls to an existing master dialog.
- Customize the layout and appearance of a master dialog.

Using the GUIDE Palette

The GUIDE palette is your basic tool for building a customized master dialog. Through the GUIDE palette, you can access the tools that you need to create a master dialog and add UIL controls to it.

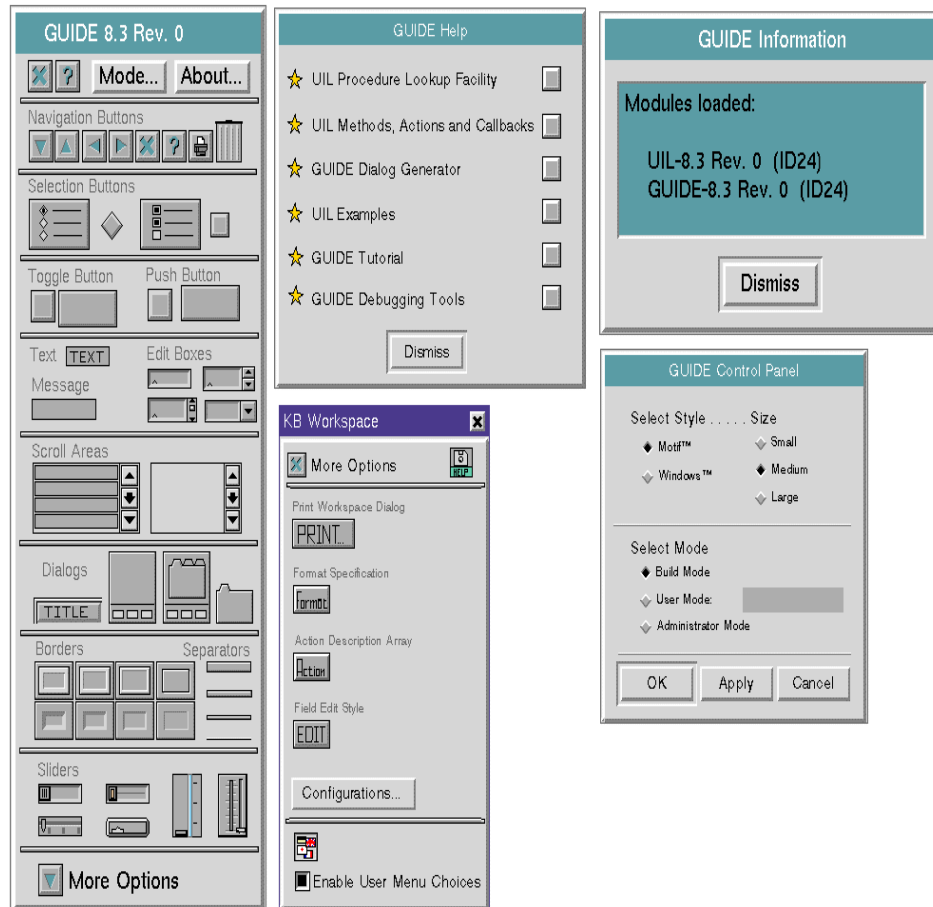
If the GUIDE palette is not visible, you can open it by selecting the following choice from the GUIDE menu bar:

View > Palettes > GUIDE Palette

Tools Provided by the GUIDE Palette

The following figure illustrates the GUIDE palette and the tools that you can access through it:

GUIDE Palette and Tools



The GUIDE palette includes the following tools:

- Icons representing dialogs and the other classes of UIL controls. You use these icons to create a master dialog and add UIL controls to it.
- A Mode button, which opens the GUIDE Control Panel. You can use the GUIDE Control Panel to select a user mode in which to run GUIDE, and to specify defaults for the window style and size of objects created from the GUIDE palette.

Note Only one copy of the GUIDE Control Panel dialog can be opened and used in each G2 window. However, users on different G2 windows can open and use different copies of the GUIDE Control Panel dialog.

- An About button, which opens the GUIDE Information dialog. This dialog tells you which versions of GUIDE and UIL are currently running.
- A Help button, which opens the GUIDE Help dialog. The GUIDE Help dialog enables you to:
 - Display argument signatures and return values of UIL procedures
 - Open dialogs for creating customized methods, actions, and callbacks
 - Start the GUIDE Dialog Generator
 - Go to a workspace that contains working GUIDE examples from the GUIDE Demo KB (*guidemo.kb*)
 - Start the GUIDE Tutorial, which leads you through the steps for creating a GUIDE user interface.

For information about the GUIDE Help dialog, see [Help Dialog](#).

- A More Options navigation button. Clicking on this button opens the More Options palette, which contains icons that you can select and drop to create the following resources:
 - A Print button. Users can click the Print button to open the Printer Setup Dialog. In this dialog, users can specify options for printing the workspace that contains the Print button.
 - A reusable Format that specifies aspects of format such as the use of quotation marks, capitalization, and hyphens. You can apply Formats to edit boxes, text objects, and message objects. Formats can also specify validation criteria for edit boxes. For information about formats, see [Formats and Validation Criteria](#).
 - An action description array. An action description array is a list of procedures that perform operations on a dialog. You can invoke an action description array on a dialog by invoking the procedure `uil-control-dialog-callback`. For information about action description arrays, see [Controlling Dialogs with Actions](#).
 - A reusable field edit style that specifies the behavior of the editor for edit boxes. You can apply field edit styles to individual edit boxes. For information about field edit styles, see [Edit Styles for Edit Boxes](#).

Steps for Building a Master Dialog

The following sections describe the steps that you must follow to build a master dialog using the GUIDE palette and other tools.

Note You cannot use the push buttons and other UIL controls on a master dialog. To test a master dialog and the UIL controls that it contains, you must launch a copy dialog of the master dialog. For information about how to do this, see [Launching Dialogs](#).

- 1 Change your user mode to Build Mode.

To change your mode, click the Mode button in the GUIDE palette. Clicking Mode opens the GUIDE Control Panel:



In the GUIDE Control Panel, select **Build Mode** and click **OK**.

If you are currently running GUIDE in Administrator Mode, you can open the GUIDE Control Panel dialog by clicking on the Mode button and choosing **select** from the Mode button's menu.

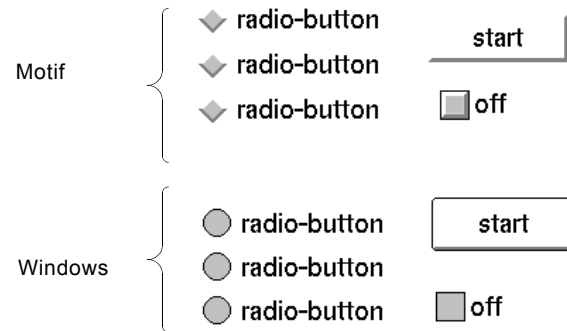
You can also use all the features of GUIDE in Administrator Mode. However, Build Mode is the recommended mode for constructing a GUIDE user interface.

- 2 Set defaults for the size (small, medium, or large) and window style (Motif™ or Windows™) of UIL controls that you subsequently add to the master dialog. The default size and style do not apply to UIL controls already on the master dialog.

You set the default size and window style In the GUIDE Control Panel.

Select **Motif** or **Windows** in the group of buttons under the label **Select Style**.

The following figure illustrates several buttons in each of these styles:



Select **Small**, **Medium**, or **Large** in the group of buttons under the label **Size**.

The following figure illustrates a push button in each of these sizes:



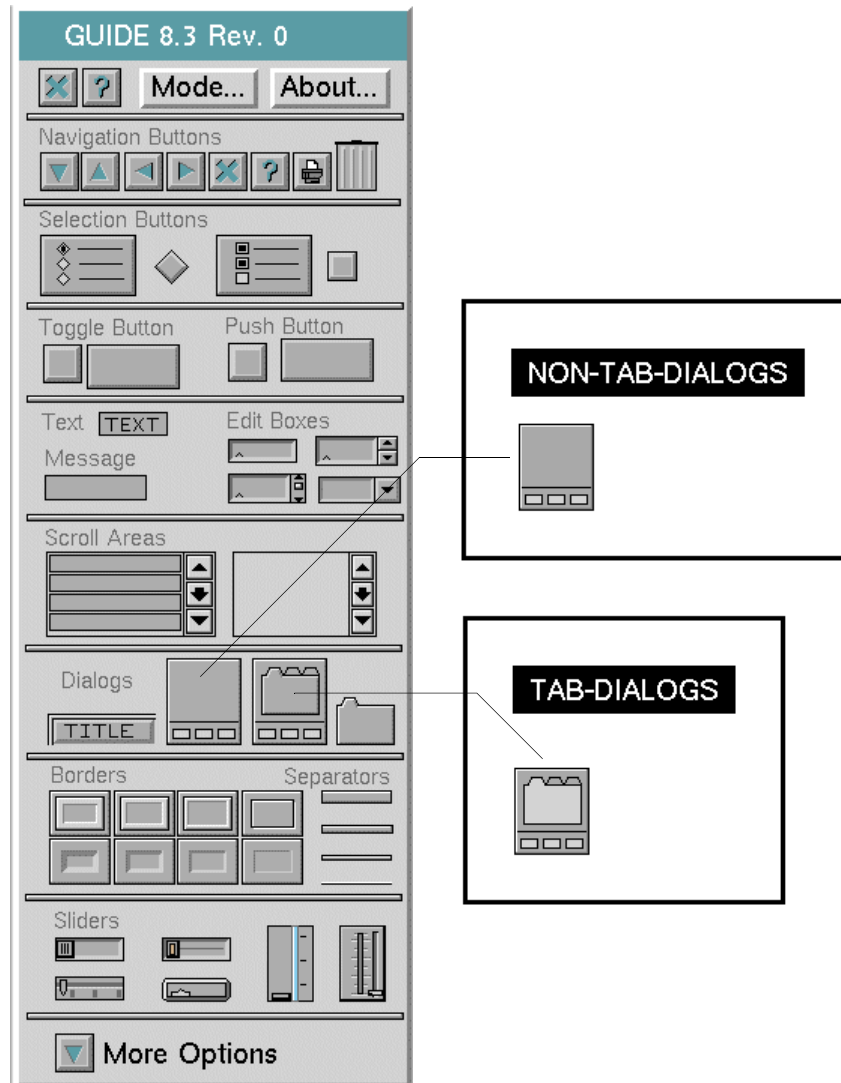
You can change the size of a UIL control after you have created it using the GUIDE editor for that UIL control, or by selecting the G2 **change size** or **change min size** menu choices.

The window style and size that you select in a GUIDE Control Panel apply globally throughout the G2 knowledge base that you are using.

Note Labels of Motif-style buttons are contained in icons of these buttons. Labels of Windows-style buttons are contained in separate UIL label objects. However, any UIL procedure that operates on button labels has the same effect on both Motif and Windows style buttons.

- 3 Click **OK** to apply the changes that you make in the GUIDE Control Panel.

- 4 Create a master dialog using the GUIDE palette. You can do this in either of two ways:
 - Select the dialog icon in the GUIDE palette for the style of master dialog that you want to create (with or without tab pages) and drop it on a workspace:

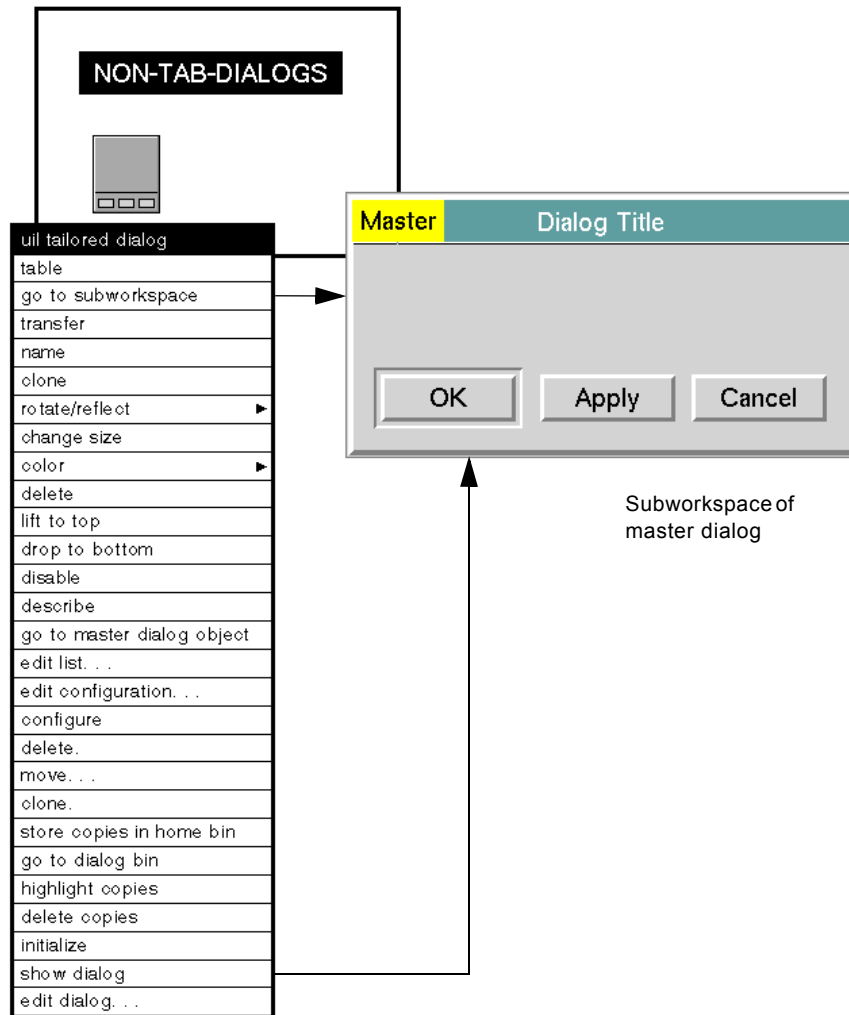


- From the GUIDE menu bar, select one of the following:
 - Item > GUIDE Objects > Dialogs > uil-tailored-dialog
 - Item > GUIDE Objects > Dialogs > uil-tab-dialog

Selecting this menu choice causes a master dialog icon to be attached to your cursor. You can drop this icon on any workspace.

- 5 Open the subworkspace of the new master dialog so that you can add UIL controls to it.

To open the subworkspace, select `go to subworkspace` or `show dialog` from the menu of the new dialog:



As shown in the figure above, the subworkspace of the new master dialog has a title bar and three push buttons: OK, Apply, and Cancel. These push buttons have appropriate default actions.

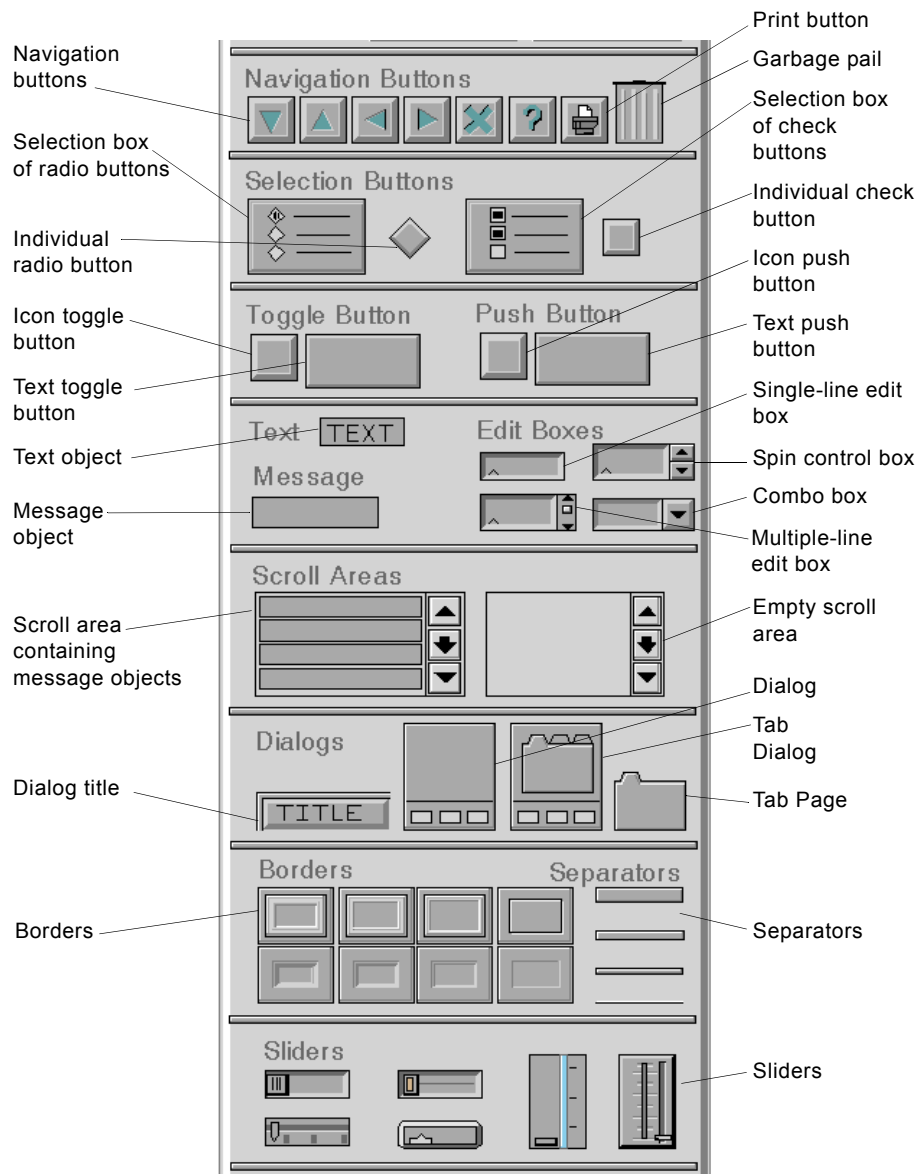
However, the subworkspace does not contain any of the UIL controls that enable users to view and edit attributes of G2 objects. You must add these components to the subworkspace, using the GUIDE palette.

Note The title bar of every master dialog contains the word Master. If you do not see Master in the title bar of a dialog subworkspace, that dialog is not a master and you cannot edit it.

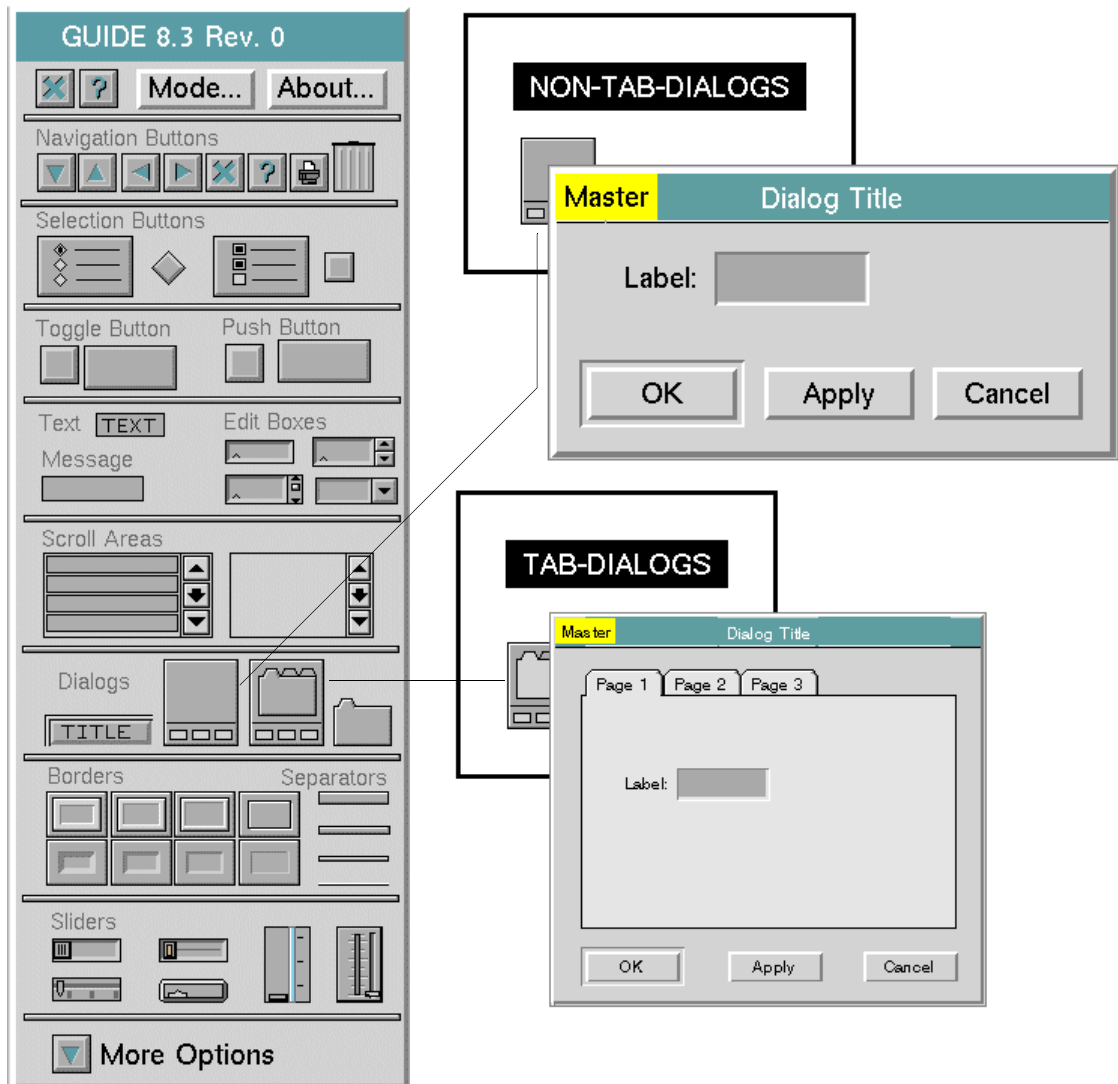
6 Add UIL controls to the master dialog.

To add a UIL control to a master dialog, select the icon on the GUIDE palette that represents the UIL control that you want to add and drop it on to the subworkspace of the dialog.

The following figure illustrates the icons on the GUIDE palette:



The following figure illustrates how to add an edit box to a dialog:



You can add most classes of UIL controls to a master dialog simply by selecting their icons and dropping them on the subworkspace of the master dialog. Before you add UIL controls to a tab dialog, first select the tab page to which you want to add the control. For information about how to do this, see [Editing a Tab Dialog](#).

You must follow special steps to add dialog titles, selection buttons, and scroll areas to master dialogs. The following sections describe how to add selection buttons and scroll areas to master dialogs.

Adding a Dialog Title

You can add dialog titles to master dialogs that do not have titles. To do this, you select the Title icon and drop it on the subworkspace of the master dialog.

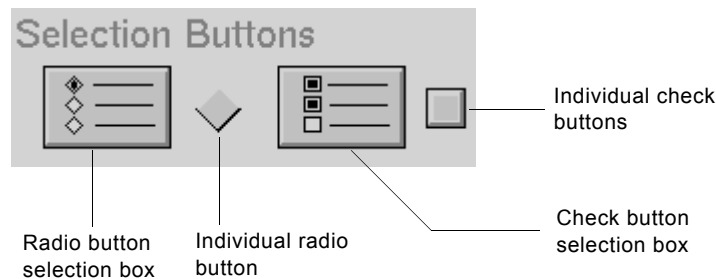
You cannot add a dialog title to a master dialog that has a title. However, you can add them to master dialogs whose titles have been deleted.

Adding Radio Buttons and Check Buttons

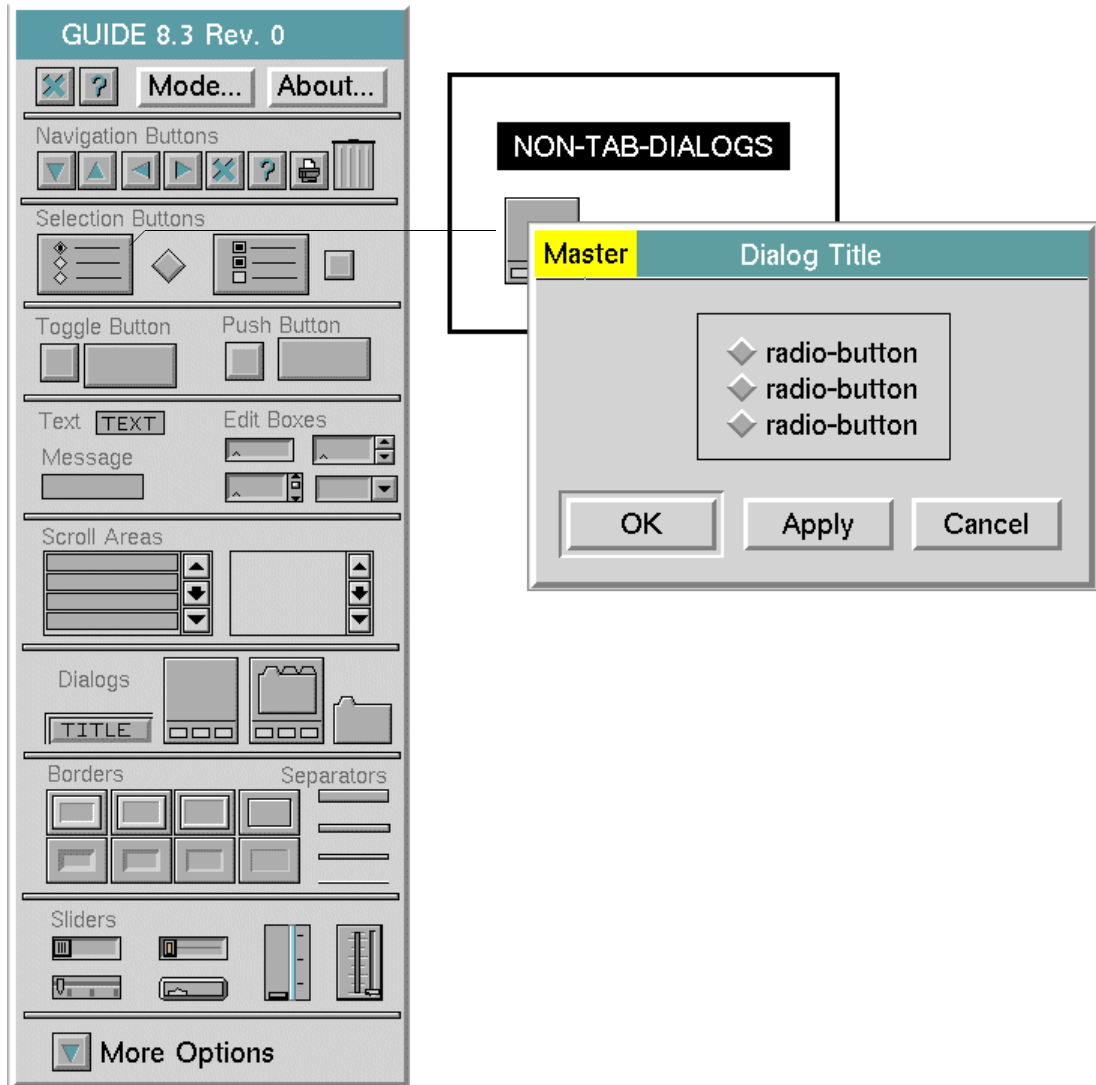
Radio buttons and check buttons can be used only in selection boxes. To add radio buttons or check buttons to a master dialog, you must first add a selection box for radio buttons or check buttons. By default, selection boxes contain three buttons.

You can then add radio buttons or check buttons to the existing selection box on the master dialog. You cannot use radio buttons and check buttons in the same selection box.

The following figure illustrates the icons on the GUIDE palette that represent selection boxes, radio buttons, and check buttons:



The following figure illustrates how to add a selection box of radio buttons to a master dialog:



To add a radio button or check button to a selection box, click the radio button or check button icon on the GUIDE palette and drop it on or near the selection box. The new radio button or check button is added at the bottom of the selection box.

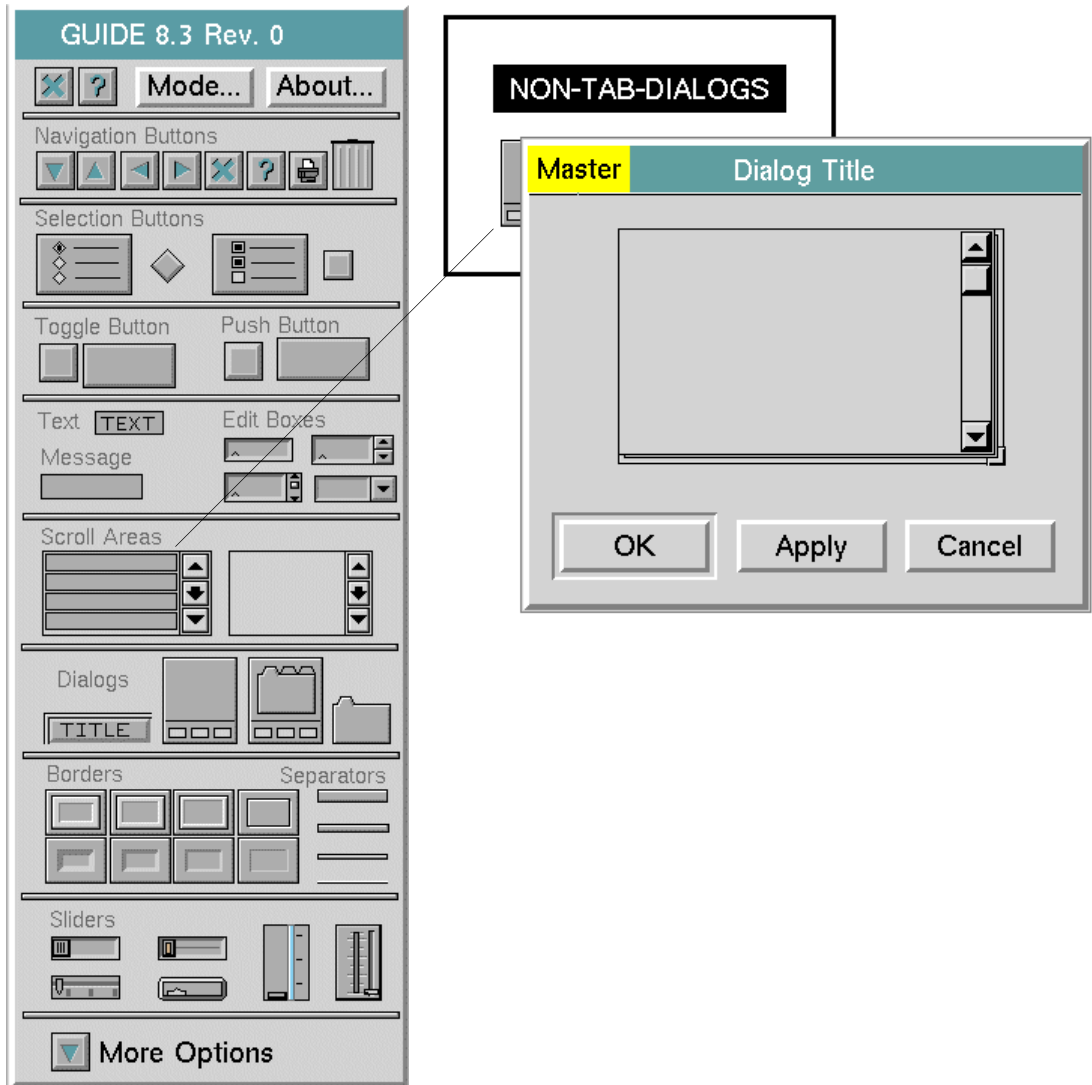
Adding Scroll Areas and Message Objects

You can add message objects only to existing scroll areas on a master dialog. Message objects can be used only in scroll areas.

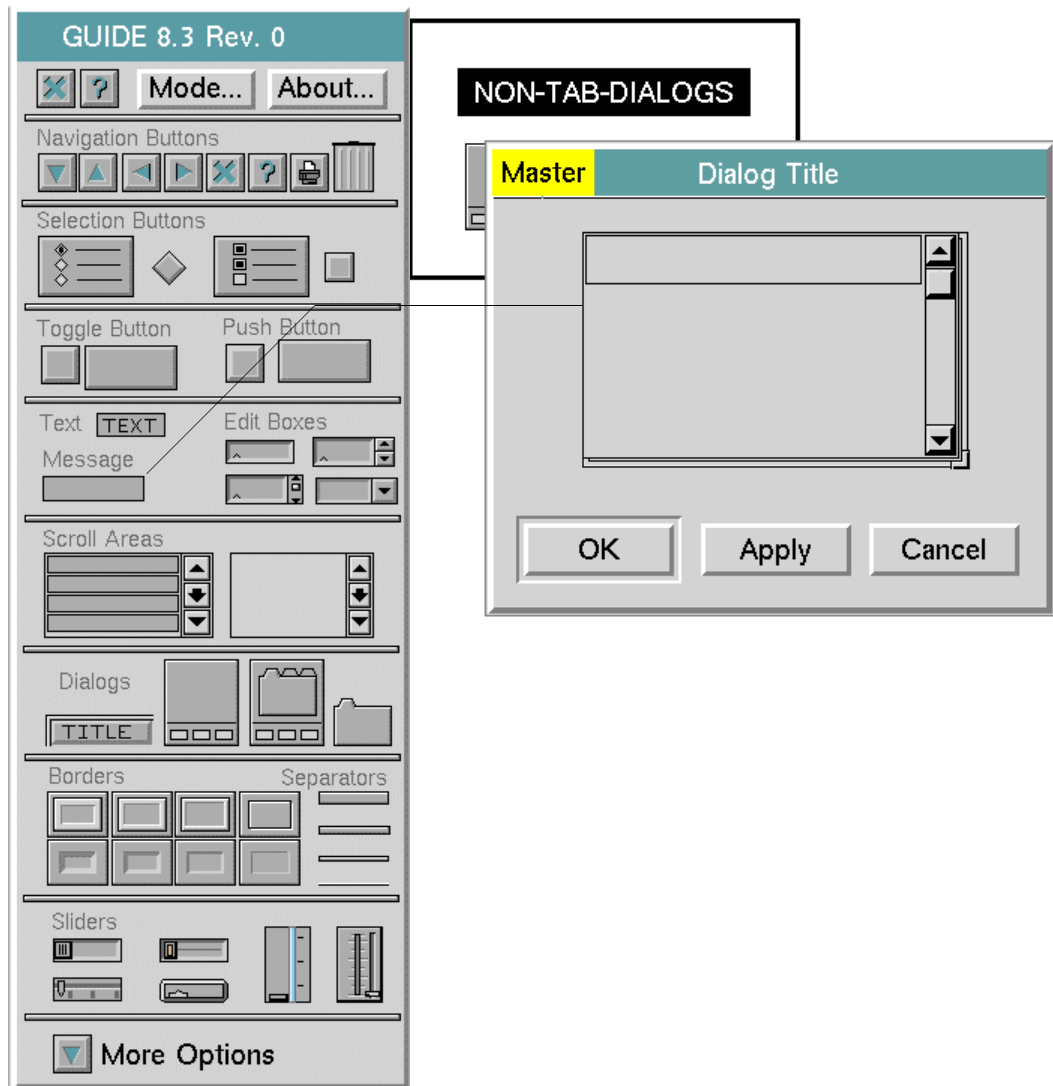
The following figure illustrates the icons on the GUIDE palette that represent scroll areas that contain message objects by default, and scroll area that contain no message objects:



The following figure illustrates how to add an empty scroll area to a master dialog:



The following figure illustrates how to add a message object to a scroll area on a master dialog:



Changing the Size of a Dialog Subworkspace

You can change the size of a dialog subworkspace by dragging the upper-right or lower-left corner of the subworkspace. Drag on the corner of the subworkspace immediately inside the subworkspace's border.

You can also enlarge a dialog subworkspace by dragging a UIL control to the edge of it. This enlarges the subworkspace in the direction in which you drag. As you drag a UIL control away from the perimeter of the subworkspace, the border remains tightly wrapped around the subworkspace.

Editing a Tab Dialog

A **tab dialog** is a dialog of the subclass of `uil-tailored-dialog` that supports the operation of tab pages on its subworkspace. Each **tab page** is an object of the subclass of `uil-grmes` that provides a grouping for other objects.

A tab page contains objects by storing them into an `item-list` attribute of the tab page. As the tab pushbutton is selected by the user, all of the contents of the tab page are raised to the top and become visible.

Each operation that affects a tab page, such as move, clone, transfer, and delete, also carries out that operation on the contents of the tab page.

A **tab push button** is a push button of the subclass of `uil-text-pushbutton` that lifts the associated tab page to the top of the stack of tab pages. On a mouse-down gesture, the tabbed button is selected. This style conforms to current window standards.

The tab pages on a dialog are managed by a **tab page group**. There can be only one tag page group on a tab dialog. If you add tab pages to a dialog, the tab pages are automatically added to the tag page group on that dialog.

Lifting a Tab Page to the Top of the Stack

To lift a tab page to the top of the stack, click the tab page. In build or administrator mode, you can then choose the select menu choice, just as you would for any UIL button. In any other mode, a single mouse click lifts that tab page to the top.

Adding UIL Controls to a Tab Page

You can add UIL objects to a tab page on a master dialog in the same way that you can add UIL objects to any dialog. You can place any G2 object, including UIL objects, on the area of the dialog outside the stack of tab pages.

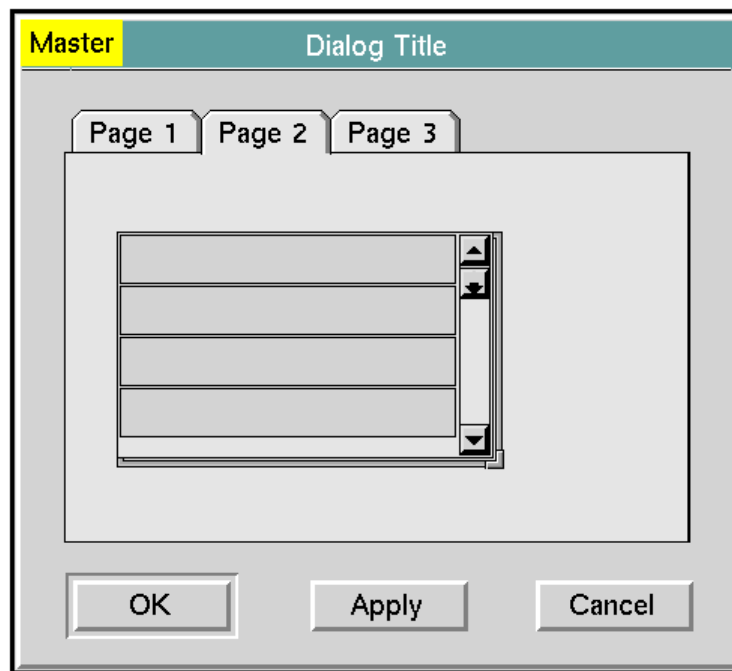
To add an object to a particular tab page:

- 1 Create the object and drop it manually onto the top page.
- 2 If the tab page to which you want to add an object is not currently the top page, click the tab for that page to bring it to the top, then drop the object on that tab page.

For example, to add a scroll area to a tab page labeled Page 2:

- 1 Select Page 2 by clicking on the button labeled Page 2.
- 2 Choose **select** from the menu.
- 3 Click the scroll area icon on the GUIDE palette and drag it on top of Page 2.

For example:



Changing the Size and Labels of Tab Buttons

You can change the size and labels of tab buttons by editing attributes of the buttons:

- To change the size of a tab button, change the value of the `uil-size` attribute to one of: `small`, `medium`, or `large`.
- To change the label of a tab button, change the `label` attribute of the button to the desired value. Enclose the label text in quotation marks (`""`).

By default, the button is automatically resized to fit the size of the new label. You can enable or disable the automatic resizing of tab buttons by setting the `resize-to-fit-label` attribute of the button to `true` or `false`.

Changing the Placement of Tab Buttons

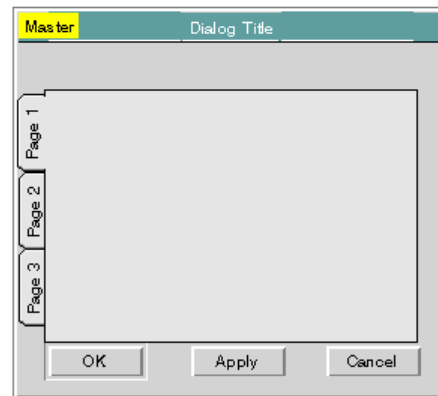
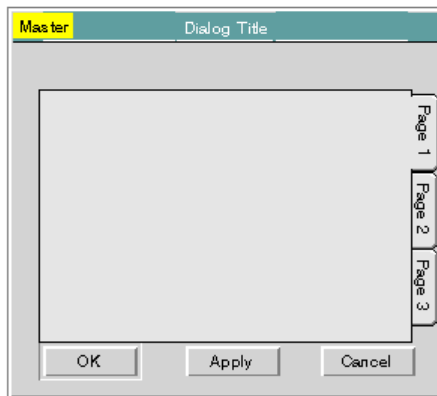
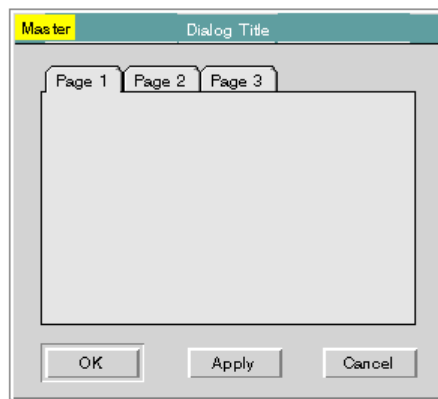
For any stack of tab pages, you can specify whether the tab buttons are placed on the top, left, or right edge of the stack.

To change the placement of tab buttons:

- 1 Select rotate tab pushbuttons from the menu of a tab page or a tab pushbutton.
This user menu choice aligns the tab buttons across the top, left, or right side of the tab pages on the dialog.
- 2 Open the attribute table of the master tab dialog, and in the `uil-tab-page-button-placement` attribute, specify one of: `top`, `left`, or `right`.

The tab pages are aligned along the left or right or across the top of the dialog, depending on the value that you specify for the `uil-tab-page-button-placement` attribute.

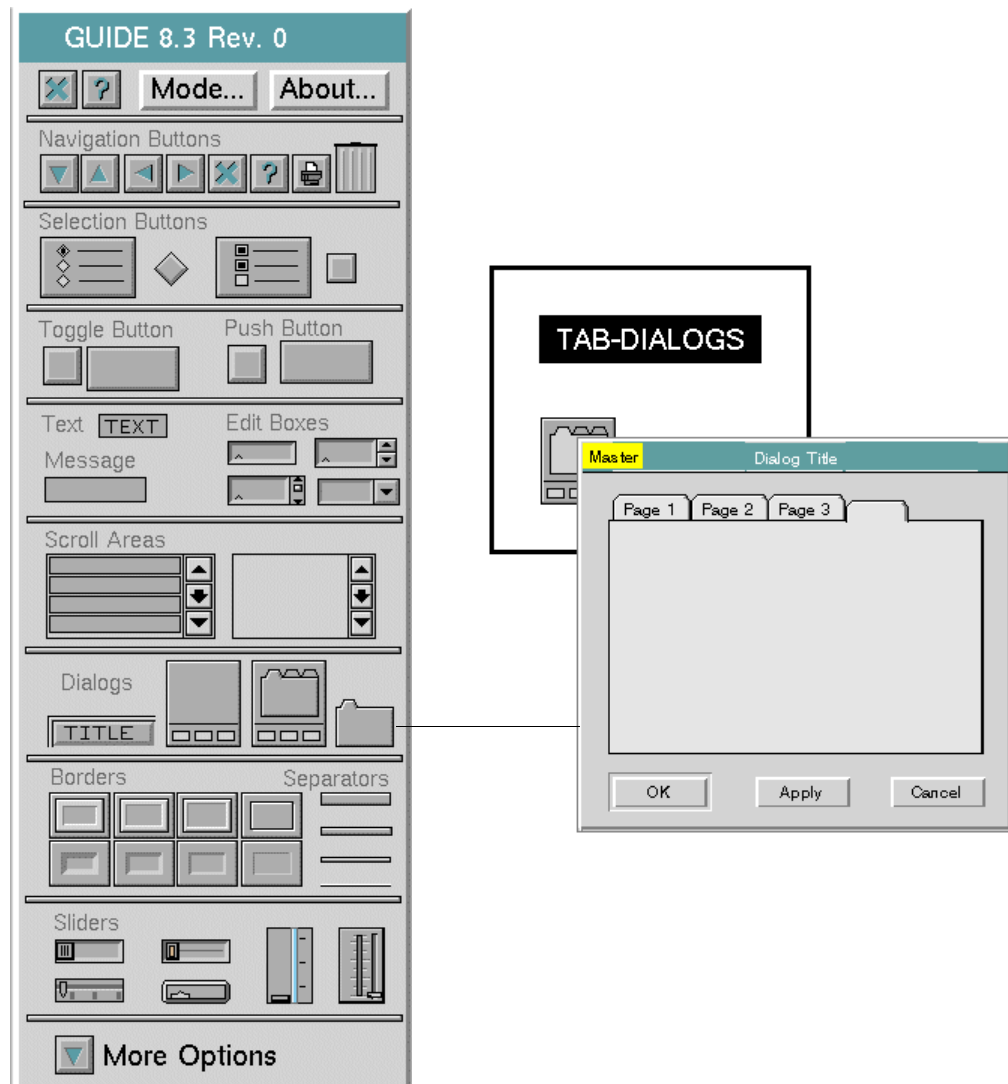
The following figure illustrates the top, right, and left placement of tab buttons:



Adding New Tab Pages

A new tabbed master dialog contains three tab pages by default. You can add any number of tab pages to the dialog.

To add a tab page to a tab dialog, select the Tab Page icon on the GUIDE palette and drop it onto the stack of tab pages in the master dialog subworkspace:



The new tab page is added to the bottom of the stack. If necessary, the stack automatically enlarges to fit the new tab page.

Deleting a Tab Page

To delete a tab page, select the tab page or tab button and choose the **delete.** menu choice. A confirmation dialog appears, asking you to confirm that you want to delete the tab page.

Deleting a tab page deletes the tab page itself, its tab button, and any `uil-grobj` or `uil-grmes` object on the tab page.

Cloning a Tab Page

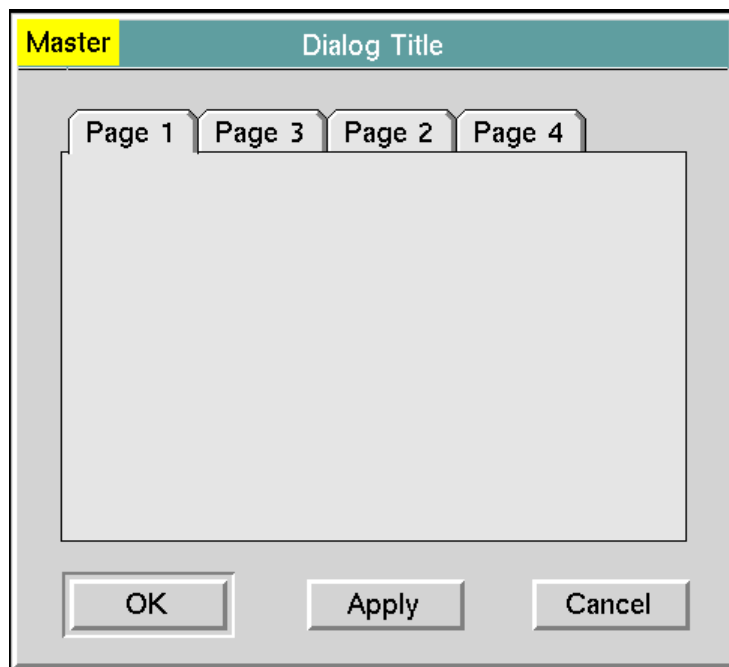
To clone a tab page, select the tab page and choose the **clone.** menu choice. The new tab page is placed at the bottom of the stack of tab pages on the dialog.

When you clone a tab page, you also clone the contents of the tab page.

Reordering Tab Pages

To move a tab page to a different place in the stack, drag that page's tab button to the desired position and release it. (In any user mode other than **administrator**, the drag motion is constrained to move horizontally or vertically, depending on the placement of the tab buttons.)

For example, to move Page 3 so that it is between pages 1 and 2, drag the tab on Page 3 to where it is between the Page 1 tab and the Page 2 tab:



Lifting and Dropping Tab Pages with non-UIL Objects

When a tab page is lifted above or dropped below other tab pages, any UIL objects on that tab page are lifted or dropped with it automatically. Tab pages are raised and lowered using G2's system procedures for dropping and lifting objects on G2 workspaces.

However, any non-UIL objects on a tab page are *not* lifted or dropped with that tab page automatically. To ensure that the non-UIL objects are lifted or dropped with the tab pages, you must write user code that maintain these objects in the tab-page-contents-list of the tab page, as well as support the lifting and dropping of items on the G2 workspace.

Moving the Stack of Tab Pages

You can move the entire stack by dragging the page that is currently at the top of the stack to the desired position. When you drop this page, the rest of the tab pages follow.

Moving the tab pages retains the positions of objects on the tab pages relative to those tab pages.

Resizing the Stack of Tab Pages

To change the size of a tab page:

- 1 Click the top tab page and select the **change size** menu choice.
- 2 Specify a new size for the top tab page.
The tab page is resized immediately.
- 3 Click the top tab page again and select the **resize all tab pages** menu choice.

All the other tab pages in the stack are resized to the same size as the top tab page.

Transferring Tab Pages

You can transfer a tab page to another master tab dialog by selecting the tab page and choosing the **transfer** menu option.

You can transfer a tab page only to another master tab dialog. If you attempt to transfer the tab page to any other location, the tab page is returned to its original location.

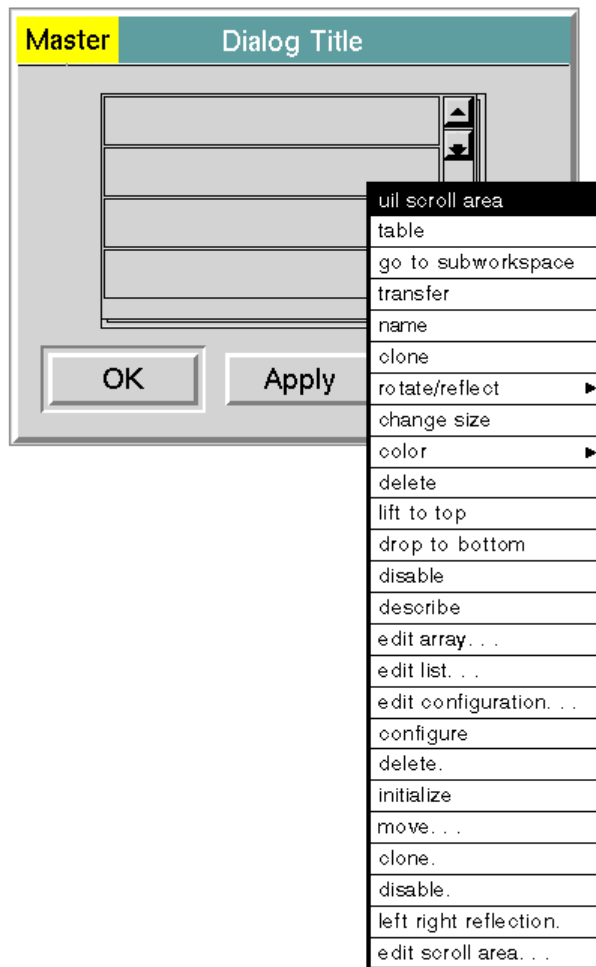
Manipulating UIL Controls through User Menu Choices

Dialogs and UIL controls have menus of choices that you can select to perform operations on these objects, such as naming, rotating, transferring, deleting, and moving them. You can also access the object editor for a UIL control through its user menu. For example, to open the Edit Scroll Area dialog to edit a particular scroll area, select **edit scroll area** from the scroll area's user menu.

To use menus of dialogs and UIL controls, you must be running in Build Mode or Administrator Mode.

To open the user menu of a UIL control, click the UIL control. You can then select menu choices to perform operations on the UIL control.

The following figure illustrates the user menu of a scroll area:



When you select a menu choice, the operation associated with that choice is carried out by a special procedure known as a **UIL method**. UIL provides a set of system-defined methods for carrying out the operations associated with user menu choices. You can create customized methods to use in place of these system-defined methods. For more information about methods, see [Methods, Actions, and Callbacks](#).

A description of the user menu choices for each kind of UIL control is included at the end of the chapter in this guide that describes the UIL control.

Moving UIL Controls

After you add UIL controls to a dialog, you can move them within the dialog.

Moving UIL Controls by Dragging Them

You can move any UIL control on a dialog subworkspace by moving the pointer to the object, holding down any mouse button, and dragging the object.

You can also drag a dialog subworkspace within the workspace where it is displayed. To do this, move the pointer to any blank area within the subworkspace and then drag.

Moving UIL Controls with Labels

When you move or transfer a UIL control with a label, the label is automatically moved or transferred along with the object. When you drag the label of an object, the object is moved along with the label.

Push buttons, radio buttons, check buttons, button groups, toggle buttons, navigation buttons, and edit boxes all have labels.

Moving UIL Controls with Borders

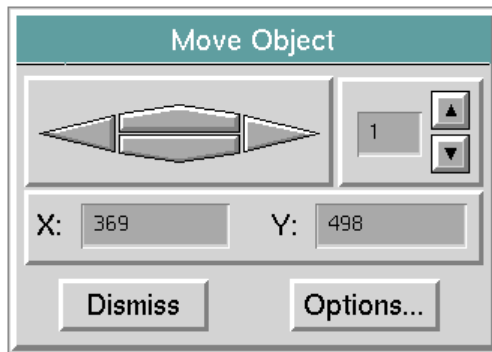
When you move or transfer a UIL control with a border, the border is automatically moved or transferred with the UIL control.

You can move a border by grabbing the upper right corner or the lower left corner with the mouse.

Edit boxes, text objects, and workspaces are the only UIL controls that can have borders.

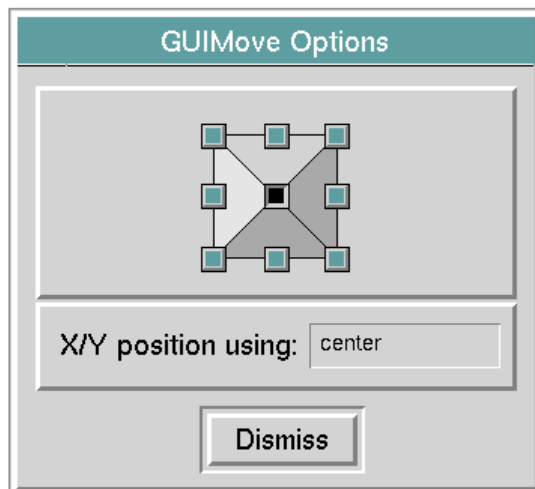
Using the Move Menu Choice

If you need to position a UIL control more precisely than you can by dragging it, select **move** from the UIL control's user menu. This opens the Move Object dialog, which enables you to specify the x and y coordinates and centering options for the UIL control:



You can move the object up or down, or to the left or right, by clicking the arrows in the upper left corner of the Move Object dialog. You can specify the number of pixels that the object moves each time you click by scrolling the value in the edit box in the upper right corner of the Move Object dialog; by default, each click moves the object 1 pixel. The edit boxes label X: and Y: display the current x and y coordinates of the object.

Clicking the Options button opens the GUI Move Options dialog, which allows you to move the object to the top-left, top-center, top-right, center-left, center, center-right, bottom-left, bottom-center, or bottom-right:



Resizing UIL Controls

To resize a toggle button, radio button, check button, push button, or navigation button, choose **change size** from its menu.

To resize an edit box, an edit box label, or a text object, select **change min size** from its menu.

When you select **change size** or **change min size**, a black border appears around the object, and a G2 dialog of resizing options appears on the workspace. For information about how to use the G2 resizing dialogs, see the *G2 Reference Manual*.

Transferring UIL Controls

You can transfer any UIL control from one subworkspace or workspace to another by selecting **transfer** from the menu of the UIL control and dropping it on the other workspace.

When you transfer a message object, you must drop it on a scroll area on the destination subworkspace. Transferring a scroll area transfers the scroll area and all of the message objects that it contains.

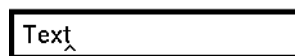
When you transfer a radio button or check button, you must drop it on a radio box or check box on the destination subworkspace. Transferring a selection box transfers all the buttons within the selection box.

Transferring objects with borders transfers the borders.

Specifying Initial Contents of Text Objects, Message Objects, and Edit Boxes

To set the initial contents of edit boxes, text objects, and message objects, select the object and choose **edit** from the object's menu. The editor appears over the object that you selected.

When you open the editor on a text object, it looks like this:



When you open the editor on an edit box with an edit style that specifies single line editing mode, the editor looks like this:



When you open the editor on an edit box that uses a multi-line edit style, the editor looks like this:



The text that you edit using the **edit** menu choice is stored in the **message-contents** attribute of the object that you are editing. This attribute contains the full, unformatted text as it has been entered by application developers or users.

If you specify a format for the text object through the Select Format dialog, this format is applied to the contents of the **message-contents** attribute, and the resulting formatted text is stored in the **text** attribute of the text object. Any limitation on the length of the text object that you specify is also applied to the formatted text in the **text** attribute. You cannot directly view or edit the **text** attribute. The formatted text in the **text** attribute is the version that is displayed to the user.

When you are working with an international language system of G2, such as Japanese or Korean, you need the edit menus in order to use the basic language features. You can enable these menus by enabling the Display menus on edit option in the Editor Behaviors dialog. For information about how to use this dialog, see [Editor Behaviors Dialog](#).

Specifying Initial Contents of an Array or List Attribute

You can set the initial contents of an attribute declared as an array or list using the Edit Array and Edit List dialogs.

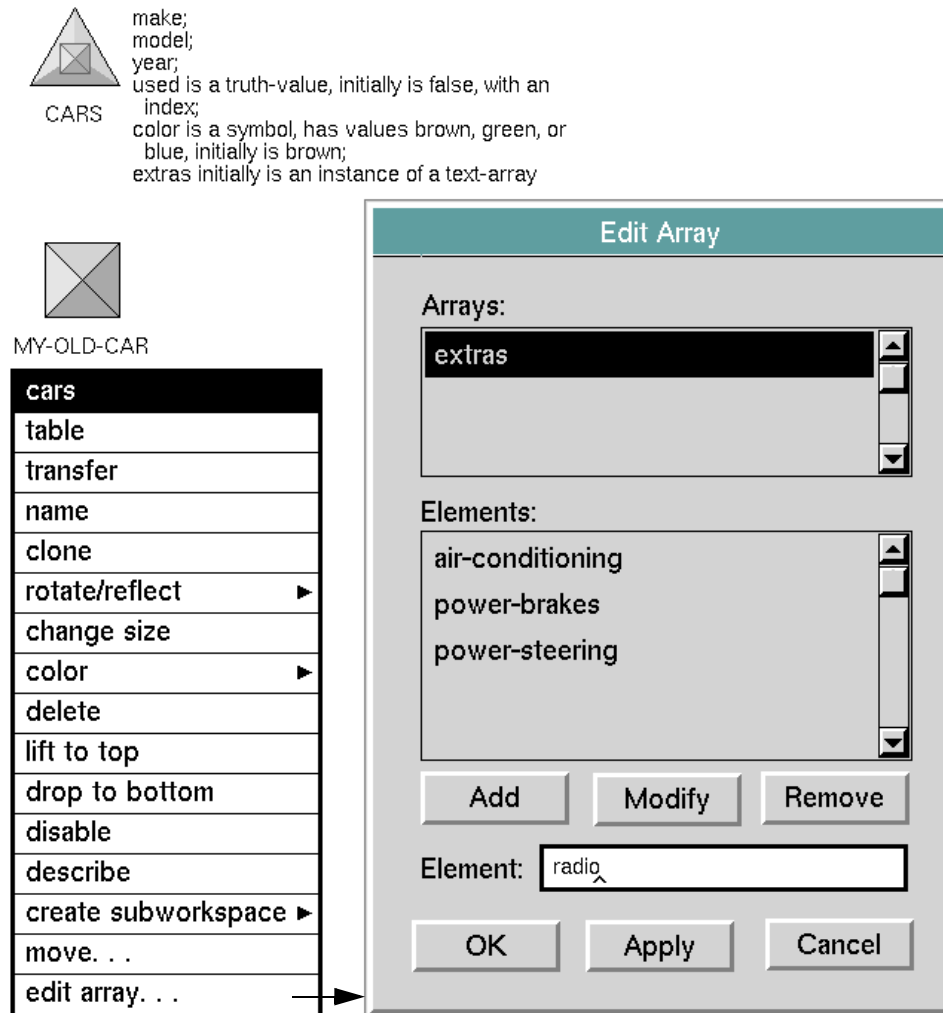
To open these dialogs, select **edit array** or **edit list** from the menu of the G2 object. These menu choices are added to the menus of G2 objects that have class-specific attributes declared as instances of **text-array** or **text-list**.

You can also launch the Edit Array and Edit List dialogs programmatically, using the UIL procedure **guidata-launch-dialog-for-item**. For information about this procedure, see the *G2 GUIDE/UIL Procedures Reference Manual*.

For example, the class **cars** has a class-specific attribute declared as follows:

```
extras initially is an instance of a text-array
```

The following figure illustrates how to open the Edit Array dialog to edit the extras attribute of an object of the cars class named my-old-car:



To use the Edit Array or Edit List dialog:

- 1 Select the name of the array or list attribute that you want to edit.
The names of the array or list attributes of the object are displayed in the scroll area at the top of the dialog. In the Edit Array dialog, this scroll area is labeled Arrays. In the Edit List dialog, the scroll area is labeled Lists.
- 2 In the Elements field, enter the text of an element that you want to add to the array or list.
- 3 Click Apply to add the element to the array or list. You can also add the element by pressing the Tab or Return key. The elements that you add appear in the Elements scroll area.

- 4 If you need to delete the contents of the Elements scroll area, click the Clear button.
- 5 Click OK to apply your edits to the array or list that you selected.

The contents that you specify for arrays or lists using the Edit Array and Edit List dialogs are not permanent. They are lost whenever the array or list attribute is updated or edited.

Specifying Source and Target Attributes of UIL Controls

When you create UIL controls using the GUIDE palette, you must edit each UIL control to specify its the source and target objects and attributes.

The source attribute and target attribute are class-specific attributes of a G2 object, where:

- The source attribute contains the value with which the UIL control is updated.
- The target attribute receives the value of the UIL control when the UIL control concludes its values.

The values of individual UIL controls are updated and concluded when the dialog as a whole is made to update or conclude its contents.

For information about how to set source and target objects and attributes of UIL controls, see [Specifying Source and Target Objects](#).

Note When you generate a dialog using the GUIDE Dialog Generator, each UIL control on the dialog is automatically assigned source and target objects and attributes. However, you can edit UIL controls on generated dialogs to change their source and target attributes.

Closing a Finished Subworkspace

When you finish adding UIL controls to the subworkspace of the new master dialog, you can close it by selecting Hide Workspace from its menu.

Creating a Customized Dialog Programmatically

You can create a customized dialog programmatically using the UIL procedure `uil-generate-customized-dialog`. This procedure enables you to specify:

- The attributes of a user-defined class for which you want to add UIL controls to the generated dialog.
- The class of UIL control that you want to use to represent each attribute.

For information about how to use `uil-generate-customized-dialog`, see the *G2 GUIDE/UIL Procedures Reference Manual*.

Using UIL Controls on a Workspace

Illustrates several ways to use UIL controls on a workspace, without incorporating them into a dialog.

Introduction 93

Examples of UIL Controls Used on Workspaces 93

Placing UIL Objects on Subworkspaces of G2 Items 98



Introduction

You can use UIL controls on a workspace without incorporating them into a dialog.

UIL controls on a workspaces can be updated from and conclude their values to G2 objects. However, the source and target objects of UIL controls on a workspace must be named objects. You cannot specify *initiating object* as the source or target object of a UIL control used on a workspace.

You may have to write procedures to update and conclude the values of UIL controls used on a workspace without the support of a dialog.

Examples of UIL Controls Used on Workspaces

This section contains several simple examples of UIL controls that are used on workspaces. These examples illustrate only a few common ways to use UIL controls on workspaces.

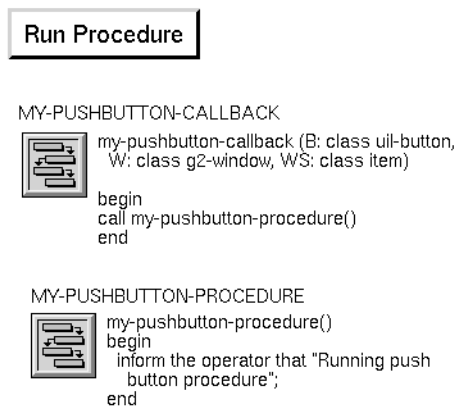
Invoking a Procedure from a Push Button on a Workspace

You can use a push button on a workspace to run a procedure or a set of procedures. When a user clicks on the push button, the callback procedure of the push button is run. You can create user-defined callbacks to run any system-defined or user-defined procedure.

To use a push button to run procedures:

- 1 Create a callback procedure that calls the procedure or procedures that you want users to be able to run by clicking on the push button.
- 2 Specify this callback procedure as the callback of the push button.

The figure below illustrates a push button labeled Run Procedure that calls a user-defined callback procedure named `my-pushbutton-callback`. This callback calls a user-defined procedure, `my-pushbutton-procedure`, that posts a message on the Message Board.



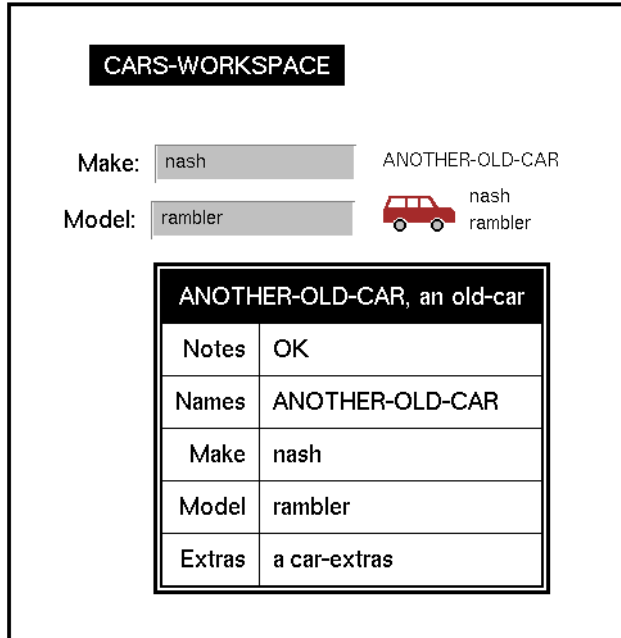
For information about how to create callback procedures, see [Methods, Actions, and Callbacks](#). For information about how to associate a user-defined callback with a push button, see [Push Buttons](#).

Using an Edit Box on a Workspace

The following example shows how to conclude the values in edit boxes on a workspace to a target object, and how to update the values in the edit boxes.

Concluding the Value in an Edit Box on a Workspace

The following example shows two edit boxes, labeled Make and Model, that update their values to attributes of an object named `another-old-car`:



The following attributes of the edit box labeled Make enable users to update the make of the object named `another-old-car`:

Uil conclude value immediately	true
Uil event target object	another-old-car
Uil event target attribute	make

The `uil-conclude-value-immediately` attribute of the Make edit box is set to `true`. This attribute value causes the value of the edit box to be concluded immediately when the edit box loses focus. Thus, users can apply the contents of the Make edit box by pressing the Return or Tab key.

The `uil-event-target-object` of the Make edit box is set to `another-old-car`, the target object of this edit box.

The `uil-event-target-attribute` of the Make edit box is set to `make`, the attribute of `another-old-car` to which this edit box concludes its value.

Updating the Value in an Edit Box on a Workspace

You can run a procedure to update an edit box on a workspace, provided that you have specified a source object and source attribute for the edit box. The source object must be a named object.

The following figure illustrates a procedure that can update an edit box on a workspace, and the action button used to run this procedure:

Update Edit Box start my-update-edit-box-on-ws(the uil-edit-box upon this workspace, this window)

MY-UPDATE-EDIT-BOX-ON-WS



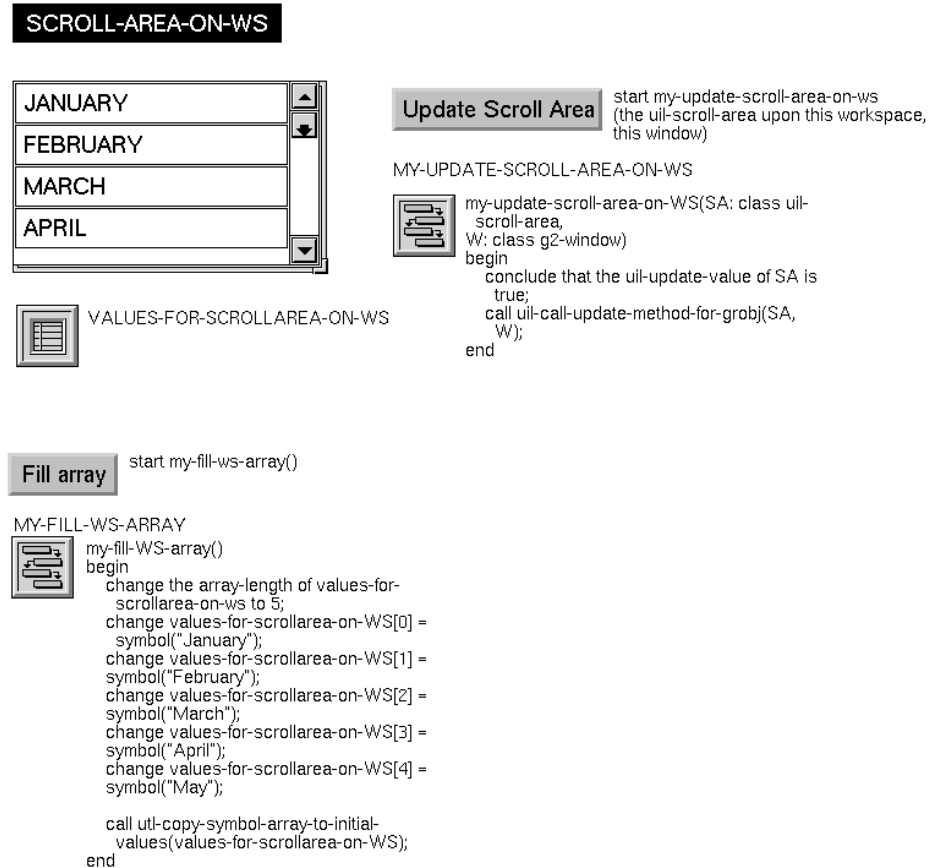
```
my-update-edit-box-on-WS(EB: class uil-edit-  
box,  
W: class g2-window)  
begin  
  conclude that the uil-update-value of EB is  
  true;  
  call uil-call-update-method-for-grobj(EB,  
  W):
```

The procedure `my-update-edit-box-on-ws` concludes that the `uil-update-value` attribute of the edit box is `true`. This allows the update method for the edit box to be run. The procedure then calls `uil-call-update-method-for-grobj`, which runs the update method of the edit box. The update method updates the edit box with the value of the edit box's source attribute.

Using a Scroll Area on a Workspace

To use a scroll area on a workspace, you must specify named objects as the source and target objects of the scroll area. You must also use procedures to update and conclude the values of the message objects in the scroll area.

The following figure illustrates how a scroll area on a workspace can be updated with the values in a symbol array:



The source object of the scroll area is the symbol array `values-for-scrollarea-on-ws`. The source attribute of the scroll area is the `initial-values` attribute of `values-for-scrollarea-on-ws`. `Initial-values` references the elements of the array.

The action button labeled **Fill array** starts a user-defined procedure, `my-fill-ws-array`, which initializes the first five elements of the array to the first five months of the year.

The action button labeled **Update Scroll Area** starts a user-defined procedure named `my-update-scroll-area-on-ws`. This procedure sets the `uil-update-value` attribute of the scroll area to `true`, so that the scroll area's update method can be run on it. The procedure then calls the UIL procedure `uil-call-update-method`, which runs the update method of the scroll area.

The update method of these scroll area updates the scroll area with the array elements referenced by the `initial-values` attribute of the symbol array `values-for-scrollarea-on-ws`.

Placing UIL Objects on Subworkspaces of G2 Items

You can place UIL objects, such as borders, buttons, and dialogs, on the subworkspaces of G2 items, such as class definitions and action buttons.

Customizing Dialogs

Describes how to edit and customize GUIDE dialogs.

Introduction	99
Editing Master Dialogs	100
Controlling Dialogs with Actions	115
Creating Systems of Cascaded Dialogs	118
Specifying a Default Button for a Dialog	120
Using Dialogs on Multiple Windows	120
Creating and Deleting Permanent Dialog Copies	121
Internationalization of Dialogs	121
Summary of Dialog Menu Choices	129



Introduction

Using the tools described in this chapter, you can:

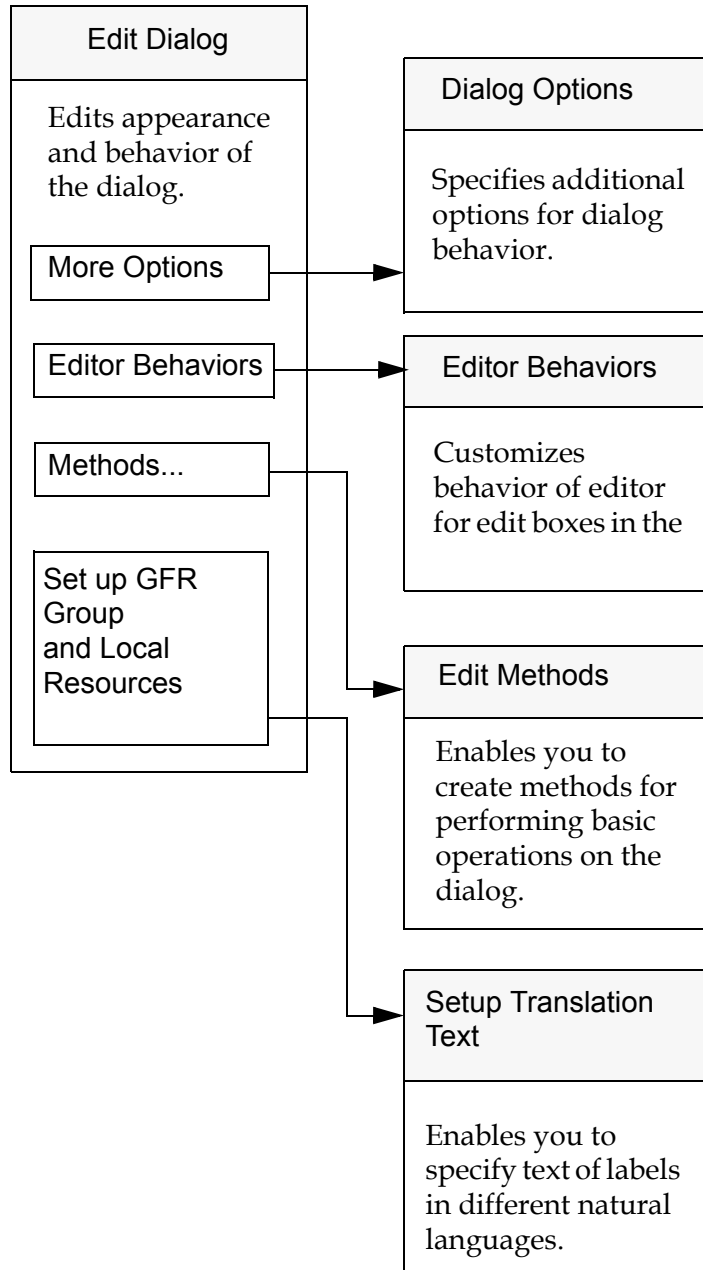
- Edit the appearance and behavior of a master dialog.
- Control dialogs by running procedures called actions on the dialogs.
- Create dialogs that can be opened by clicking on buttons in other dialogs.
- Specify a default button for a dialog.
- Use dialogs on multiple G2 windows.

Editing Master Dialogs

You can edit the appearance and behavior of a master dialog, using the Edit Dialog dialog, Editor Behaviors dialog, Dialog Options dialog, and Edit Methods dialog.

The following figure illustrates the dialogs that you use to edit dialogs and their behaviors:

Dialogs for Editing the Appearance and Behavior of Master Dialogs



Edit Dialog Dialog

To open the Edit Dialog dialog, click the icon for the dialog and choose **edit dialog** from the dialog's menu.

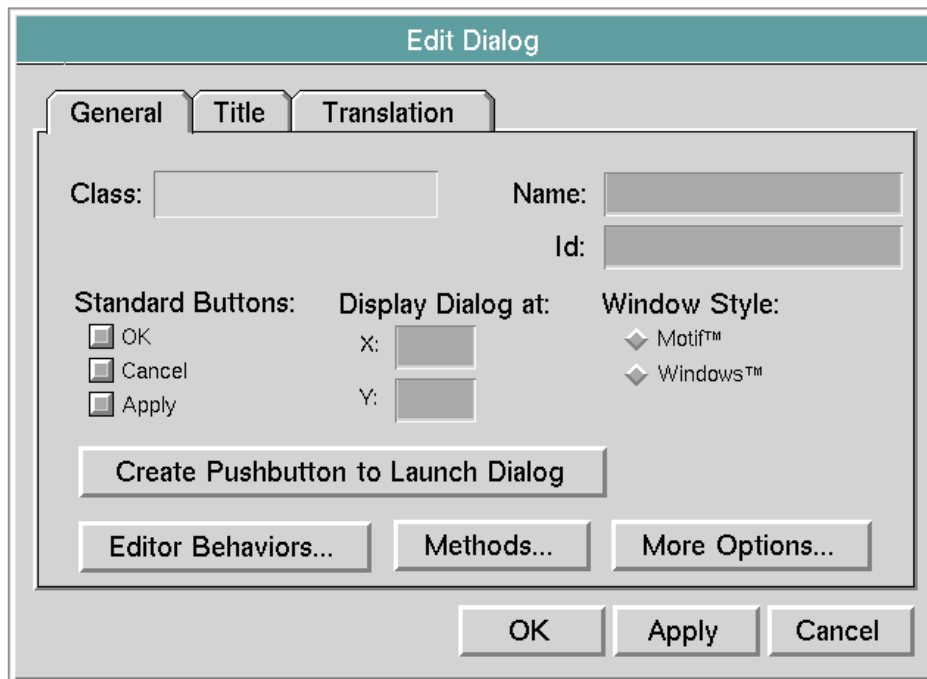
The components of the Edit Dialog dialog are presented on three tab pages:

- **General** – Contains components that you use to specify the class, name, ID, contents, and behaviors of the dialog.
- **Title** – Contains components that you use to specify the title text, justification, size, title bar button, and border display of the dialog title.
- **Translation** – Contains components that you use to specify versions of dialog label text in different natural languages.

To use a tab page, click its tab button to bring it forward so that all its components are visible.

General Tab Page of Edit Dialog Dialog

The General tab page of the Edit Dialog dialog looks like this:



The follow table lists and describes the components on the General tab page:

Components on General Tab Page of Edit Dialog Dialog

Component	Description
Class	(read-only) Displays the class of the dialog that you are editing.
Name	(optional) Displays the current name of the dialog that you are editing. Changing the displayed value updates the name of the dialog. This field does not require a value. Its contents, if any, must be a valid symbolic name.
Id	<p>(optional) Displays the ID of the dialog that you are editing. Changing the ID updates the id attribute of the dialog.</p> <p>Specifying an ID for a dialog provides a way to identify the dialog as the dialog on which to run a set of actions. In the Edit Dialog Actions dialog, one of the options for identifying the target dialog is by its ID.</p>
Standard Buttons	(optional) The toggle buttons below the label Standard Buttons enable you to add standard OK, Apply, and Cancel buttons to the dialog that you are editing.
OK	<p>Selecting this button adds an OK button to the dialog that you are editing. The OK button is preconfigured with the actions <code>uil-call-conclude-method</code>, <code>uil-unsimulate-play-mode</code>, <code>uil-hide-dialog</code>, and <code>uil-release-dialog</code>.</p> <p>If you do not select this button, the OK button is removed from the dialog that you are editing.</p>
Apply	<p>Selecting this button adds an Apply button to the dialog that you are editing. The Apply button is preconfigured with the action <code>uil-call-conclude-method</code>.</p> <p>If you do not select this button, the Apply button is removed from the dialog that you are editing.</p>

Components on General Tab Page of Edit Dialog Dialog

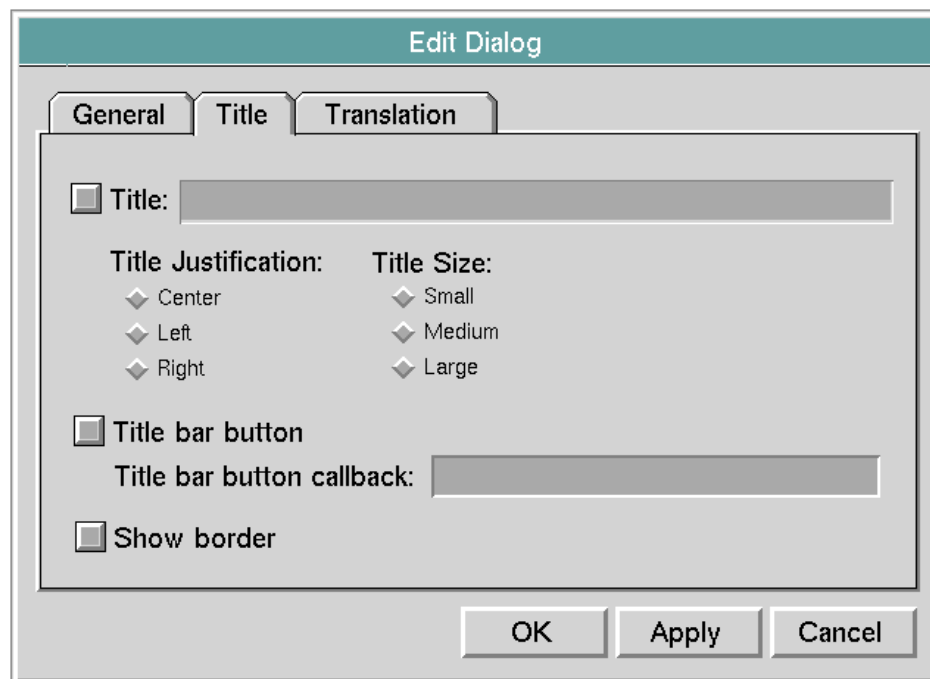
Component	Description
Cancel	<p>Selecting this button adds a Cancel button to the dialog, if the dialog does not already contain a Cancel button. The Cancel button is configured with the default actions <code>uil-unsimulate-play-mode</code>, <code>uil-hide-dialog</code>, and <code>uil-release-dialog</code>.</p> <p>If you turn off this button and click OK or Apply, the Cancel button is removed from the dialog that you are editing.</p>
Display Dialog at	<p>Specify the <code>x-item</code> and <code>y-item</code> coordinates at which the dialog is displayed in the edit boxes labeled X and Y.</p>
Window Style	<p>(required) The radio buttons below the label Window Style (Motif™, Windows™) indicate the current window style used for the graphical objects on the dialog that you are editing. If the dialog contains graphical objects in both styles, neither radio button is selected.</p> <p>You can change the window style of the graphical objects in the dialog by selecting the Motif™ or Windows™ radio button. GUIDE converts all graphical objects in the dialog to the new window style.</p>
Create Pushbutton to Launch Dialog	<p>Clicking this button creates a push button that users can click to open the dialog that you are editing. The push button appears on an automatically created workspace, and can be transferred to any dialog or workspace.</p> <p>By default, the push button is named Launch <i>id</i>, where <i>id</i> is the <i>id</i> of the dialog that you are editing. When a user clicks the button, the actions for updating and displaying a dialog are run. By default, these actions are <code>uil-call-update-method</code>, <code>uil-simulate-play-mode</code>, and <code>uil-show-managed-dialog</code>.</p>

Components on General Tab Page of Edit Dialog Dialog

Component	Description
Editor Behaviors	Clicking the Editor Behaviors button opens the Editor Behaviors dialog. See for more information about this dialog, see Editor Behaviors Dialog .
Methods	Opens the Edit Methods dialog, in which you can create customized methods for the dialog. For information about how to create customized methods, see Methods, Actions, and Callbacks
More Options	Clicking on this button opens the Dialog Options dialog. See the following section for information about this dialog.

Title Tab Page of Edit Dialog Dialog

The Title tab page of the Edit Dialog Dialog looks like this:



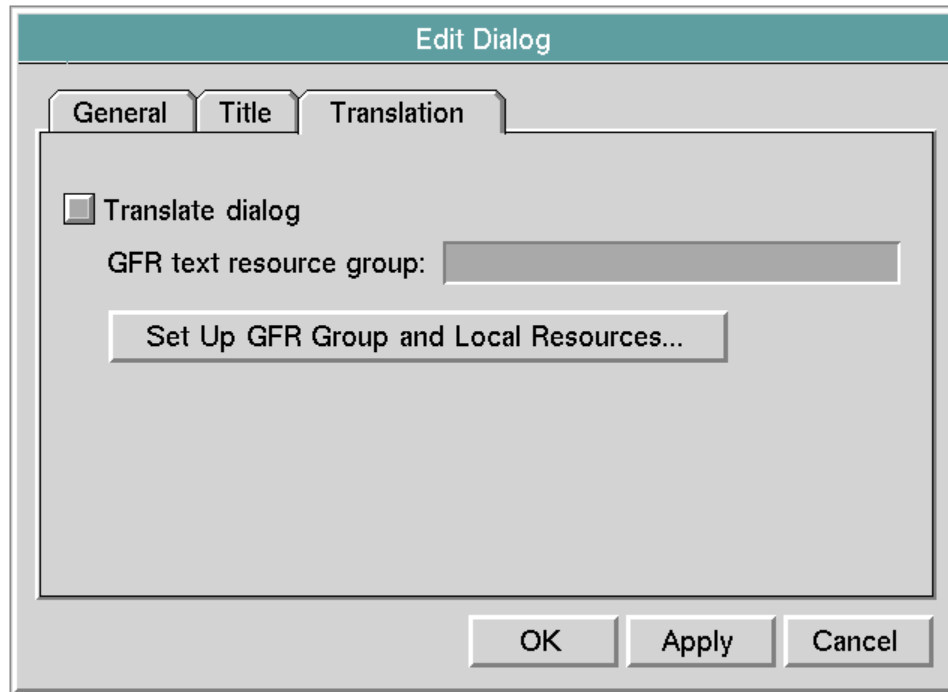
The follow table lists and describes the components on the Title tab page:

Components on Title Tab Page of Edit Dialog Dialog

Component	Description
Title	(optional) Indicates whether or not the dialog that you are editing has a title. If toggled off, the dialog title object is deleted from the dialog that you are editing. If toggled on, a title is created using the text that you enter in the edit box to the right of the title toggle button.
Title Justification	Select the Center , Left , or Right radio buttons to specify how the dialog title is justified in the title bar across the top of the dialog.
Title Size	Select the Small , Medium , or Large radio button to specify the size of the dialog title.
Title bar button	Selecting this button causes the hide button on the dialog title to be displayed.
Title bar button callback	<p>You can specify a non-default callback procedure that is invoked when a user clicks the hide button. If you do not specify a non-default callback procedure, the procedure <code>uil-title-button-callback</code> is invoked when a user clicks the hide button.</p> <p>For information about hide buttons on dialog titles, see Using the Hide Button on Title Bars.</p>
Show border	<p>Selecting this button creates a border and attaches it to the edges of the dialog that you are editing. When a dialog has a border, automatic shrink-wrapping is activated.</p> <p>If you do not select this button, the border is deleted.</p> <p>While borders on dialogs are optional, they are needed if the dialog has a title. The title maintains its position by keying off the location of the anchored dialog border.</p>

Translation Tab Page of Edit Dialog Dialog

The Translation tab page of the Edit Dialog dialog looks like this:



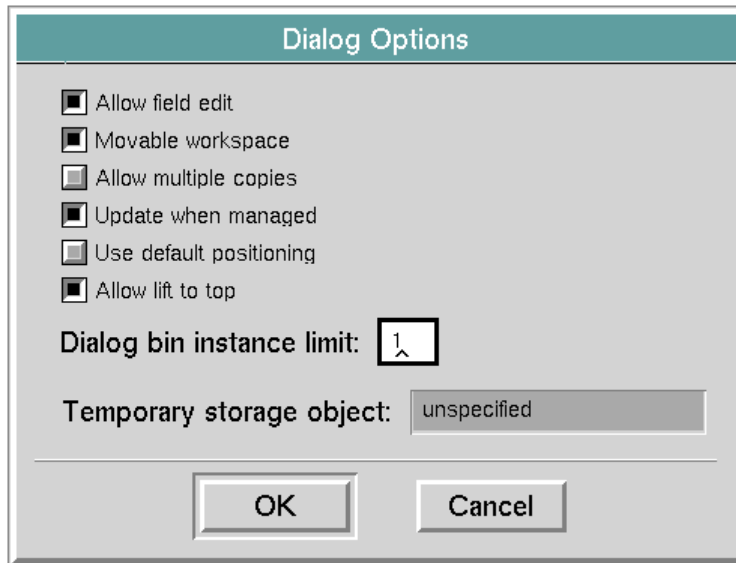
The following table describes the components on the Translation tab page of the Edit Dialog dialog:

Components on Translation Tab Page of Edit Dialog Dialog

Component	Description
Translate dialog	Select this button to enable the following components in the dialog.
GFR text resource group	The GFR text resource group that provide the text in different natural languages for the labels of UIL controls in the master dialog that you are editing.
Set Up GFR Group and Local Resources	Opens the Setup Translation Text dialog. For information about how to use this dialog, see Internationalization of Dialogs .

Dialog Options Dialog

To open the Dialog Options dialog, click the More Options button in the Edit Dialog dialog. The Dialog Options dialog looks like this:



The following table describes the components of the Dialog Options dialog:

Components of Dialog Options Dialog

Component	Description
Allow field edit	(optional) When this button is selected, users can navigate to an edit box in this dialog by pressing the tab, return, or abort key. Navigating to an edit box opens the field editor on it. When this button is not selected, users can open the field editor on this edit box only by clicking on the edit box.

Components of Dialog Options Dialog

Component	Description
Movable workspace	<p>Indicates whether or not users can move this dialog.</p> <p>When the Movable workspace toggle button is off, the position of dialogs is locked. Users cannot reposition them. The only way to dismiss a dialog in the state is to select its OK or Cancel button.</p> <p>When the Movable workspace toggle button is on, users can reposition dialogs by dragging them.</p>
Allow multiple copies	<p>Indicates whether or not the user is allowed to display more than one copy of the master dialog that you are editing at a time on the same G2 window.</p> <p>When the Allow multiple copies toggle button is selected, an unlimited number of copies of the master dialog can be displayed on the same G2 window at the same time.</p> <p>When the Allow multiple copies toggle button is not selected, only one copy of the master dialog that you are editing can be displayed at a time.</p>
Update when managed	<p>If this button is selected, the update method (if any) that is run on this dialog when the dialog is displayed updates all UIL objects on the dialog, regardless of how the UIL objects are configured for updates.</p> <p>If this button is not selected, the update method does not override the update configurations of individual UIL objects on the dialog.</p>

Components of Dialog Options Dialog

Component	Description
Use default positioning	<p>If this option is set to <code>True</code>, the dialog, when first launched, appears at the default screen position indicated by the <code>uil-x-position</code> and <code>uil-y-position</code> attributes (0,0). If this option is set to off, the location at which the dialog was last displayed is written to <code>uil-x-position</code> and <code>uil-y-position</code>, and the dialog is displayed at this position the next time it is launched.</p>
Allow lift to top	<p>If this option is selected, the dialog is lifted above other dialogs or objects when a user clicks on its title or background. If this option is not selected, clicking on the dialog does not lift it.</p>
Dialog bin instance limit	<p>Specifies the maximum number of copies of this dialog that can collect in the dialog bin. Copies can be created in excess of this limit, but they are deleted when they are dismissed, rather than saved in the pool.</p> <p>If Dialog instance pool limit is set to zero, a slight delay occurs each time the dialog is requested. The delay occurs because GUIDE must make a copy of the dialog for each request. The watch cursor is displayed in the upper-left corner until the dialog appears.</p> <p>If Dialog instance pool limit is set to one, the delay occurs only the first time the dialog is accessed. After that, a copy of the dialog is retrieved from the dialog bin and displayed whenever the dialog is requested. This eliminates the delay that occurs when GUIDE must make a copy of the dialog.</p> <p>A good practice is to allow at least as many dialog instances as users of the system.</p>

Components of Dialog Options Dialog

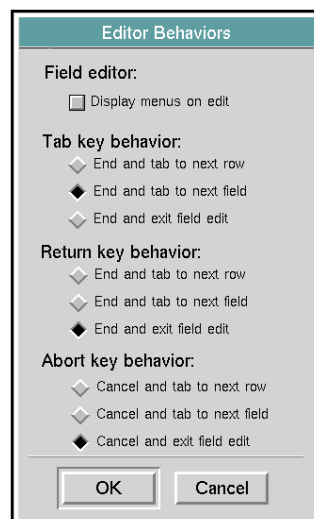
Component	Description
Temporary storage object	<p>(optional) If you are using temporary storage objects with this dialog, enter the name of the class that you have defined for temporary storage objects. Whenever a copy of the dialog that you are editing is reserved, an object of this class is created for the dialog to use as its temporary storage object. The default value is the symbol <code>unspecified</code>. This field does not require a value. Its contents, if any, must be a valid symbolic name.</p> <p>For information about temporary storage objects, see Creating Temporary Storage Objects.</p>

Editor Behaviors Dialog

In the Editor Behaviors dialog, you can specify what happens when a user presses the Tab, Return, and Abort keys while editing an edit box. You can also choose whether or not special editor menus appear when a user edits the edit boxes.

The behaviors that you specify in the Editor Behaviors dialog apply only to edit boxes on the dialog that you are currently editing.

To open the Editor Behaviors dialog, click the Editor Behaviors button in the Edit Dialog dialog. The Editor Behaviors dialog looks like this:



The following table describes the components of the Editor Behaviors dialog:

Components of Editor Behaviors Dialog

Component	Description
Field editor	The toggle button below the label Field editor, Display menus on edit, controls the display of menus in the editor for edit boxes.
Display menus on edit	<p>When the Display menus on edit button is selected, the editor has special editor buttons and menus.</p> <p>This option is useful when edit menus are needed to support internationalization or for other purposes.</p> <p>When the Display menus on edit toggle button is not selected, the editor does not have the special editor buttons and menus.</p>
Tab key behavior	Select one of the radio buttons below the label Tab key behavior to specify what happens when a user clicks the Tab key while editing the contents of an edit box.
End and tab to next row	<p>When this button is selected, clicking the tab key does the following things:</p> <ol style="list-style-type: none">1 Closes the edit session on the current edit box.2 Runs the unselection method, if defined, for the edit box.3 Refocuses the editor to the leftmost edit box in the next row.

Components of Editor Behaviors Dialog

Component	Description
End and tab to next field	<p>When this button is selected, clicking the tab key does the following things:</p> <ol style="list-style-type: none"> 1 Closes the edit session on the current edit box. 2 Runs the unselection method, if defined, for the edit box. 3 Refocuses the editor to the next edit box. <p>The ordering for edit boxes is based on workspace position. Ordering starts at the top left corner and progresses right and downward to the bottom right corner.</p>
End and exit field edit	<p>When this button is selected, clicking the tab key does the following things:</p> <ol style="list-style-type: none"> 1 Closes the edit session on the current edit box. 2 Runs the unselection method, if defined, for the edit box. 3 Attempts to run the actions associated with the default button on the dialog. If there is no default button, then the editor is shut down.
Return key behavior	<p>Select one of the radio buttons below the label Return key behavior to specify what happens when a user presses the Return key while editing the contents of an edit box.</p>
End and tab to next row	<p>When this button is selected, pressing the Return key does the following things:</p> <ol style="list-style-type: none"> 1 Closes the edit session on the current edit box. 2 Runs the unselection method, if defined, for the edit box. 3 Refocuses the editor to the leftmost edit box in the next row.

Components of Editor Behaviors Dialog

Component	Description
End and tab to next field	<p>When this button is selected, pressing the Return key does the following things:</p> <ol style="list-style-type: none">1 Closes the edit session on the current edit box.2 Runs the unselection method, if defined, for the edit box.3 Refocuses the editor to the next edit box. <p>The ordering for edit boxes is based on workspace position. Ordering starts at the top left corner and progresses right and downward to the bottom right corner.</p>
End and exit field edit	<p>When this button is selected, pressing the Return key does the following things:</p> <ol style="list-style-type: none">1 Closes the edit session on the current edit box.2 Runs the unselection method, if defined, for the edit box.3 Attempts to run the actions associated with the default button on the dialog. If there is no default button, then the editor is simply shut down.
Abort key behavior	<p>Select one of the radio buttons below the label Abort key behavior to specify what happens when a user presses the Abort key (Ctrl + a) while editing the contents of an edit box.</p>
Cancel and tab to next row	<p>When this button is selected, pressing the abort key does the following things:</p> <ol style="list-style-type: none">1 Closes the edit session on the current edit box.2 Runs the unselection method, if defined, for the edit box.3 Refocuses the editor to the next edit box in the next row.

Components of Editor Behaviors Dialog

Component	Description
Cancel and tab to next field	<p>When this button is selected, pressing the Abort key (Ctrl + a) does the following things:</p> <ol style="list-style-type: none"> 1 Closes the edit session on the current edit box. 2 Runs the unselection method, if defined, for the edit box. 3 Refocuses the editor to the next edit box. <p>The ordering for edit boxes is based on workspace position. Ordering starts at the top left corner and progresses right and downward to the bottom right corner.</p>
Cancel and exit field edit	<p>When this button is selected, pressing the Abort key (Ctrl + a) does the following things:</p> <ol style="list-style-type: none"> 1 Closes the edit session on the current edit box. 2 Runs the unselection method, if defined, for the edit box. 3 Attempts to run the actions associated with the default button on the dialog. If there is no default button, then the editor is shut down.

Controlling Dialogs with Actions

Actions are procedures that perform operations on a dialog, such as opening or closing the dialog, or updating or concluding the values in it.

UIL provides system-defined actions that perform all the commonly required operations on dialogs. You can create customized actions to perform any specialized operations that your application requires.

You can run actions on dialogs through push buttons or by invoking the procedure `uil-control-dialog-callback`.

When action buttons and procedures invoke `uil-control-dialog-callback`, this procedure references a named action description array. An **action description array** is an object that stores a list of actions. The procedure runs the actions listed in the action description array on a specified dialog. For information about how to

invoke `uil-control-dialog-callback` from an action button or procedure, see [Launching Dialogs](#).

`uil-control-dialog-callback` is the default callback of push buttons. However, the actions that a push button runs on a dialog are stored in the `uil-action-description` attribute of the push button, rather than in an action description array.

Specifying the Actions Run by a Push Button

The OK, Apply, and Close push buttons that appear on a master dialog by default reference appropriate sets of actions. Your application can use the OK, Apply, and Close push buttons with their default actions for almost all purposes.

You can change the set of actions run by any push button, using the Edit Dialog Actions dialog, which edits list of actions referenced by the `uil-action-description` attribute of the push button. For information about how to use this dialog, see [Edit Dialog Actions Dialog](#).

To edit the actions run by a push button:

- 1 Select **edit array** from the menu of the push button.
This opens the Edit Array dialog, in which you can specify the actions that you want to run.
- 2 Click `uil-action-description` in the Arrays scroll area in the Edit Array dialog.
The actions currently referenced by the `uil-action-description` attribute are listed in the Elements scroll area in the Edit Array dialog. You edit this list to specify which actions are run when a user clicks on the push button.
- 3 To add an action to `uil-action-description`, enter its name in the Element field and click Apply.
- 4 To delete an action from `uil-action-description`, select the name of the action in the Elements scroll area and click the Remove button.
- 5 Click OK to apply your changes and close the Edit Array dialog.

Note To use the Edit Array dialog to associate actions with a push button, you must know the names of all the actions that you want to use. The Edit Array dialog does not prompt you with lists of existing actions or provide any other help.

Creating an Action Description Array

To create an action description array:

- 1 Select the following choice from the GUIDE menu bar:

Item > GUIDE Objects > Action Array

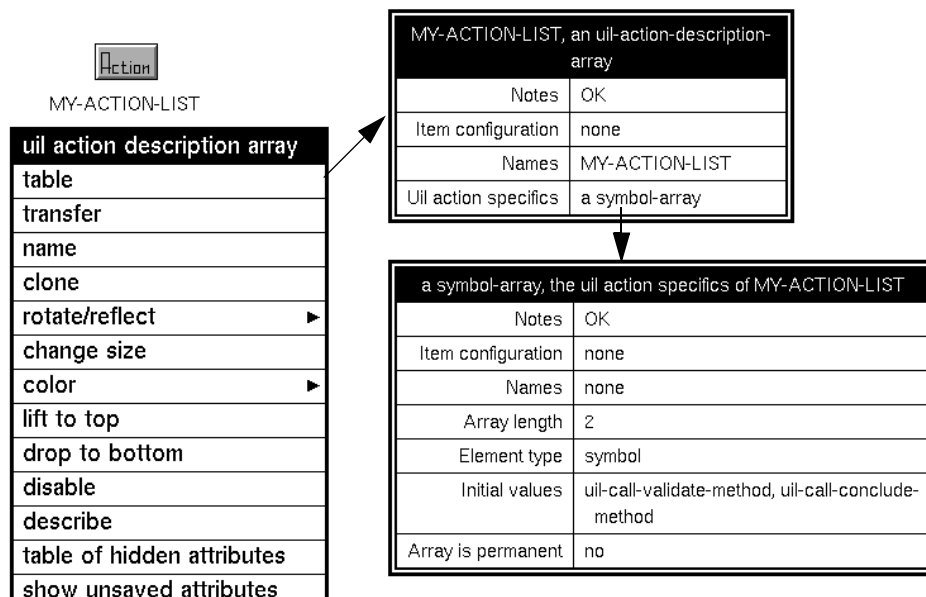
The icon for the action description array becomes attached to your cursor. You can drop it on any workspace. The action description array looks like this:



You can also add action description arrays by:

- Clicking on the More Options button on the G2 GUIDE palette to open the More Options palette. In the More Options palette, click the Action Description Array icon. Drag the icon to the workspace where you want to create the action description array, and click again.
- Opening the KB Workspace menu on a workspace and choosing:
 - New Object > choose a class > uil-action-description-array

Each action description array has an attribute named `uil-action-specifics`, which points to a symbol array that stores the names of the actions. The actions are listed in the `initial-values` attribute of the symbol array. The following figure illustrates an action description array, its table, and the table of the symbol array stored in the `uil-action-specifics` attribute.



- 2 You can edit an existing action description array by choosing edit action description from its menu.

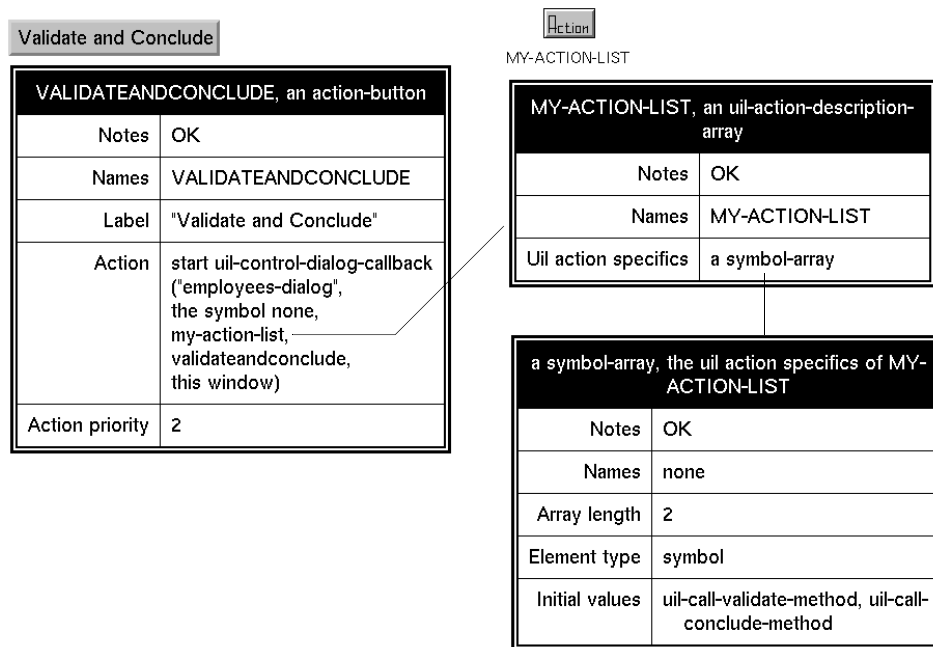
This opens the Customize Dialog Actions dialog, in which you can specify the actions that are included in the action description array. For information about this dialog, see [Customize Dialog Actions Dialog](#).

Invoking uil-control-dialog-callback

Once you have defined an action description array, you can run the actions in the action description array using the procedure uil-control-dialog-callback.

You can invoke uil-control-dialog-callback from an action button or a procedure.

The following figure illustrates an action button, Validate and Conclude, that starts uil-control-dialog-callback to run the actions in my-action-list on a dialog whose ID is employees-dialog:



For a complete description of uil-control-dialog-callback, see the *G2 GUIDE/UII Procedures Reference Manual*.

Creating Systems of Cascaded Dialogs

It is often convenient or necessary for users to be able to open one dialog from within another dialog. Dialogs that are related to each other in this way are said to be **cascaded**. GUIDE enables you to combine the dialogs that you create into systems of cascaded dialogs.

When one dialog is opened from within another dialog, the first open dialog is known as the **parent dialog**, and the dialog that is opened from within the parent dialog is called the **child dialog**.

Users can open child dialogs from within their parent dialogs by clicking push buttons. Dismissing a parent dialog triggers the actions associated with the child dialog's Cancel button. To ensure that changes made on a child dialog are applied, a user can dismiss a child dialog by selecting the equivalent of its OK button before dismissing the parent dialog.

The **initiating object** of a dialog is the object that launches the dialog. For example, when a user chooses a menu choice from the menu of the G2 object, that object becomes the initiating object of the dialog. In a series of cascaded dialogs, all dialogs have the same initiating object – the object that launches the first dialog in the series.

To close a child dialog and return control to its parent dialog, users click a push button with standard OK or Cancel functionality.

To make it possible to open a dialog by clicking a push button, you edit the push button to specify that:

- The actions associated with the push button are directed to the dialog that you want to open
- The actions associated with the push button include opening and displaying this target dialog

For information about how to do this, see [System-Defined Actions for Dialog Processing](#). There is no limit on the number of child dialogs that a parent dialog can have. However, a child dialog can have only one parent. A system of cascaded dialogs can include as many levels as your G2 application requires.

Note It is not recommended practice to use a navigation button to open a dialog. A dialog that is opened by a navigation button does not have full GUIDE support for field editing, automatic updating and concluding of values, and other essential features of the GUIDE dialog system.

Automatically Creating a Push Button for Starting a Dialog

You can create a push button for starting a dialog by clicking Create a Pushbutton to Launch Dialog button in the Edit Dialog dialog. This creates a push button that is automatically configured to start the dialog that you are editing.

You can transfer this push button to the subworkspace of a different master dialog.

To transfer a push button to a master dialog, follow these steps:

- 1 Select transfer from the menu of the push button.
- 2 Move the pointer to the subworkspace of the master dialog to which you want to add the push button.
- 3 Click to drop the push button on the subworkspace.

Relations Among Cascaded Dialogs

For each UIL object, GUIDE provides a set of relations that define the possible relationships between that UIL object and other UIL objects. These relations make it possible for users to traverse the parent and child hierarchy of cascaded dialogs, for data to pass back and forth among dialogs, and for users to access UIL objects procedurally.

For information about the relations among cascaded dialogs, see the *G2 GUIDE/UIL Procedures Reference Manual*.

Specifying a Default Button for a Dialog

You can designate a push button on each dialog to be the **default button** for that dialog. The default button is activated when the user presses the Return key.

For information about how to designate a push button as the default button for a dialog, see [Edit Pushbutton Dialog](#).

Using Dialogs on Multiple Windows

Users sharing a knowledge base can open different copies of the same GUIDE dialog on different G2 windows. UIL provides API calls for dialogs and their windows. All API calls contain a window argument enabling GUIDE to determine the exact dialog to use in the procedure.

Each time a dialog is reserved, a copy is retrieved from the dialog bin or generated by cloning the master if no copies are available. This dialog is then assigned to the window specified in the window argument of the reserve dialog operation. GUIDE automatically associates the dialog with this window through the relation `a-uil-dialog-presently-in-use-on-window`.

If, after the dialog is released, it is accessed by a different user, and a different window argument is passed into the reserve dialog operation, the dialog is re-assigned to the new window.

Creating and Deleting Permanent Dialog Copies

You can create and delete permanent dialog copies using the GUIDE menu choices:

- Tools > Create Permanent Dialog Copies
- Tools > Delete Permanent Dialog Copies

These menu choices enable you to create and delete permanent copy dialogs that are remain in your GUIDE application when you reset or save your KB.

For more information about how GUIDE uses dialog copies to provide more efficient dialog of dialogs, see [Launching Dialogs](#).

Internationalization of Dialogs

GUIDE/UIIL supports the internationalization of dialogs through the use of G2 Foundation Resources (GFR). Through GFR, GUIDE can maintain different natural language versions of the text in UIL controls on dialogs, making it easy to change the text displayed to any desired natural language.

GUIDE can translate the text in UIL objects of the classes `uil-grmes` or `uil-button`, or any subclass of these buttons. These classes include the following UIL controls:

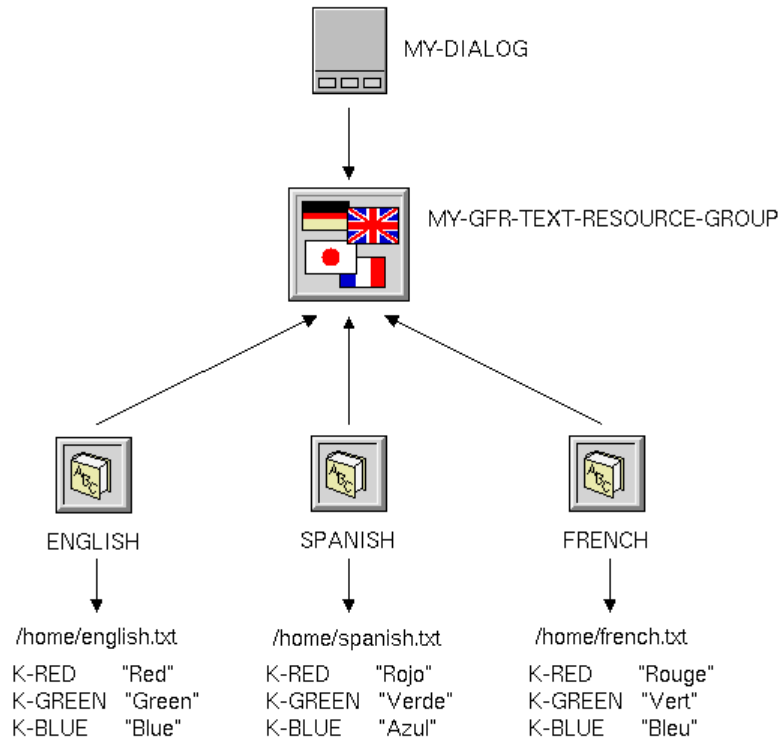
- Edit boxes
- Text objects
- Message objects
- All buttons

GFR Objects that Support Internationalization

To support internationalization, GUIDE uses two classes of GFR objects: GFR Text Resource Groups (`gfr-text-resource-group`) and GFR Local Text Resources (`gfr-local-text-resource`):

- A GFR Text Resource Group serves as a unifying point for all the possible language versions. The GFR Text Resource Group used by a dialog is specified in the `uil-gfr-text-resource-group` attribute of the dialog.
- A GFR Local Text Resource contains label text in a specific language and the name of a `gfr-text-resource-group` with which it is associated.

For example, if your application supports English, Spanish and French users, you need at least one `gfr-text-resource-group` and three `gfr-local-text-resource` objects – one for the English text, one for the Spanish, and one for the French:




The dialog `my-dialog` references a `gfr-text-resource-group`, which in turn is referenced by each of the three `gfr-local-text-resource` objects. Each `gfr-local-text-resource` object holds the name of a file that contains a textual lookup table, indexed by the same constants used in the `uil-label-constant` attributes of the UIL objects to be translated.

For example, if a UIL object references the constant `k-red` in its `uil-label-constant` attribute, that UIL object displays the text “Red”, “Rojo”, or “Rouge”, depending on whether English, Spanish, or French is the currently chosen natural language.


The following figure illustrates the tables for the GFR objects:

SAMPLE-RESOURCE-GROUP



SAMPLE-RESOURCE-GROUP, a gfr-text-resource-group	
Notes	OK
Names	SAMPLE-RESOURCE-GROUP
Gfr version	"6.0"
Gfr default language	english
Gfr use default language	true

spanish



a gfr-local-text-resource	
Notes	OK
Names	none
Gfr language	spanish
Gfr resource group	sample-resource-group
Gfr version	"6.0"
Gfr file location	"/home/ex-spn.txt"
Gfr preload resource	true

The file `/home/ex-spn.txt` contains the following:

```
EXAMPLE-GROUP-RESOURCE
7.0
SPANISH
K-HOUSE, "Casa"
K-COLOR, "Color"
K-BLUE, "Azul"
K-GREEN, "Verde"
K-RED, "Rojo"
```

UIL Object Attributes that Support Internationalization

Dialogs and text objects have attributes that support the internationalization of dialogs:

- `uil-label-constant`, an attribute of the classes `uil-text` and `uil-button`, and any of their subclasses. Thus, text objects and all buttons have this attribute. The `uil-label-constant` attribute stores a symbol used as a look-up constant.
- `uil-translate-dialog`, an attribute of dialogs. This attribute contains a truth-value, which when set to `true` causes UIL to attempt to translate the text on the dialog before displaying it to the user. UIL performs the translation during execution of the `uil-show-managed-dialog` action.

- `uil-gfr-text-resource-group`, an attribute of dialogs. This attribute specifies the `gfr-text-resource-group` used by this dialog to support internationalization.

How the Translation Works

GFR and GUIDE follow these steps to translate a dialog when the dialog is launched:

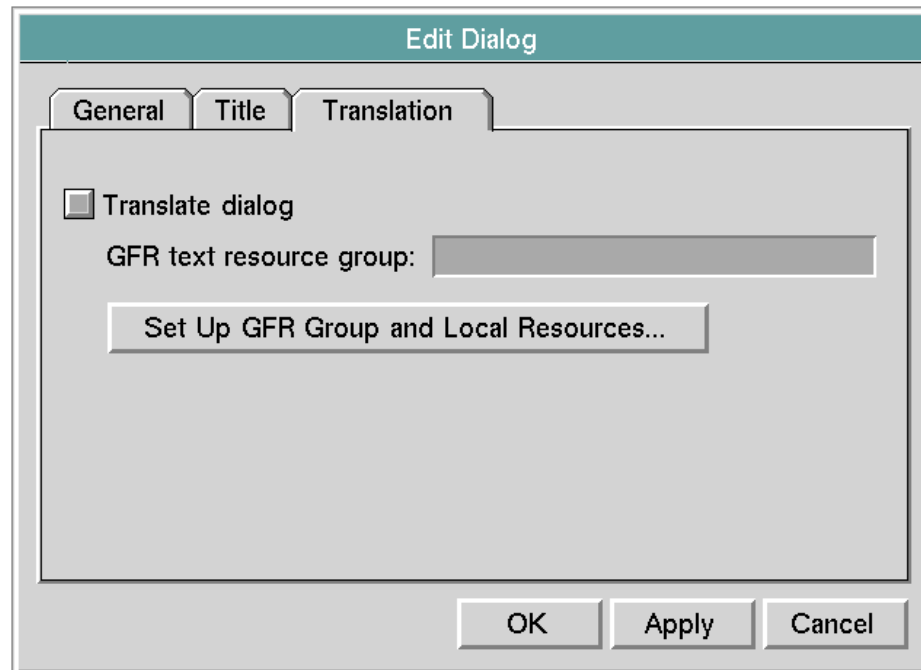
- 1 Find the user's current language.
 - a Look at the `g2-window-specific-language`, which you can change when you enter `Ctrl + y`.
 - b If the `g2-window-specific-language` is none, look at the top-level KB's current language, which you can change in the Language Parameters system table. G2 must know about this language; by default, it is English.
- 2 Use the current language to translate the dialog.
 - a Find the `gfr-text-resource-group` associated with the dialog.
 - b Search all `gfr-local-text-resources` associated with the `gfr-text-resource-group`, until you find one that matches the current language.
 - c If no match exists, check the `use-default-language` flag on the `gfr-text-resource-group`. If the flag is true, use the local resource that matches the default language in the `gfr-text-resource-group`; if it is false, signal an error.
 - d Using the matched `gfr-local-text-resource`'s lookup table, find text strings for each `uil-text` upon the dialog.

The UIL procedure `uil-translate-dialog` handles the translation of the text of dialogs. For information about this procedure, see the *G2 GUIDE/UIL Procedures Reference Manual*.

Creating GFR Objects to Support Internationalization

GUIDE provides editors for creating `gfr-text-resource-group` objects and `gfr-local-text-resource` objects for your application to use.

The Edit Dialog dialog contains options that access the attributes you need for translation, using GFR:



The options on the Edit Dialog dialog that support internationalization are:

- A toggle button labeled Translate dialog
- An edit box labeled GFR text resource group
- A push button labeled Set Up Gfr Group and Local Resources

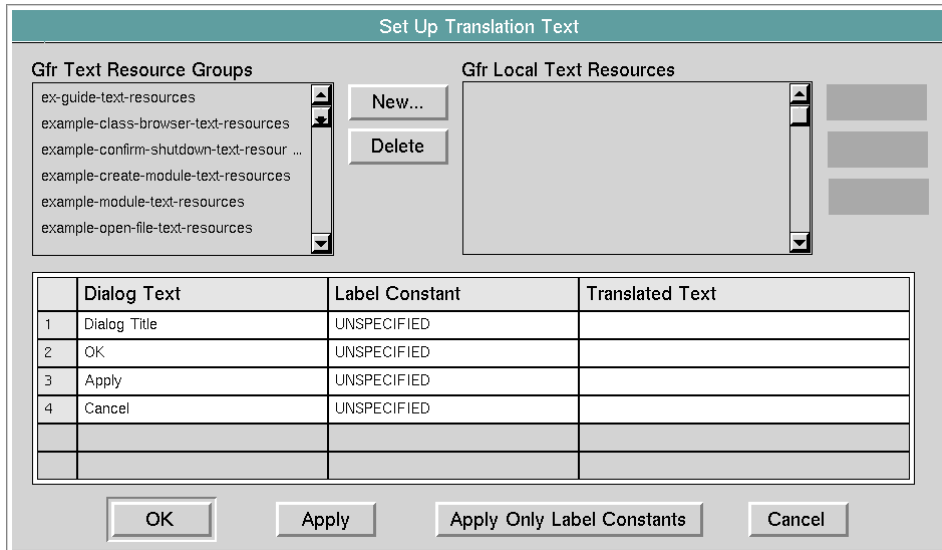
When you toggle off the Translate dialog toggle button, you disable the Gfr text resource group edit box and the Setup Gfr Group and Local Resources pushbutton. The Translate dialog attribute must be set to true (toggle on) to enable UIL to perform translation.

Make sure the Gfr text resource group edit box contains the name of a `gfr-text-resource-group` object. You can create and edit `gfr-text-resource-group` objects and `gfr-local-text-resource` objects from the Setup Translation Text dialog that you launch by clicking the Set Up Gfr Group and Local Resources button.

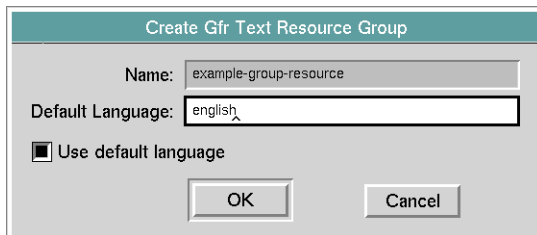
The Setup Translation Text Dialog enables you to select the GFR local text resource that provides the label text for a master dialog.

To open the Setup Translation Text Dialog:

- 1 Click the Translate dialog button in the Edit Dialog dialog.
- 2 Click the Set Up GFR Group and Local Resources button:



Click the New button directly to the right of the scroll area labeled Gfr Text Resource Groups to launch a dialog that enables you create a new `gfr-text-resource-group`:



After you type in the name for the new `gfr-text-resource-group`, and specify the default language and whether or not to use the default language if the specified language is not found, click OK to create a `gfr-text-resource-group` object, and place it just above the dialog being edited by the Edit Dialog dialog. The name of the `gfr-text-resource-group` is updated into the scroll area labeled Gfr Text Resource Groups on the Setup Translation Text dialog.

Note Selecting a `gfr-text-resource-group` from the GFR Text Resource Groups scroll area fills in the scroll area labeled Gfr Local Text Resources with entries for each `gfr-local-text-resource` that is associated with the selected `gfr-text-resource-group`. Clicking the New button directly to the right of the scroll area labeled Gfr Local Text Resources, launches a dialog that handles creating new `gfr-local-text-resource` objects.

After you type in the language for the `gfr-local-text-resource`, and specify the file location and whether or to preload the resource when starting G2, select OK to create a `gfr-local-text-resource` object and place it above and to the right of the dialog being edited by the Edit Dialog dialog. The file location and language information is concatenated and added to the scroll area labeled Gfr Local Text Resources on the Setup Translation Text dialog.

Selecting a message in the Gfr Local Text Resources scroll area enables the three scroll areas and edit boxes on the lower half of the Setup Translation Text dialog.

- The scroll area labeled Dialog Text fills in with the labels or message contents of `uil-grmes` objects that can be translated.
- The scroll area labeled Label Constant fills in with the value found in the `uil-grmes` attribute, `uil-label-constant`. By default, this value is unspecified.
- The scroll area labeled Translated Text fills in when a match is found for the label constant in the selected `gfr-local-text-resource`.

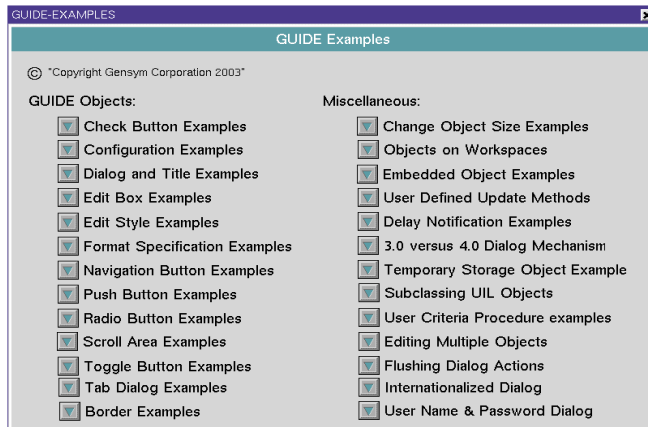
To modify label constant and translated text values:

- ➔ Type the new text into the edit box directly above the scroll area and then do *one* of the following:
- Select the icon button with the down arrow to the far left of the row of edit boxes.
 - Tab out of the edit box.
 - Press the Return key.

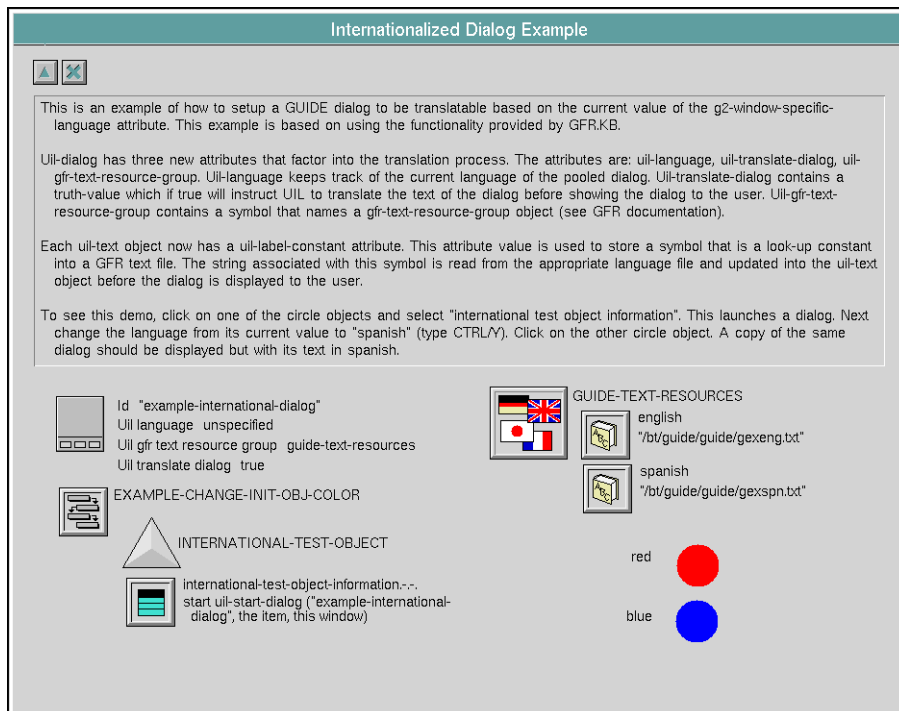
Click the Edit button directly to the right of the scroll area labeled Gfr Local Text Resources to launch the Edit Gfr Local Text Resource dialog. This dialog enables

you to modify the attributes of the `gfr-local-text-resource`. For information about how to use this dialog, see the *G2 Foundation Resources User's Guide*.

`guidemo.kb` contains a working example of a dialog that can be translated from English to Spanish:



Select the navigation button labeled Internationalized Dialog to take you to the working example:



For more information about current GFR functionality, see the *G2 Foundation Resources User's Guide*.

Summary of Dialog Menu Choices

Menu Choice	Description
table	Shows the G2 table for the dialog.
go to subworkspace	Opens the subworkspace of this dialog. The subworkspace is the dialog that users see and use.
transfer	Places the dialog on the mouse, so that you can drop it on a different workspace.
name	Opens an editor in which you can specify or edit a name for this dialog.
rotate/reflect	Opens a menu of rotation and reflection options: 90 clockwise, 90 counterclockwise, 180, left-right reflection, up-down-reflection.
change size	Opens a G2 dialog providing change size options.
color	Opens a series of G2 menus enabling you to change colors of regions of the dialog icon. GUIDE provides the Configuration Editor as the recommended way to change icon regions.
lift to top	Opens a dialog on top of another object on a workspace.
drop to bottom	Drops a dialog behind another object on a workspace.
disable/enable	Disables or enables the dialog.
describe	Shows a description of the object and its relations.
go to master dialog object	Closes this menu and returns focus to the master dialog.
edit list	Do not select this menu choice.
edit configuration	Opens the GUIDE Configuration Editor. For information about this editor, see Using The GUIDE Configuration Editor .

Menu Choice	Description
configure	Applies the current configuration to the dialog. This menu choice is visible only when the dialog is disabled.
delete.	Opens a confirmation dialog to the user and if confirmed, deletes the dialog. This menu choice is visible only when the dialog is enabled.
move	Opens the Move Object dialog, which enables you to move the object precisely. This menu choice is visible only when the dialog is enabled.
clone.	Makes a copy of the dialog and places it next to the original dialog on the workspace. This menu choice is visible only when the dialog is enabled.
go to dialog bin	Shows the workspace used to store dialog copies of the dialog. This menu choice is visible only when the dialog is enabled.
highlight copies	Highlights copies of the dialog in the dialog bin or on other workspaces to distinguish them from master dialogs. This menu choice is visible only when the dialog is enabled.
delete copies	Clears copies of dialog from dialog bin. This menu choice is visible only when the dialog is enabled.
initialize	Calls the initialization method of the dialog. This menu choice is visible only when the dialog is enabled.
show dialog	Displays the subworkspace of this master dialog. This menu choice is visible only when the dialog is enabled.
edit dialog	Open the Edit Dialog dialog, in which you can edit the various attributes of the dialog. This menu choice is visible only when the dialog is enabled.
rotate tab pushbuttons	Rotates the placement of tab push buttons 90 degrees clockwise (from top of tab page, to right side, to bottom, to left side, and so on).

Launching Dialogs

Describes how to launch a dialog from an action button, a user-menu choice, a user-defined procedure, a push button, or a rule.

Introduction	131
Pooling Reusable Dialogs for Quick Retrieval	132
Procedures that Launch Dialogs	133
uil-control-dialog-callback	135
uil-start-dialog	137
uil-start-or-refocus-dialog	138
uil-start-dialog-processing	139
Launching a Dialog from an Action Button	140
Launching a Dialog from a User-Defined Procedure	142
Launching a Dialog from a User Menu Choice	145
Launching a Dialog from a Push Button	147
Launching a Dialog from a Rule	148



Introduction

Users can launch a dialog from an action button, a user menu choice, a user-defined procedure, a push button, or a rule. All these techniques invoke system-defined procedures to launch dialogs. The procedures specify the dialogs to launch by their IDs.

Pooling Reusable Dialogs for Quick Retrieval

If you want to use dialogs in your user interface, you must create templates, or master dialogs. The dialogs that users see are **copy dialogs**, which GUIDE creates by cloning master dialogs. All the UIL controls that you add to a master dialog, such as edit boxes, scroll areas, and push buttons, are reproduced in each copy that GUIDE makes of the master.

Using Dialogs in the Dialog Bin

GUIDE stores copy dialogs in a workspace known as the **dialog bin**. The dialog bin is designed to enable G2 applications to retrieve copy dialogs quickly and efficiently. To view the dialogs in the dialog bin, select View > Dialog Bins > GUIDE from the GUIDE menu bar.

When a G2 application requests a dialog, for example, when a user clicks on a button or selects a user menu choice designed to open the dialog, GUIDE looks in the dialog bin for a copy of the requested master dialog. If GUIDE finds a copy, it displays it. If GUIDE does not find a copy, it clones the master dialog and displays the resulting copy. GUIDE creates additional copies of a master dialog only when there are no copies in the dialog bin to meet requests for dialogs.

The process of creating and displaying a copy dialog, or displaying an existing copy dialog, is called **reserving** a dialog.

When a user closes a dialog – for example, by clicking the Cancel or OK button on the dialog – GUIDE places the copy in the dialog bin. Your G2 application can reuse these copies from the dialog bin whenever it needs them.

Users can open copies of the same master dialog on any number of different G2 windows that are running on the same knowledge base.

Copies of a single master dialog can accumulate in the dialog bin until a limit is reached. This limit is specified by an attribute of the master dialog (`uil-dialog-instance-pool-limit`). Copies can be created and displayed in excess of this limit, but they are deleted when users dismiss them, rather than being saved in the bin. For information about how to set this limit for a master dialog, see [Dialog Options Dialog](#).

By default, copies of a master dialog are permanent. **Permanent copies** are not deleted when you restart or reset G2. A permanent copy is the first copy to be used when a dialog is reserved. If no permanent copies are available to meet a request, additional copy dialogs are cloned from the master dialog.

Note Permanent copies are deleted if their subworkspaces are displayed when you reset G2. To preserve the permanent copies, close them before you reset.

Each master dialog has an attribute named `uil-origin` that is automatically set to the symbol `master`. Each copy dialog has an attribute named `uil-origin` that is automatically set to the symbol `permanent`.

Releasing and Returning Dialogs

When an application finishes using copy dialogs, it can either release or return the dialogs.

Releasing a copy dialog returns the copy dialog to the dialog bin, so that it becomes available for use by other users. To release a copy dialog, run the `uil-release-dialog` action on the dialog.

Returning a copy dialog returns control of the dialog to the procedure that launched the dialog, but does not return the dialog to the dialog bin. You can return a copy dialog when the procedure that launched the copy needs to perform processing on the contents of the copy before allowing other users to use the copy. To return a dialog, run the `uil-return-dialog` action on the dialog. The procedure to which the copy is returned should itself call `uil-release-dialog` to release the copy when it finishes processing the copy.

For more information about how to run actions on dialogs, see [Push Buttons](#).

Creating and Deleting Permanent Copies

You can create and delete permanent copies of all your master dialogs by selecting the following choices from the GUIDE menu bar:

Tools > Create Permanent Dialog Copies > GUIDE
Tools > Delete Permanent Dialog Copies > GUIDE

You can also delete permanent copies of a master dialog by choosing `delete copies` from the menu of the master dialog.

Procedures that Launch Dialogs

Every technique for launching a dialog must invoke the procedure `uil-control-dialog-callback` either directly, or indirectly through calls to other procedures. The procedure `uil-control-dialog-callback` is responsible for displaying a copy of the master dialog that you launch. The process of creating and displaying a copy dialog, or displaying an existing copy dialog, is called **reserving** a dialog.

Push buttons invoke `uil-control-dialog-callback` directly.

Action buttons, user menu choices, and user-defined procedures can invoke `uil-control-dialog-callback` directly, or indirectly through calls to the following procedures:

- `uil-start-dialog`, which passes a system-defined set of actions to `uil-control-dialog-callback` for launching the dialog.
- `uil-start-or-refocus-dialog`, which passes a set of actions to `uil-control-dialog-callback` for launching the dialog or for updating the dialog if it is already displayed.
- `uil-start-dialog-processing`, which can run different sets of actions on dialogs, depending on whether it launches a dialog or refocuses a dialog that is already displayed.

The following sections describe these procedures.

uil-control-dialog-callback

Reserves a dialog and runs actions on the dialog.

Synopsis

uil-control-dialog-callback

(*target-object*: item-or-value, *target-attribute*: symbol,
action-array: item-or-value, *initiating-item*: item-or-value,
window: class g2-window

-> {*dialog*: item-or-value | dialog-not-found: symbol},
 {*button*: item-or-value | none: symbol}

Argument	Description
<i>target-object</i>	<p>The dialog on which actions are run. Specify either:</p> <ul style="list-style-type: none"> • A text value representing a valid dialog ID, enclosed in quotation marks. • the symbol <code>host-dialog</code> <p>Specify the symbol <code>host-dialog</code> if you are invoking <code>uil-control-dialog-callback</code> from a button on a dialog and you want to run actions on the dialog that contains the button.</p>
<i>target-attribute</i>	Not used. Specify the symbol <code>none</code> .
<i>action array</i>	<p>An action description array that specifies the actions to run on the dialog. For information about how to create action description arrays, see Creating an Action Description Array.</p>

Argument	Description
<i>initiating-item</i>	The object that launches this dialog (for example, an action button, push button, or user menu choice), or the symbol <code>none</code> . You can specify this item if you invoke <code>uil-control-dialog-callback</code> from a user menu choice.
<i>window</i>	The G2 window on which the dialog is managed. You can specify this window if you invoke <code>uil-control-dialog-callback</code> from a user menu choice. Note: This procedure creates an inoperable dialog if <i>window</i> specifies an invalid G2 window. An invalid G2 window is one whose <code>g2-connection-status</code> is <i>not</i> equal to the symbol <code>connected</code> .

Return Values	Description
<i>dialog</i>	The dialog or the symbol <code>dialog-not-found</code> .
<i>button</i>	The button that was activated on the dialog, or the symbol <code>none</code> .

Description

`uil-control-dialog-callback` reserves the dialog with a specified ID and sets up an event loop for the dialog. The event loop processes any actions that are called to run on the dialog. Actions perform operations such as showing the dialog and updating its values.

You can specify actions in the call to `uil-control-dialog-callback`, or pass actions to `uil-control-dialog-callback` through other procedures that invoke `uil-control-dialog-callback`.

`uil-control-dialog-callback` analyzes the button's `uil-target-object`, which is either a dialog name, ID, or the symbol `host-dialog`, and adds its actions into the event handler for the target dialog. If no event handler is active, it starts one.

Caution An action fails if it attempts to invoke `uil-control-dialog-callback` on the target dialog of the push button that invokes the action itself.

uil-start-dialog

Launches a dialog, using a simplified set of arguments.

Synopsis

uil-start-dialog

(*dialog-id*: text, *initiating-item*: item-or-value,
window: class g2-window)

-> {*dialog*: item-or-value | dialog-not-found: symbol},
 {*button*: item-or-value | none: symbol}

Argument	Description
<i>dialog-id</i>	The ID of the dialog to display.
<i>initiating-item</i>	The item responsible for launching the dialog or the symbol none. You can specify this item if you invoke uil-start-dialog from a user menu choice.
<i>window</i>	The window on which to manage the dialog. You can specify this window if you invoke uil-start-dialog from a user menu choice.
Return Value	Description
<i>dialog</i>	The dialog, or the symbol dialog-not-found.
<i>button</i>	The button that was activated on the dialog, or the symbol none.

Description

This procedure invokes uil-control-dialog-callback to display the dialog whose ID matches the specified *dialog-id* on the specified *window*.

uil-start-dialog passes the following actions to uil-control-dialog-callback:

- uil-call-update-method
- uil-simulate-play-mode
- uil-show-managed-dialog

uil-start-or-refocus-dialog

Launches or updates a dialog.

Synopsis

uil-start-or-refocus-dialog

(*dialog-id*: text, *initiating-item*: item-or-value,

window: class g2-window)

-> {*dialog*: item-or-value | dialog-not-found: symbol},

{*button*: item-or-value | none: symbol}

Argument	Description
<i>dialog-id</i>	The ID of the dialog to display.
<i>initiating-item</i>	The item responsible for launching dialog or the symbol none. You can specify this item if you invoke uil-start-or-refocus-dialog from a user menu choice.
<i>window</i>	The window on which to manage the dialog. You can specify this window if you invoke uil-start-or-refocus-dialog from a user menu choice.

Return Value	Description
<u><i>dialog</i></u>	The dialog, or the symbol dialog-not-found.
<u><i>button</i></u>	The button that was activated on the dialog, or the symbol none.

Description

uil-start-or-refocus-dialog first checks to see if a dialog whose ID matches *dialog-id* is currently displayed on the specified *window*. If so, the dialog is updated to display information associated with the new *initiating-item*.

If uil-start-or-refocus-dialog does not find a dialog with the specified ID on the window, it searches the dialog bin for a copy dialog to display. If the dialog bin does not contain a copy dialog, uil-start-or-refocus-dialog clones the master dialog and displays the resulting copy dialog.

uil-start-dialog-processing

Launches or updates a dialog. Runs different actions on the dialog, depending on whether it launches or updates the dialog.

Synopsis

uil-start-dialog-processing

(*dialog-id*: text, *initiating-item*: item-or-value, *window*: class g2-window,
start-actions: item-or-value, *refocus*: truth-value,
conclude-values: truth-value, *refocus-actions*: item-or-value,
prompt-for-conclude: truth-value)
 -> {*dialog*: item-or-value | dialog-not-found: symbol},
 {*button*: item-or-value | none: symbol}

Argument	Description
<i>dialog-id</i>	The ID of the dialog to display or refocus.
<i>initiating-item</i>	The item to bind as initiating-object or the symbol <code>none</code> . You can specify this item if you invoke <code>uil-start-dialog-processing</code> from a user menu choice.
<i>window</i>	The window on which to manage the dialog, or to check for if already managed. You can specify this window if you invoke <code>uil-start-dialog-processing</code> from a user menu choice.
<i>start-actions</i>	The action(s) to run if dialog is started. Specify a symbol naming an existing action, the name of a <code>uil-action-description-array</code> , or the symbol <code>none</code> .
<i>refocus</i>	If <code>true</code> , attempt to refocus dialog. If <code>false</code> , display a new copy.
<i>conclude-values</i>	If <code>true</code> , run the <code>conclude</code> method before refocusing the dialog.

Argument	Description
<i>refocus-actions</i>	Specifies action(s) to run if dialog is refocused. You can specify a symbol naming an existing action, the name of a <code>uil-action-description-array</code> , or the symbol <code>none</code> .
<i>prompt-for-conclude</i>	If <code>true</code> , prompt user to run <code>conclude</code> method before refocusing dialog, or not to run <code>conclude</code> .

Return Value	Description
<i><u>dialog</u></i>	The dialog, or the symbol <code>dialog-not-found</code> .
<i><u>button</u></i>	The button that was activated on the dialog, or the symbol <code>none</code> .

Description

The `uil-start-dialog-processing` procedure displays or refocuses the dialog whose ID matches the specified ID on the given window.

Launching a Dialog from an Action Button

A G2 action button can launch a dialog by invoking `uil-control-dialog-callback` directly, or by invoking it through calls to `uil-start-dialog`, `uil-start-or-refocus-dialog`, or `uil-start-dialog-processing`.

The action button must use the `start` action to invoke these procedures.

Note In order to launch a dialog from an action button, you must first edit each UIL control on the master dialog to specify the source and target object and attribute of that UIL control.

Example: Launching the "current-rates" dialog from an action button

Suppose that an action button named `show-dialog-action-button` starts a dialog whose ID is `current-rates`. The action button contains the following `start` action:

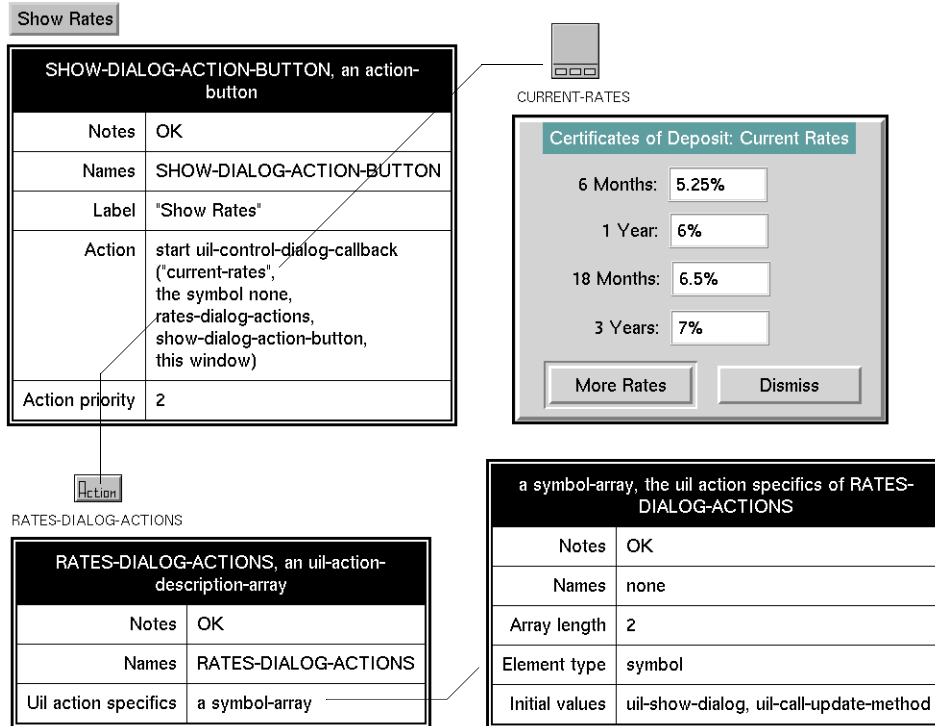
```
start uil-control-dialog-callback ("current-rates", the symbol none,  
    rates-dialog-actions, show-dialog-action-button, this window)
```

where:

- `current-rates` is the ID of the dialog that is launched.
- `rates-dialog-actions` is the action description array that specifies the actions run on the `current-rates` dialog. `rates-dialog-actions` contains the following actions:
 - `uil-show-managed-dialog` (show the dialog)
 - `uil-call-update-method` (update all the objects on the `current-rates` dialog that can receive values from source objects)
- `show-dialog-action-button` is the action button that launches the dialog.

Note You can omit the two `item-or-value` return arguments of `uil-control-dialog-callback` when you invoke this procedure using the `start` action.

The following figure illustrates the show-dialog-action-button action button labeled Show Rates, the Current Rates dialog, and the rates-dialog-actions action description array:



Launching a Dialog from a User-Defined Procedure

A user-defined G2 procedure can launch a dialog by invoking the uil-control-dialog-callback procedure directly, or by invoking it indirectly through calls to uil-start-dialog, uil-start-or-refocus-dialog, or uil-start-dialog-processing.

A user-defined procedure can use the call or start action to invoke these procedures.

Example: Launching the Current-Rates Dialog through a User-Defined Procedure

Suppose that you want to write a procedure that shows and updates the dialog whose ID is current-rates. You also want to invoke this procedure from an action button.

To do this, you follow these steps:

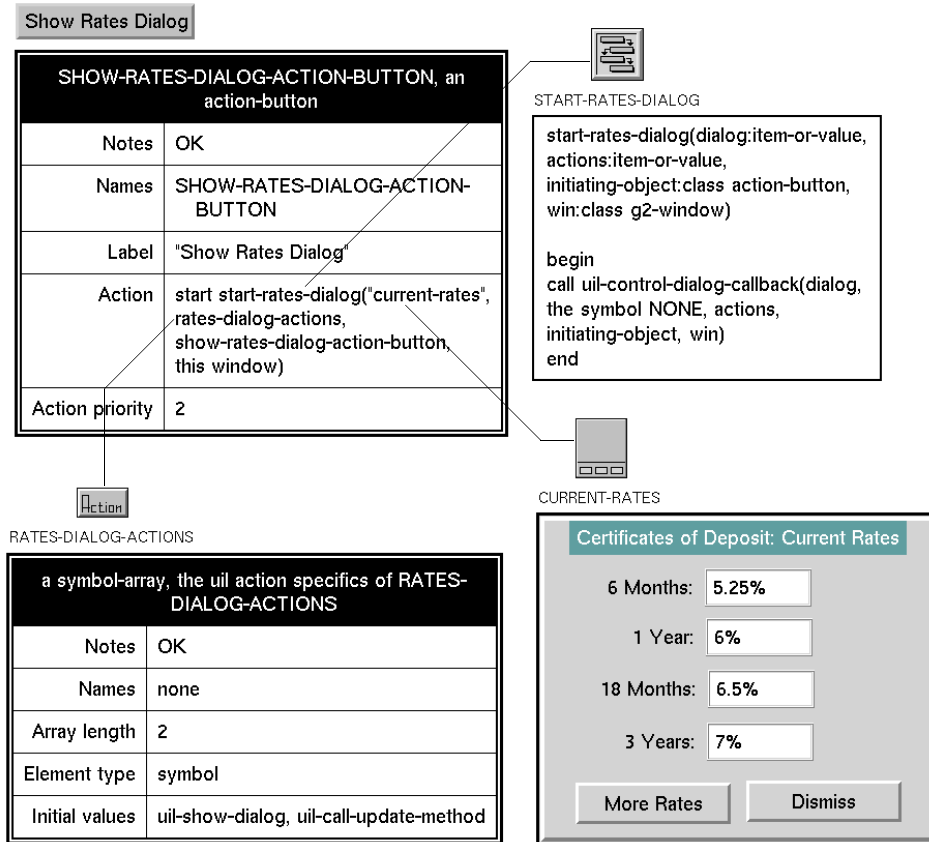
- 1 Write a procedure that calls `uil-control-dialog-callback` to show and update the `current-rates` dialog. In this example, the procedure is named `start-rates-dialog`.
- 2 Create an action description array that includes the actions that you want to run on the `current-rates` dialog:
 - `uil-show-managed-dialog` (show the dialog)
 - `uil-call-update-method` (update all the objects on the dialog that can receive values from source objects)

In this example, the action description array is named `rates-dialog-actions`.

- 3 Create an action button that launches the procedure `start-rates-dialog`.

The action button should use a `start` action that passes to the procedure the ID of the dialog that you want to launch, and the name of the action description array that you want to run on the dialog.

The following figure illustrates how the action button named `show-rates-dialog-action-button` launches the procedure `start-rates-dialog` to show and update the dialog:



For more information about `uil-control-dialog-callback`, see the *G2 GUIDE/UII Procedures Reference Manual*.

Processing a Dialog Before Returning it to the Dialog Bin

For some purposes, a procedure that launches a dialog may need to process values in a dialog after a user closes it, but before the dialog is returned to the dialog bin. In this case, you must use the `call` action to call the procedure that launches the dialog and use the action `uil-return-dialog` to return the dialog to the procedure that launched it. For example, an OK or Dismiss button that includes `uil-return-dialog` among its actions can return the dialog to the launching procedure. When a dialog is returned by `uil-return-dialog`, its contents can still be accessed by the launching procedure, even though users can no longer use the dialog.

When the procedure finishes processing a dialog that has been returned to it by `uil-return-dialog`, the procedure should call the `uil-release-dialog` action to return the dialog to its dialog bin and make it available for reuse by the application. The action `uil-release-dialog` returns the dialog to the dialog bin, rather than to the procedure that launched the dialog.

Launching a Dialog from a User Menu Choice

A G2 user menu choice can launch a dialog by invoking the `uil-control-dialog-callback` procedure directly, by invoking it through calls to `uil-start-dialog`, `uil-start-or-refocus-dialog`, or `uil-start-dialog-processing`.

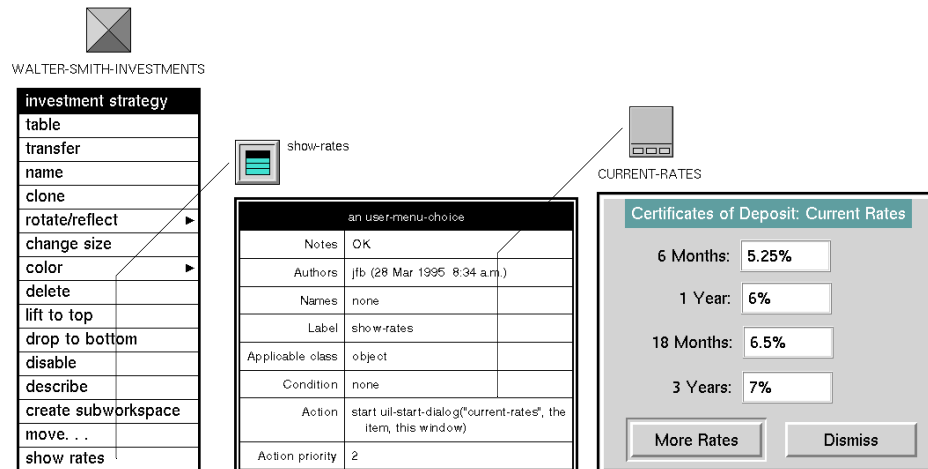
The user-menu choice must use the `start` action to invoke these procedures.

Example: Launching the Current-Rates Dialog from a User-Menu Choice using `uil-start-dialog`

The following statement opens the dialog whose ID is `current-rates` on the G2 window where the user menu choice is used:

```
start uil-start-dialog ("current-rates", the item, this window)
```

The following figure illustrates the `show rates` menu choice on an object named `walter-smith-investments`:



The `show rates` menu choice appears on the table for `walter-smith-investments` because this menu choice applies to the class object, and `walter-smith-investments` is an instance of a class whose direct superior class is `object`.

Example: Launching the Current-Rates Dialog from a User-Menu Choice using uil-start-or-refocus-dialog

The following statement launches or refocuses (updates) the dialog whose ID is `current-rates` on the G2 window where the user menu choice is used:

```
start uil-start-or-refocus ("current-rates", the item, this window)
```

Example: Launching the Current-Rates Dialog from a User-Menu Choice using uil-start-dialog-processing

The following statement opens the dialog whose ID is `current-rates` on the G2 window where the user menu choice is used:

```
start uil-start-dialog-processing ("current-rates", the item, this window,  
    rates-dialog-actions, true, true, uil-call-update-method, false)
```

where:

- `current-rates` is the ID of the dialog.
- `the item` is this user-menu choice.
- `this window` launches or updates the dialog on the window that displays the object with this user menu choice.
- `rates-dialog-actions` is an action description array that is run if the dialog is launched. This action array contains the actions:
 - `uil-call-update-method`
 - `uil-simulate-play-mode`
 - `uil-show-managed-dialog`
- `true` refocuses the dialog if it is already displayed.
- `true` runs the `conclude` method on this dialog before it is refocused.
- `uil-call-update-method` is a UIL procedure that is called if the dialog is already displayed and needs only to be updated. If you need to run more than one action on the dialog when it is refocused, you can specify an action array containing all the required actions, rather than a single action.
- `false` means do not prompt the user about running the `conclude` method.

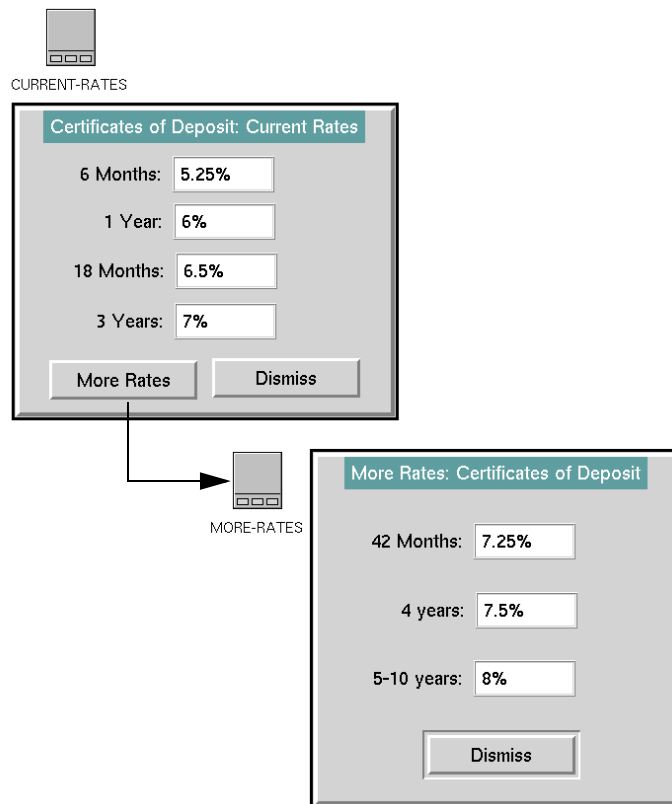
Launching a Dialog from a Push Button

To make it possible to open a dialog by clicking a push button, you edit the push button to specify that:

- The actions associated with the push button are directed to the dialog that you want to open.
- The actions associated with the push button include opening and displaying this target dialog

For information about how to do this, see [System-Defined Actions for Dialog Processing](#).

For example, a dialog with title Certificates of Deposit: Current Rates contains a push button with the label More Rates. Users can click the More Rates button to open a second dialog with the title More Rates: Certificates of Deposit. The following figure illustrates the relationship between these two dialogs:



To make it possible to open the second dialog by clicking the More Rates button, you must edit the More Rates button to specify that its target object is the second dialog. In addition, the actions on the More Rates button must include showing the dialog, as well other actions that you want to run on the dialog, such as updating it.

Creating Push Buttons to Launch Dialogs

The Edit Dialog dialog enables you to create push buttons to launch existing dialogs.

To create a push button to launch an existing dialog, follow these steps:

- 1 Select **edit dialog** from the menu of the dialog that you want to launch to open the Edit Dialog dialog.
- 2 In the Edit Dialog dialog, click the button **Create Pushbutton to Launch Dialog** to create the new push button.

The target object of the new push button is the dialog that you are editing. When a user clicks on the push button, it runs the following actions on the target dialog:

```
uil-call-update-method  
uil-simulate-play-mode  
uil-show-managed-dialog
```

- 3 Transfer the push button to the workspace or dialog subworkspace where you want users to be able to use it.

The GUIDE Dialog Generator enables you to create push buttons to launch the master dialogs that you generate. For information about the GUIDE Dialog Generator, see [Generating Master Dialogs](#).

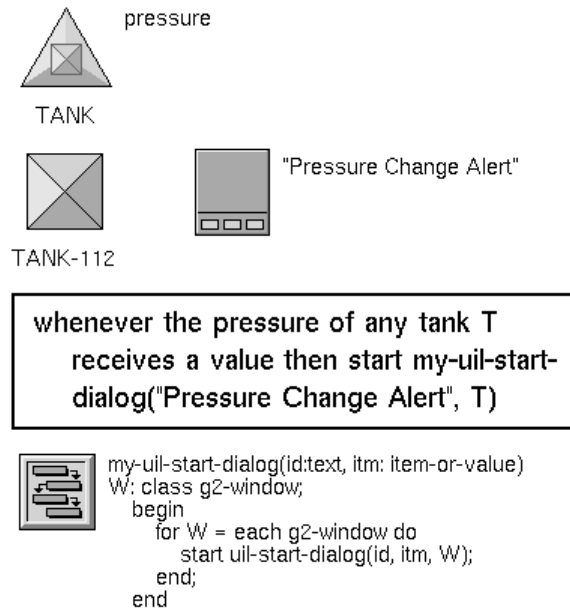
Specifying Source and Target Objects for UIL Controls on a Dialog Launched from a Push Button

If you intend to launch a dialog from a push button, you must edit each UIL control on the dialog to specify its source and target objects and attributes. For information about how to do this, see [Specifying Source and Target Objects](#).

Launching a Dialog from a Rule

A rule can launch a dialog by invoking a user-defined procedure that identifies the G2 window on which the dialog is to be launched. Procedures that have an argument for G2 window cannot be invoked directly from a rule, because the rule itself cannot pass a value to the G2 window argument.

The following figure illustrates a rule that launches a dialog, Pressure Change Alert, by invoking a user-defined procedure, my-uil-start-dialog:



In the example shown in the figure above, **tank** is a user-defined class with a class-specific attribute named **pressure**.

Whenever the **power** attribute of an instance of **tank**, such as **tank-112**, receives a value, the rule invokes the user defined procedure **my-uil-start-dialog**.

my-uil-start-dialog invokes **uil-start-dialog** to launch the dialog Pressure Change Alert on any G2 window where the dialog is found.

System-Defined Dialogs

Describes the predefined message, confirmation, query, and notification dialogs and delay notification icon.

Introduction **151**

Message, Query, Confirmation, and Notification Dialogs **151**

Delay Notification **158**



Introduction

GUIDE provides templates for the system-defined message, confirmation, query, and notification dialogs available in most windowing systems. It also supports a delay notification icon that you can post to inform the user that a time-consuming task is being processed by the application.

Message, Query, Confirmation, and Notification Dialogs

You can display these dialogs with customized messages, using the `uil-post-generic-dialog` procedure.

`uil-post-generic-dialog` returns values that indicate the user's response to the dialog, such as clicking on a button or entering a value. The values returned depend on the type of the dialog posted.

The following section describes `uil-post-generic-dialog`.

Using uil-post-generic-dialog to Post Dialogs

The system-defined procedure `uil-post-generic-dialog` posts a confirmation, message, query, or notification dialog.

Synopsis

`uil-post-generic-dialog`

(*txt*: text, *type*: symbol, *size*: symbol, *window*: class g2-window,
icon: symbol)

-> *button-label*: text, *return-sstring*: text, *dialog*: item-or-value

Argument	Description
<i>txt</i>	The text to post in the dialog.
<i>type</i>	The type of generic dialog to use. Specify confirmation, message, query, or notification.
<i>size</i>	The size of the text. Specify small, medium, or large.
<i>window</i>	The G2 window on which to post the dialog. Note: This procedure creates an inoperable dialog if <i>window</i> specifies an invalid G2 window. An invalid G2 window is one whose <code>g2-connection-status</code> is <i>not</i> equal to the symbol connected.
<i>icon</i>	The class-name of an icon to display with text. UIL provides the following icons: <code>uil-information-icon</code> , <code>uil-question-icon</code> , <code>uil-warning-icon</code> .
Return Values	Description
<u><i>button-label</i></u>	The label of the button selected: Yes, No, OK, Cancel. Query dialogs return Yes if the OK button is selected or No if the Cancel button is selected.
<u><i>return-string</i></u>	The value returned by a query dialog, or "" (empty string).
<u><i>dialog</i></u>	The dialog, if type specified is notification. Otherwise, the symbol <code>dialog-not-available</code> .

Description

This procedure posts one of GUIDE's system-defined dialogs: Confirmation, Message, Query, Notification.

If you use this procedure to post a notification dialog, you must also call `uil-remove-notification-dialog` to hide and release the dialog.

For example, the following user-defined procedure, `post-generic-dialog`, calls `uil-post-generic-dialog`:

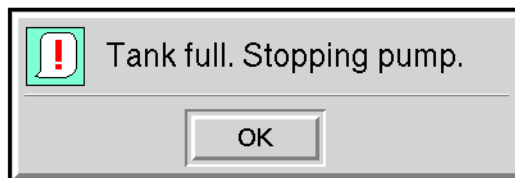
```
post-generic-dialog(TEXT: text, TYPE: symbol,  
SIZE: symbol, WIN: class g2-window, ICON: symbol)  
BUTTON-LABEL, RETURN-VALUE: text;  
DIALOG: item-or-value;  
  
begin  
BUTTON-LABEL, RETURN-VALUE, DIALOG =  
call uil-post-generic-dialog(TEXT, TYPE, SIZE, WIN, ICON);  
inform the operator that "[BUTTON-LABEL], [RETURN-VALUE], [DIALOG]";  
end
```

Message Dialog

A message dialog displays a customized message and contains an OK button, which a user can click to dismiss the dialog. The following figure illustrates a sample message dialog:

Alert Operator

POST-WARNING, an action-button	
Notes	OK
Names	POST-WARNING
Label	"Alert Operator"
Action	start uil-post-generic-dialog ("Tank full. Stopping pump.", the symbol message, the symbol large, this window, the symbol uil-warning-icon)
Action priority	2



Query Dialog


A query dialog contains a user-defined message, an edit box in which the user can respond to the message, and OK and Cancel buttons.

The following figure illustrates a query dialog that displays a message, and contains an edit field into which a user can enter a response to the query:

Get tank name

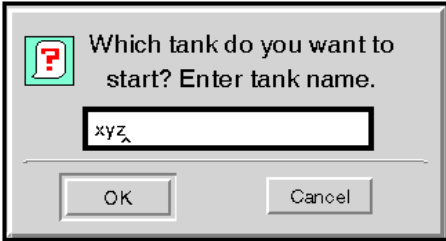
```
start post-query-dialog
("Which tank do you want to start? Enter tank
name.",
 the symbol query,
 the symbol large,
 this window,
 the symbol uil-question-icon)
```

POST-QUERY-DIALOG



```
post-query-dialog
(Text: text,
 Dialog-type: symbol,
 Dialog-size: symbol,
 Window: class g2-window,
 Icon: symbol)
Button-label, Return-value: text;
Dialog: item-or-value;

begin
Button-label, Return-value, Dialog =
call uil-post-generic-dialog(Text, Dialog-type,
 Dialog-size, Window, Icon);
{statements that start tank specified by user}
inform the operator that "Tank [Return-value]
is being started.";
end
```



MESSAGE-BOARD

#69 2:58:58 p.m. Tank xyz is being started.

In this example, the action button labeled **Get tank name** starts a user-defined procedure named `post-query-dialog`. The text of the message that is to be displayed in the query dialog, *Which tank do you want to start? Enter tank name.*, is passed by the `start` action to the user-defined procedure.

The user-defined procedure, rather than the `start` action, is used to call `uil-post-generic-dialog`, because a `start` action cannot capture the return values of procedures. Thus, the `start` action could not capture the value entered by the user in the query dialog.

The user-defined procedure captures user input to the query dialog in the argument `Return-value`, which can be referenced by statements that start the tank (not shown). The user-defined procedure also informs the operator that the tank whose name was entered by the user in the query dialog is being started.

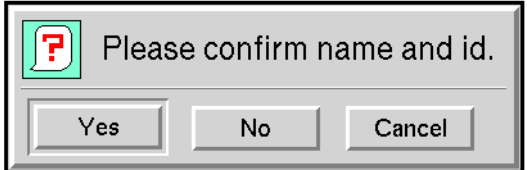
Confirmation Dialog

A confirmation dialog prompts users to confirm that they want to take an action, such as deleting an object. The dialog contains Yes, No, and Cancel buttons for responding to the prompt.

For example, the following confirmation dialog appears when a user selects a small radio button and chooses delete from the button's menu:

Please confirm

START-CONFIRMATION, an action-button	
Notes	OK
Names	START-CONFIRMATION
Label	"Please confirm"
Action	start uil-post-generic-dialog ("Please confirm name and id.", the symbol confirmation, the symbol large, this window, the symbol uil-question-icon)
Action priority	2



Click Yes to delete the object. Click No if you decide that you do not want to delete the object.

Notification Dialog

The Notification Dialog displays a string of text that you specify. You can use this dialog to inform users about a delay in processing or any other condition. For example:



To open a Notification Dialog, call the procedure `uil-post-notification-dialog`. To close the Notification Dialog, call `uil-remove-notification-dialog`. You can change the message in an open Notification Dialog, using the procedure `uil-update-notification-text`.

For example, the following user-defined procedure, `post-notification-dialog`, calls `uil-post-notification-dialog`, starts a time-consuming process, and calls `uil-remove-notification-dialog` when the processing is done:

```

post-notification-dialog
(TEXT: text,
SIZE: symbol,
WIN: class g2-window,
ICON: symbol)

DIALOG: item-or-value;

begin
DIALOG = call uil-post-notification-dialog(TEXT, SIZE, WIN, ICON);

{The following wait action is used as an example of a time-consuming process.}
wait for 60 seconds;

call uil-remove-notification-dialog(DIALOG, WIN);
end
    
```

The following figure illustrates an action button that starts the user-defined procedure `post-notification-dialog`:

Start Lengthy Process

START-NOTIFICATION, an action-button	
Notes	OK
Names	START-NOTIFICATION
Label	"Start Lengthy Process"
Action	start post-notification-dialog ("Your request is being processed. Please wait.", the symbol large, this window, the symbol uil-information-icon)
Action priority	2

Delay Notification

The Delay Notification displays a clock face and a text string. You can use this icon to tell users that a delay in processing is happening.

To open and close the Delay Notification, your application can call the procedures `uil-post-delay-notification` and `uil-remove-delay-notification-if-any`.

The following figure illustrates a Delay Notification and an action button, labeled `Post Delay`, that starts the Delay Notification. The action button starts a user-defined procedure, `post-delay-notification`, that displays the Delay Notification for 15 seconds, with the message `Waiting`.



start `post-delay-notification` (this window, "Waiting")



```
POST-DELAY-NOTIFICATION
post-delay-notification (W: class g2-window,
MSG: text)
  S: class uil-wait-semaphore;
  PI: class procedure-invocation;
begin
  PI = this procedure-invocation;
  S = call uil-post-delay-notification (W, PI,
MSG);
  wait for 15 seconds;
end
```

Note The `class-of-procedure-invocation` attribute of the user-defined procedure must be set to `procedure-invocation`. This setting causes the Delay Notification to be removed automatically when the procedure that posts the delay notification finishes executing.

For complete descriptions of `uil-post-delay-notification` and `uil-remove-delay-notification-if-any`, see the *G2 GUIDE/UIL Procedures Reference Manual*.

Editing User Interface Components

Chapter 9: Push Buttons

Describes how you can create and edit push buttons to run specific sets of actions on dialogs, such as opening and closing them, or updating or concluding their values.

Chapter 10: Radio Buttons

Describes how to create and edit groups of radio buttons to provide users with sets of mutually exclusive choices.

Chapter 11: Check Buttons

Describes how to create and edit groups of check buttons, in which users can select any number of choices.

Chapter 12: Toggle Buttons

Describes how to create and edit toggle buttons, which represent two mutually exclusive choices.

Chapter 13: Edit Boxes, Combo Boxes, and Spin Controls

Describes how to create and customize edit boxes, combo boxes, and spin controls.

Chapter 14: Scroll Areas and Message Objects

Describes how to create and edit scroll areas and message objects.

Chapter 15: Sliders

Describes how to use and edit Sliders.

Chapter 16: Text Objects

Describes how to create and edit text objects, which display read-only text.

Chapter 17: Title Bars, Borders, and Separators

Describes how to use title bars, border, and separators to provide your user interface with visual definition.

Chapter 18: Navigation Buttons and Other Tools

Describes how to add navigation buttons, help buttons, print workspaces, and the GUIDE garbage pail to workspaces.

Push Buttons

Describes how you can create and edit push buttons to run specific sets of actions on dialogs, such as opening and closing them, or updating or concluding their values.

Introduction **161**

Editing Pushbuttons **169**

Summary of Push Button Menu Choices **181**



Introduction

GUIDE supports two kinds of push buttons: icon push buttons and text push buttons:



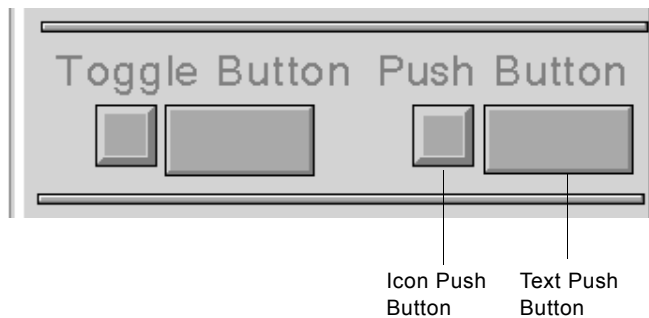
Icon push buttons and text push buttons differ from each other only in appearance. You can use either kind of push button to:

- Perform operations on dialogs by running procedures called **actions**. Actions can perform operations on a dialog such as opening or closing the dialog, or updating or concluding its values. The following section describes how to use a push button to run actions on a dialog.
- Run a user-defined callback procedure to perform specialized processing. GUIDE continues to support the features of GUIDE 3.0 that enable developers

to use system-defined callbacks and/or write their own procedures to perform desired behaviors and functionality through push buttons. For information about how to create and use callbacks, see [Methods, Actions, and Callbacks](#).

Adding Push Buttons to a Master Dialog

The GUIDE palette provides icons for two distinct kinds of push buttons, icon push buttons and text push buttons:



You can add push buttons to a master dialog by clicking either of these icons and dragging them to the subworkspace of the dialog. You can also add push buttons by selecting either of these choices from the GUIDE menu bar:

- Item > GUIDE Objects > Buttons > uil-icon-pushbutton
- Item > GUIDE Objects > Buttons > uil-text-pushbutton

The icon for the class of push button that you select becomes attached to your cursor. You can drop it on the subworkspace of the master dialog that you are editing.

Using Push Buttons to Perform Operations on Dialogs

You can create push buttons that enable users to perform operations on dialogs such as updating and concluding the values in the dialog, opening another dialog, or closing the dialog.

Push buttons that perform operations on dialogs should use the default callback for push buttons, `uil-control-dialog-callback`. When a user clicks on the push button, this callback runs the set of actions associated with the push button.

Actions perform operations such as displaying or hiding the dialog, or running the update or conclude methods of the UIL objects on the dialog. The actions can be run on the dialog that contains the push button, or on a different dialog.

You can associate a wide variety of actions with push buttons. GUIDE provides system-defined actions for displaying, updating, and dismissing the dialog, as well as for other common operations. For more information about actions, see [Summary of Push Button Menu Choices](#).

Each dialog that is open on a window has an **event queue**. When a user clicks on the push button or when a developer running in Build Mode or Administrator Mode chooses **select** from the push button's menu, the push button's actions are inserted into the dialog's event queue and are processed.

The dialog on which the actions are run is specified by the push button's `uil-event-target-object` attribute. The actions associated with a push button are stored in a symbol array, which is referenced by an attribute of the push button named `uil-action-description`.

To specify which actions are run when a user pushes a push button, use the Edit Dialog Actions dialog, which you can access through the Edit Pushbutton dialog. The Edit Dialog Actions dialog contains a set of radio buttons that are preconfigured with sets of actions for OK, Apply, Cancel, Update, and Conclude buttons. The Edit Dialog Actions dialog also specifies the dialog on which the actions are run.

You can create a customized set of actions and associate it with a push button, using the Customize Dialog Actions dialog. The customized set of actions can include both system-defined actions and user-defined actions. You can access the Customize Dialog Actions dialog through the Edit Dialog Actions dialog.

You can also edit the contents of the symbol array referenced by the `uil-action-description` attribute directly. To do this, click the attribute value (a symbol-array) of the `uil-action-description` attribute and select **subtable**. The subtable for the symbol array appears. In this subtable, you edit the list of actions stored in the `initial-values` attribute. The following figure illustrates the subtable of a symbol array:

a symbol-array, the uil action description of some uil-medium-text-pushbutton	
Notes	OK
Names	none
Array length	4
Element type	symbol
Initial values	uil-call-conclude-method, uil-unsimulate-play-mode, uil-hide-dialog, uil-release-dialog
Array is permanent	no

Note If you edit the actions directly in this sub-table without using the GUIDE dialogs, you must set the array length to 0, and then to the actual number of actions specified. This technique forces the array to reinitialize itself.

By default, the OK, Apply, and Cancel buttons on GUIDE dialogs are preconfigured to run the standard OK, Apply, and Cancel actions.

System-Defined Actions for Dialog Processing

The following list describes the system-defined actions that perform operations on dialogs. These actions can be associated with a push button or included in an Action Description Array that is run from a procedure.

System-Defined Actions for Dialog Processing

Action	Description
uil-call-conclude-method	<p>Runs the conclude methods of all UIL objects on the dialog that have state.</p> <p>The conclude method looks at the <code>uil-event-target-object</code> and <code>uil-event-target-attribute</code> to identify the location where the state of the object is to be directed upon execution of the conclude method.</p> <p>Conclude methods can handle complex data structures such as arrays, lists, variables, and parameters. If the <code>uil-event-target-object</code> and <code>uil-event-target-attribute</code> of a scroll-area point to a symbol array, for example, all of the message objects of the scroll-area conclude their values into the target array.</p>
uil-call-conclude-method-for-parent	<p>Runs the conclude method for the parent dialog, if one exists. The conclude method is run iteratively on every UIL object that has a value that can be concluded.</p>
uil-call-conclude-method-for-children	<p>Runs the conclude method for all child dialogs, if these methods exist. On each child dialog, the conclude method is run on every UIL object that has a value that can be concluded.</p>

System-Defined Actions for Dialog Processing

Action	Description
uil-call-update-method	An update method can be run on any UIL object. The update method looks at the uil-event-source-object and uil-event-source-attribute to identify the source of the UIL object's state value. The update method is run on every UIL object on the dialog that has a value that can be updated.
uil-call-update-method-for-parent	Runs the update method on the parent dialog. On the parent dialog, the update method is run iteratively on every UIL object that has a value that can be updated.
uil-call-update-method-for-children	Runs the update method on all child dialogs. On each dialog, the update method is run iteratively on every UIL object that has a value that can be updated.
uil-call-configuration-method	Runs the configuration method specified in the attribute uil-configuration-method of the object. The configuration method applies the configuration identified in the configuration attribute of the uil-grobj or uil-grmes.
uil-simulate-play-mode	Places the dialog into a temporary state of proprietary behavior, simulating play mode. When the dialog is in a proprietary state, users cannot access its menu.
uil-show-managed-dialog	Displays the indicated dialog on the indicated window. If the dialog contains editable fields, the field editor is invoked and takes focus on the first field whose Allow field edit option is set to true.
uil-hide-dialog	Hides the indicated dialog on the indicated window. Any edit session in progress is canceled.
uil-unsimulate-play-mode	Restores the indicated dialog to a normal (non-proprietary) state.

System-Defined Actions for Dialog Processing

Action	Description
uil-release-temporary-storage-object	Releases the temporary storage object from its relation with the indicated dialog if one exists.
uil-delete-temporary-storage-object	Deletes the temporary storage object related to the indicated dialog if one exists.
uil-return-dialog	<p>Returns the dialog to the procedure that launched it, without returning the dialog to the dialog bin.</p> <p>The dialog is not available for reuse by your application until it is returned to the dialog bin. The application is responsible for calling uil-release-dialog to return the dialog to the dialog bin.</p>
uil-release-dialog	Releases the indicated dialog and returns the dialog instance to the pool of available dialogs. This dialog can now be reused by another client.

System-Defined Actions for Dialog Processing

Action	Description
uil-delete-dialog	Deletes the instance (copy) of the indicated dialog. It will not, therefore, be placed in the dialog bin or available for reuse.
uil-call-validate-method	<p>This method only applies to uil-edit-box instances. The validation method compares the contents of the edit box against the valid values as defined by the format specified for the edit box.</p> <p>If this method is run on a dialog, the fields in the dialog are validated in order, beginning with the top-left field and continuing to left to right and top to bottom.</p> <p>Should there be any discrepancies between the format and the edit-box value, an error is generated. See uil-validations discussed later in this guide.</p> <p>For information about how uil-call-validate-method validates edit boxes, see How GUIDE Validates Edit Boxes.</p>

Creating Actions

You can create dialog actions using the GUIDE Method Help dialog. This dialog creates an empty action with the set of arguments required for all actions. In the body of the action, you specify the operations that the action is to perform. You can invoke any UIL procedures in the body of the action.

For information about how to create actions, see [Creating Callbacks, Methods, Procedures, Functions, and Actions Using the GUIDE Method Help Dialog](#).

Setting a Target Object for a Push Button

The target object of a push button is the dialog on which the actions associated with the push button are run when a user pushes the push button.

Unlike other kinds of buttons, push buttons do not have state values that can be concluded to or updated from other objects. Push buttons do not have source

objects. For information about how to specify the target object of a push button, see [System-Defined Actions for Dialog Processing](#).

Specifying Labels for Push Buttons

By default, the labels of Motif-style push buttons created in GUIDE 5.0 or higher are contained in the icons of the buttons. Because the label is part of the icon, a Motif-style button and its label move and remain together whenever you drag either of them.

However, Windows-style push buttons use separate label objects for labels, as in versions of GUIDE/UII prior to 5.0.

For both Motif and Windows style push buttons, you specify label text in the Edit Pushbutton dialog. In this dialog, you also specify whether you want a push button to resize itself to fit its label. For information about how to use the Edit Pushbutton dialog, see [Edit Pushbutton Dialog](#).

Upgrading Push Buttons Created with Previous Versions of GUIDE/UII

GUIDE/UII supports push buttons with separate label objects created in versions of GUIDE/UII prior to 5.0. It does not automatically convert them to place their labels in their icons. For information about how to convert existing Motif-style push buttons to the GUIDE/UII 5.0 styles, see Chapter 26, “Upgrading Guide Applications” in the *G2 GUIDE User’s Guide G2 Utilities Version 5.0* manual.

Using Label Objects for Push Button Labels

Dialogs that contain Motif-style push buttons created with GUIDE/UII 5.0 or higher launch more quickly than dialogs that contain push buttons with separate label objects. However, if you want to use separate label objects for button text, as was used in versions of GUIDE/UII prior to 5.0, set the `uil-use-icon-text-for-buttons` parameter to `false`. You can set this parameter using the following menu choice from the GUIDE menu bar:

Tools > GUIDE 50r0 Migration Tools > Create 50r0 Buttons

When Create 50r0 Buttons is not selected, GUIDE/UII creates a separate object to contain the label for each button that you create.

You can also set the `uil-use-icon-text-for-buttons` parameter using a `conclude` statement such as the following in an action button:

```
conclude that uil-use-icon-text-for-buttons is false
```

Caution Application code that accesses text in separate label objects will not work if you use the GUIDE/UIL 5.0-style buttons that include button and text within a single object.

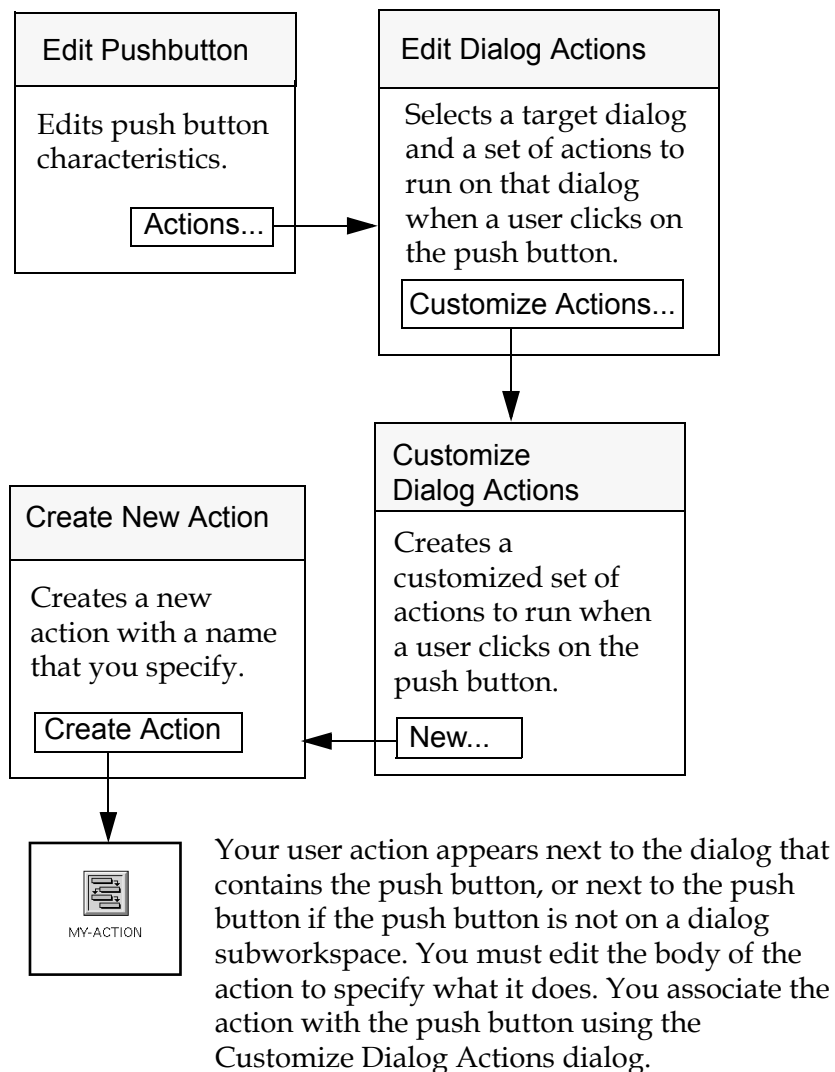
Do not attempt to extract text from buttons using relations. Instead, you can use the UIL procedure `uil-get-label-text` to return the `uil-text` object that provides the text of another object's label.

Editing Pushbuttons

You can edit the appearance and behavior of push buttons using the Edit Pushbutton dialog, Edit Dialog Actions dialog, Customize Dialog Actions dialog, and Create New Actions dialog.

The following figure illustrates the dialogs that you use to edit push buttons and their actions:

Dialogs for Editing Push Buttons and Their Actions



Edit Pushbutton Dialog

The Edit Pushbutton dialog enables you to edit basic characteristics of a push button, such as its name, label, ID, size, and location.

You can also specify the actions that are run when a user pushes a push button, using dialogs that you access from the Edit Pushbutton dialog.

To open the Edit Pushbutton dialog, click the push button and choose edit push button from the push button menu. The Edit Pushbutton dialog looks like this:

The following table describes the components of the Edit Pushbutton dialog:

Components of Edit Pushbutton Dialog

Component	Description
Class	(read-only) Indicates the class of push button that you are editing.
Name	(optional) Displays the current name of the push button that you are editing. You can edit this field to change the name of the push button. The contents, if any, must be a valid symbolic name.

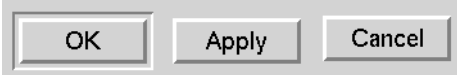
Components of Edit Pushbutton Dialog

Component	Description
Label	(optional) Displays the label of the push button that you are editing. You can edit this field to change the text of the label for the push button. This field does not require a value. Its contents, if any, must be a valid text entry. Quotation marks are not required. If you include quotation marks, they become part of the label.
Id	(optional) Displays the ID of the push button that you are editing. You can edit this field to change the id attribute of the push button. This field does not require a value.
Position	(required) The two edit box fields (X, Y) below the label Position display the current <code>item-x-position</code> and <code>item-y-position</code> of the push button that you are editing. Changes to the X and Y values move the push button to the new location. This is a required field and defaults to the push button's current position on the workspace. Values for X and Y must be integers.
Dimensions	The two text fields below the label Dimensions display the current <code>item-width</code> and <code>item-height</code> of the push button that you are editing. You cannot edit this field.
Size	(required) The radio buttons below the label Size indicate the current size (small, medium, or large) of the push button that you are editing. You can change the size of the push button by selecting the radio button whose label indicates the desired size. When you apply this change, GUIDE regenerates the push button in the new size without changing the button's other attributes.

Components of Edit Pushbutton Dialog

Component	Description
Style	(required) The radio buttons below the label Style (Motif™, Windows™) indicate the current window style of the push button that you are editing. You can change the window style by selecting the radio button whose label indicates the desired style. When you apply this change, GUIDE regenerates the push button in the new style without changing the button's other attributes.
Options	The toggle buttons below the label Options offer optional settings for the push button that you are editing.
Default button	<p>Makes the push button that you are editing the default button for this dialog. The default button is activated when the user hits the Return key.</p> <p>To make this push button the default button of its dialog, you must also configure the Return key behavior, as follows:</p> <ol style="list-style-type: none"> 1 Open the Edit Dialog dialog to edit the dialog that contains the push button you are editing. 2 Open the Editor Behaviors dialog by clicking the Editor Behaviors button in the Edit Dialog dialog. 3 In the Editor Behaviors dialog, select the End and exit field edit option under Return Key Behavior. <p>See Editor Behaviors Dialog.</p>
Cancel button	Makes the push button that you are editing the cancel button.
Update button	Identifies the pushbutton as an Apply type button.

Components of Edit Pushbutton Dialog

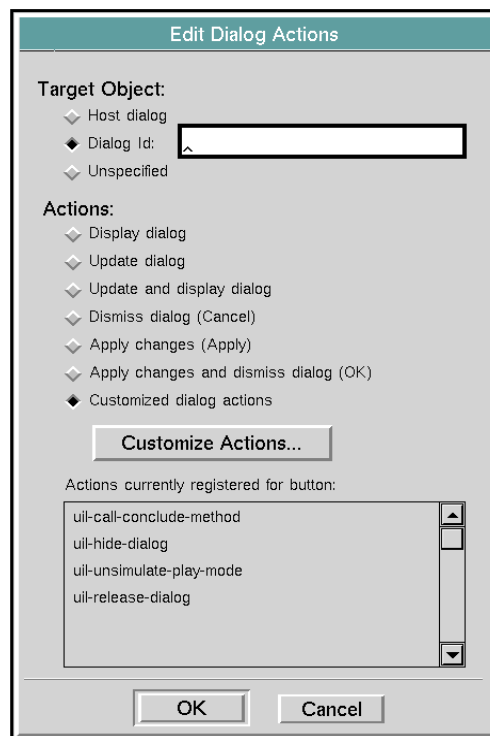
Component	Description
Default button configurations	<p>Applies the configuration for default buttons to the push button that you are editing. This configuration places a border around the default push button to distinguish it from the other push buttons in the dialog. The following figure illustrates an OK button with the default button configuration:</p> <div style="text-align: center;">  </div>
Release button after selection	<p>If selected, this option causes the button to become deselected and reusable as soon as it has initiated its actions.</p> <p>If this option is not selected, the button does not become deselected and reusable until the actions that it initiates are completed.</p>
Resize button to fit label	<p>If this option is selected (the default), GUIDE/UIIL automatically resizes the push button to fit the label text that you specify for the button. If this option is not selected, the button is not resized automatically.</p>
State	<p>(required) Select one of the two radio buttons (Enabled, Disabled) below the label State to enable or disable the push button.</p>
Callback	<p>Specify the callback associated with this push button. For information about callbacks, see Methods, Actions, and Callbacks.</p>
Actions	<p>Opens the Edit Dialog Actions dialog. See the following section for information about this dialog.</p>
Methods	<p>Opens the Edit Methods dialog, in which you can create customized methods for the push button.</p> <p>For information about how to create customized methods, see Methods, Actions, and Callbacks.</p>

Edit Dialog Actions Dialog

The Edit Dialog Actions dialog enables you to:

- Specify the dialog (target object) on which actions are run when a user clicks on this push button.
- Select a system-defined set of actions to run when a user clicks on the push button. The set of actions that you select are stored in the `uil-action-description` attribute of the push button.
- Create a customized set of actions to run when a user clicks on this push button.

To open the Edit Dialog Actions dialog, push the Actions push button in the Edit Pushbutton dialog. The Edit Dialog Actions dialog looks like this:



The following table describes the components of the Edit Dialog Actions dialog:

Components of Edit Dialog Actions Dialog

Component	Description
Target Object	<p>The radio buttons below the label Target Object indicate the dialog on which the actions are run when a user clicks the push button that you are editing.</p> <p>For information about target objects, see Edit Target Object & Attribute Dialog.</p>
Host dialog	<p>Indicates that the specified actions are run on the dialog where the push button is located.</p>
Dialog Id	<p>Indicates that the specified actions are to be run on the dialog whose id attribute matches the ID specified in the edit box to the right of the Dialog Id radio button. If the Dialog Id radio button is selected, the edit box to the right of the radio button requires a value. Id values are case sensitive.</p>
Unspecified	<p>No target object is specified for this push button.</p>
Actions	<p>Each radio button below the label Actions is associated with a different set of UIL actions. Selecting one of these buttons specifies that the corresponding set of UIL actions is run when a user clicks the push button. When you select a set of actions, that set replaces any existing set of actions associated with the push button. The current set of actions associated with the push button is displayed in the scroll area in the bottom half of the Edit Dialog Actions dialog.</p> <p>The actions are run in the order in which they are listed in the scroll area.</p>
Display dialog	<p>Selects the default actions for displaying a dialog. The default actions are: uil-simulate-play-mode, uil-show-managed-dialog.</p>

Components of Edit Dialog Actions Dialog

Component	Description
Update dialog	Selects the default action for updating a dialog. The default action is: <code>uil-call-update-method</code> .
Update and display dialog	Selects the default actions for updating and displaying a dialog. The default actions are: <code>uil-call-update-method</code> , <code>uil-simulate-play-mode</code> , <code>uil-show-managed-dialog</code> .
Dismiss dialog (Cancel)	Selects the default actions for dismissing a dialog. The default actions are: <code>uil-unsimulate-play-mode</code> , <code>uil-hide-dialog</code> , <code>uil-release-dialog</code> . These actions are appropriate for a Cancel button.
Apply changes (Apply)	Selects the default action associated with applying changes made on a dialog. The default action is: <code>uil-call-conclude-method</code> . These actions are appropriate for an Apply button.
Apply changes and dismiss dialog (OK)	Selects the default actions associated with applying changes made on a dialog and then dismissing the dialog. The default actions are: <code>uil-call-conclude-method</code> , <code>uil-unsimulate-play-mode</code> , <code>uil-hide-dialog</code> , <code>uil-release-dialog</code> . These actions are appropriate for an OK button.
Customized dialog actions	Select this button when you want to create a customized set of actions. The customized set can include user-defined actions.
Customize Actions	Clicking this push button opens the Customize Dialog Actions dialog. See the following section for information about the Customize Dialog Actions dialog.

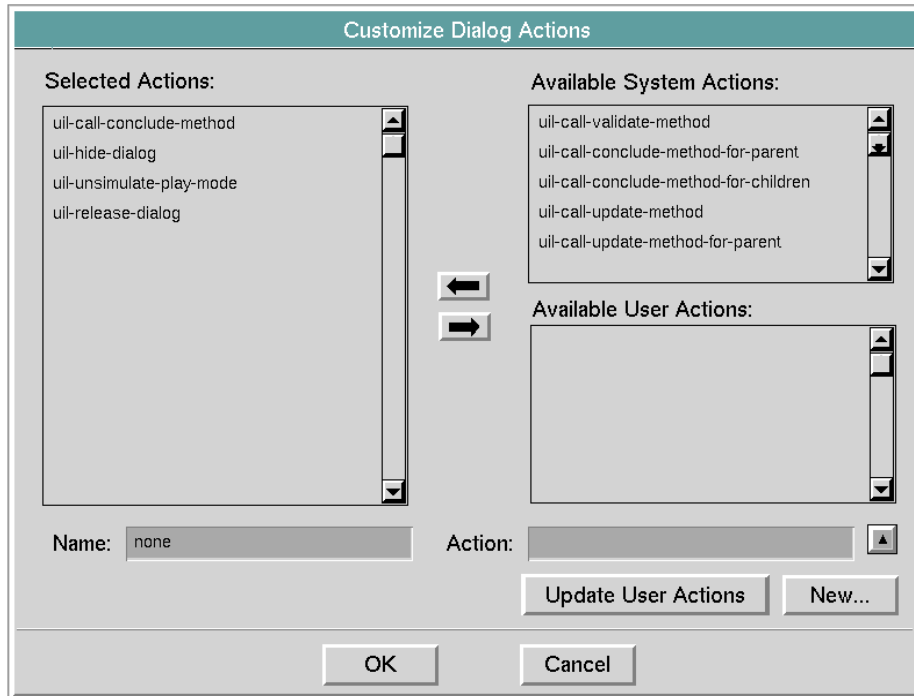
Customize Dialog Actions Dialog

The Customize Dialog Actions dialog enables you to specify a combination of UIL actions and user-defined actions that are run when a user clicks the push button.

To open the Customize Dialog Actions dialog, click the Customize Actions push button on the Edit Dialog Actions dialog. You can also open the Customize Dialog

Actions dialog to edit the actions in an Action Description Array by choosing **edit action description** from the menu of the Action Description Array.

The Customize Dialog Actions dialog looks like this:



The following table describes the components of the Customize Dialog Actions dialog:

Components of Customize Dialog Actions Dialog

Component	Description
Selected Actions	<p>Lists the actions that are run when a user clicks on the push button. You add actions to the Selected Actions list by moving them from the Available System Actions scroll area and/or the Available User Actions scroll area.</p> <p>You can reorder the actions in the Selected Actions scroll area by dragging actions up or down within the scroll area.</p> <p>To remove an action from the Selected Actions scroll area, select the action and then select the right arrow button. The selected action is transferred back to its original scroll area (Available System Actions or Available User Actions).</p>
Available System Actions	<p>Lists all the system-defined UIL actions. To transfer an action from this scroll area to the Selected Actions scroll area, select the action and click the left arrow button.</p> <p>You can select and transfer more than one action at a time.</p>
Available User Actions	<p>Lists all user-defined actions. To transfer an action from this scroll area to the Selected Actions scroll area, select the action and click the left arrow button.</p> <p>You can select and transfer more than one user-defined action at a time.</p>
Action	<p>You can add the name of a user-defined action to the Available User Actions list by entering its name in this field and clicking on the up arrow.</p>
Update User Actions	<p>Updates the Available User Actions scroll area to list all user-defined actions.</p>

Components of Customize Dialog Actions Dialog

Component	Description
New	<p>Opens the Create New Action dialog. See the following section for information about the Create New Action dialog.</p> <p>The name of the new action that you create appears in the field below the scroll area labeled Available User Actions. You can then transfer this user action to the Selected Actions scroll area.</p>
Name	<p>Displays the name of the Action Description Array that you are editing. This field contains a value only if you are editing a named Action Description Array that you created by selecting New Object > choose a class > uil-action-description-array from the KB Workspace menu.</p> <p>For information about Action Description Arrays, see Controlling Dialogs with Actions.</p> <p>You can also enter the name of an existing action.</p>
OK	<p>Copies the actions listed in the Selected Actions scroll area into the scroll area on the Edit Dialog Actions dialog and then dismisses the Customize Actions dialog.</p>
Cancel	<p>Dismisses the dialog.</p>

Create New Action Dialog

The Create New Action dialog enables you to specify a name for a user-defined action and generates the a stub for the action.

To open the Create New Action dialog, click the New button on the Customize Dialog Actions dialog. The Create New Action dialog looks like this:



The following table describes the components of the Create New Action dialog:

Components of Create New Action Dialog

Component	Description
Name	Enter a unique name for the user-defined action in this field. By default, this field is blank. You must enter a valid text name in this field.
Create Action	If the name that you enter in the Name field is unique, GUIDE creates a subworkspace with a new action on it. If the name is not unique, GUIDE displays a message informing you that you must enter a unique procedure name, and does not create an action.

The new action looks like this:



MY-NEW-ACTION

You should transfer the action to another workspace.

To edit the action, click it and choose **edit** from the procedure menu. In the editor window, you see a default definition with the required argument signature:

```
my-new-action (D:class uil-dialog, B: item-or-value, W: item-or-value,
              O: item-or-value, SL: class symbol-list)
begin
end
```

You must supply the code to specify what the action does when it is run.

Summary of Push Button Menu Choices

Menu Choice	Description
table	Shows the G2 table for the push button.
transfer	Places the push button on the mouse and transfers it to a different workspace.

Menu Choice	Description
name	Opens an editor allowing a name to be entered or changed.
rotate/reflect	Opens a menu of rotation and reflection options.
change size	Opens a G2 dialog providing change size options. Note: Use this menu choice only with push buttons whose label text is contained in separate label objects. It does not work with GUIDE 5.0 buttons whose icons contain the label text of the buttons.
color	Opens a series of G2 menus enabling you to change the colors of regions of the push button icon. GUIDE provides the Configuration Editor as the recommended way to change colors of icon regions. For information about this editor, see Using The GUIDE Configuration Editor .
lift to top	Places the button on top of an other object on a workspace.
drop to bottom	Drops the button behind another object on a workspace.
describe	Shows a description of the object and its relations.
describe configuration	Displays a description of the dialog's current configuration.
create subworkspace	Creates and shows a subworkspace for the button.
edit array	Opens the Edit Array dialog, in which you can specify the actions that are run when a user clicks on this push button.
edit configuration	Opens the GUIDE Configuration Editor for selecting, deleting, copying, or editing configurations.
configure	Applies the current configuration to the push button.

Menu Choice	Description
delete.	Opens a confirmation dialog to the user and if confirmed, deletes the push button.
move	Opens the Move Object dialog, which enables you to move the object precisely. This menu choice is visible only when the dialog is enabled.
clone.	Makes a copy of the button and places it next to the original button on the workspace.
disable./enable.	Disabling the push button prevents the select menu choice from being shown or run on the button's menu. The colors of the button change to reflect its disabled state.
select	Executes the uil-handler-method specified on the button. The handler method will subsequently execute the callback for the button.
initialize	Calls the initialization method of the push button. This menu choice is visible only when the push button is enabled.
edit push button	Opens the Edit Pushbutton dialog, which you can use to edit attributes of the push button.

Radio Buttons

Describes how to create and edit groups of radio buttons to provide users with sets of mutually exclusive choices.

Introduction **185**

Editing Radio Boxes **189**

Editing Radio Buttons **192**

Summary of Radio Box Menu Choices **196**

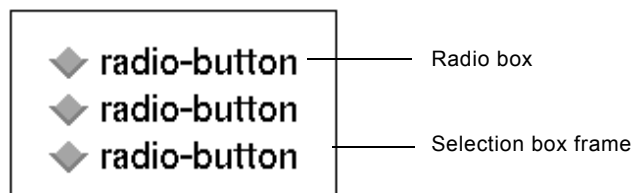
Summary of Radio Button Menu Choices **198**



Introduction

Radio buttons are used in groups of two or more buttons to represent mutually exclusive choices.

Each group of radio buttons is managed by a **radio box**. In a radio box, one and only one radio button is always selected. Selecting a radio button causes all other radio buttons in the same radio box to be deselected. By default, the radio box is surrounded by a selection box frame:



You can show and hide the selection box frame using the **hide selection box** and **show selection box** choices on the user menu of the radio box.

You can edit attributes of radio boxes and radio buttons to modify their appearance and behavior.

Selecting Motif or Windows Style Buttons

You can create Motif or Windows style buttons. You select the style of buttons that you want to create in the GUIDE Control Panel. For information about how to do this, see [Steps for Building a Master Dialog](#).

Adding Radio Buttons to a Master Dialog

Radio buttons cannot be used outside of radio boxes. To add radio buttons to a master dialog, you first click the radio box icon on the GUIDE palette and drop it on the dialog subworkspace. By default, a radio box contains three radio buttons.

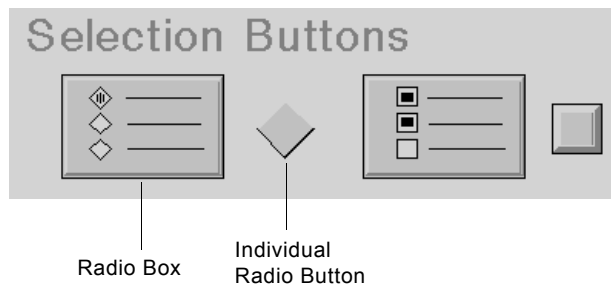
You add radio buttons to a radio box by clicking on the icon for individual radio buttons on the GUIDE palette and dropping the buttons on or near the radio box on the dialog subworkspace.

You can also add radio boxes and radio buttons to a master dialog by selecting the following choices from the GUIDE menu bar:

- Item > GUIDE Objects > uil-radio-box
- Item > GUIDE Objects > Buttons > uil-radio-button

The object that you select becomes attached to your cursor. You can drop it on the subworkspace of the master dialog that you are editing.

The GUIDE palette includes separate icons for radio boxes and for individual radio buttons:



Note In the icon for radio boxes, only one button is shown selected. This distinguishes the radio box icon from check box icon to its right, in which two buttons are selected.

Moving Radio Buttons

You can move an entire group of radio buttons by dragging the topmost button. The topmost button is called the **lead button**.

You can rearrange the buttons in a group by dragging individual buttons up or down within the group. You can add additional buttons to groups in any position. The lead button, when moved to a new location within the group, relinquishes its lead to the new topmost button.

Resizing Radio Buttons

You can change the size of a radio button by selecting a size (small, medium, or large) in the Edit Radio Button dialog.

You can also resize a radio button by choosing **change size** from its menu. When you select **change size**, a black border appears around the object, and a G2 dialog of resizing options appears on the workspace. For information about how to use the G2 resizing dialogs, see the *G2 Reference Manual*.

Deleting Radio Buttons and Radio Boxes

To delete a radio button from a radio box, choose **delete.** from the menu of the radio button. To delete a radio box and all the radio buttons that it contains, choose **delete.** from the menu of the radio box.

A radio box must contain at least two radio buttons in order to provide a set of mutually exclusive choices.

Updating and Concluding Radio Buttons

Radio buttons can be updated from source objects and can conclude their state into target objects. Radio buttons are commonly updated and concluded when a user clicks a push button that updates or concludes all UIL objects on the dialog. Update and conclude methods can be run on radio boxes, but not on individual radio buttons.

When an update method is run on a radio box, the update method examines the **on-value** of every radio button in the radio box. It then selects the radio button, if any, whose **on-value** attribute matches the update value that the update method is using. If there is no radio button whose **on-value** matches the update value, the update operation has no effect on the radio buttons.

When a conclude method is run on a radio box, the **on-value** of the currently selected radio button in the radio box is concluded to the target attribute of the target object for that radio button.

Specifying Labels for Radio Buttons

By default, the labels of Motif-style radio buttons created in GUIDE 5.0 or higher are contained in the icons of the buttons. Because the label is part of the icon, a Motif-style button and its label move and remain together whenever you drag either of them.

However, Windows-style radio buttons use separate label objects for labels, as in versions of GUIDE/UIL prior to 5.0.

For both Motif and Windows style radio buttons, you specify label text in the Edit Radio Button dialog. In this dialog, you also specify whether you want a radio button to resize itself to fit its label. For information about how to use the Edit Radio Button dialog, see [Editing Radio Buttons](#).

Upgrading Radio Buttons Created with Previous Versions of GUIDE/UIL

GUIDE/UIL supports radio buttons with separate label objects created in versions of GUIDE/UIL prior to 5.0. It does not automatically convert them to place their labels in their icons. For information about how to convert existing Motif-style radio buttons to the GUIDE/UIL 5.0 styles, see Chapter 26, “Upgrading Guide Applications” in the *G2 GUIDE User's Guide G2 Utilities Version 5.0* manual.

Using Label Objects for Radio Button Labels

Dialogs that contain Motif-style buttons created with GUIDE/UIL 5.0 or higher launch more quickly than dialogs that contain buttons with separate label objects. However, if you want to use separate label objects for button text, as was used in versions of GUIDE/UIL prior to 5.0, set the `uil-use-icon-text-for-buttons` parameter to `false`. You can set this parameter using the following menu choice from the GUIDE menu bar:

Tools > GUIDE 50r0 Migration Tools > Create 50r0 Buttons

When Create 50r0 Buttons is not selected, GUIDE/UIL creates a separate object to contain the label for each button that you create.

You can also set the `uil-use-icon-text-for-buttons` parameter using a `conclude` statement such as the following in an action button:

```
conclude that uil-use-icon-text-for-buttons is false
```

Caution Application code that accesses text in separate label objects will not work if you use the GUIDE/UII 5.0-style buttons that include button and text within a single object.

Do not attempt to extract text from buttons using relations. Instead, you can use the UIL procedure `uil-get-label-text` to return the `uil-text` object that provides the text of another object's label.

Editing Radio Boxes

The Edit Radio Box dialog enables you edit the size, label, location, and other properties of radio boxes.

To open the Edit Radio Box dialog, click the radio box and choose **edit radio box** from its menu. You can also open the Edit Radio Box dialog by choosing **edit radio box** from the menu of any radio button in the radio box.

The Edit Radio Box dialog looks like this:

The screenshot shows the 'Edit Radio Box' dialog box with the following settings:

- Class:** none
- Name:** NONE
- Label:** (empty)
- Id:** (empty)
- Label offset from buttons:** X: 5, Y: 25
- Current Value:** uil-none
- Box offset from buttons:** X: 10, Y: 5
- State:** Enabled, Disabled
- Orientation:** Vertical, Horizontal
- Height:** 97
- Width:** 166
- Buttons:** Source Object..., Target Object..., Methods...
- Bottom Buttons:** OK, Apply, Cancel

The following table describes the components of the Edit Radio Box dialog:

Components of Edit Radio Box Dialog

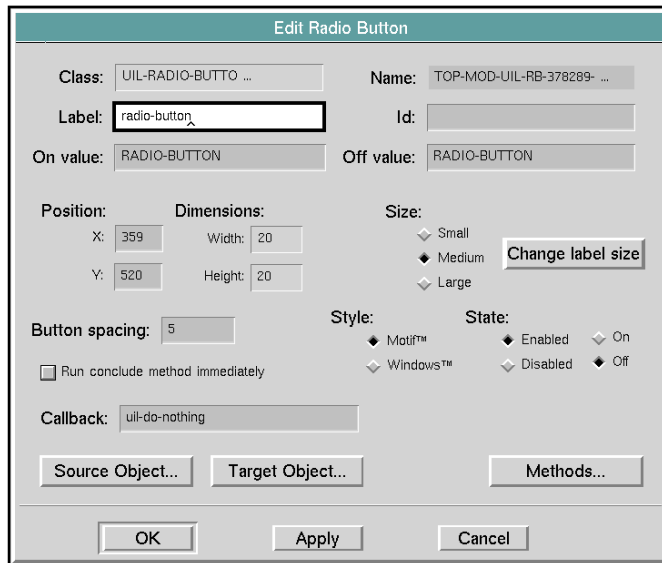
Component	Description
Class	(read-only) Displays the class of radio box that you are editing.
Name	Displays the current name of the radio box that you are editing. Changing the displayed value updates the name of the radio button. This field does not require a value. Its contents, if any, must be a valid symbolic value.
Label	(optional) Specify the text of the label that appears with the radio box.
Id	(optional) The ID of the radio box. Specify an ID for the radio box if you need to manipulate the radio box using a UIL procedure that references it by ID.
Label offset from buttons	Specify the horizontal (X) and vertical (Y) offset of the label from the radio box, in pixels. The X and Y values must be integers.
Current value	The On value of the currently selected radio button, or the value <code>uil-none</code> .
Box offset from buttons	The horizontal (X) and vertical (Y) offset of the radio button from the lead button in the radio box, in pixels. The X and Y values must be integers.
Source object	Opens the Edit Source Object & Attribute dialog, in which you can specify a source object for the radio buttons in this radio box. When the radio box is updated with the value of the source object, this value determines which radio button in the radio box is selected. For information about how specify the source object, see Edit Source Object & Attribute Dialog .

Components of Edit Radio Box Dialog

Component	Description
Target object	Opens the Edit Target Object & Attribute dialog, in which you can specify a target object for the radio buttons in this radio box. The value of the currently selected radio button is concluded to this object when the values in the dialog are concluded. For information about how to specify the target object, see Edit Target Object & Attribute Dialog .
State	Select the Enabled or Disabled button to enable or disable the radio box and all the radio buttons that it contains.
Orientation	Select Horizontal or Vertical to specify whether you want the radio buttons in the radio box to appear in a vertical column or in a horizontal row.
Methods	Opens the Edit Methods dialog, in which you can create customized methods for the radio box. For information about how to create customized methods, see Methods, Actions, and Callbacks .

Editing Radio Buttons

To open the Edit Radio Button dialog, choose **edit radio button** from the menu of the radio button that you want to edit. The Edit Radio Button dialog looks like this:



The following table describes the components of the Edit Radio Button dialog:

Components of Edit Radio Button Dialog

Component	Description
Class	(read-only) Displays the class of radio button that you are editing.
Name	(required) Displays the current name of the radio button that you are editing. Changing the displayed value updates the name of the radio button. Note that all radio buttons must have a name. Although the name is used primarily as a grouping mechanism (the selection box that groups selection buttons knows the name of each button in the group), you can change the name here for convenience.

Components of Edit Radio Button Dialog

Component	Description
Label	(optional) Displays the label of the radio button that you are editing. Changing the label causes a new label to be generated for the radio button with the new text. This is a non-required field, but if entered, it must be a valid text entry. Quotation marks are not required. If included, they will become part of the label.
Id	(optional) The ID of the radio button that you are editing. Changing the ID updates the <code>id</code> attribute of the radio button in its table.
On value	(optional) The current on-value for the radio button that you are editing. This is the value of the radio button when it is selected. Update and conclude methods require a value in this field. The default value is the symbol <code>radio-button</code> .
Off value	(optional) The current off-value for the radio button that you are editing. This is the value of the radio button when it is not selected. Update and conclude methods require a value in this field. The default value is the symbol <code>radio-button</code> .
Position	(required) The two edit box fields below the label Position display the current <code>item-x-position</code> and <code>item-y-position</code> of the radio button that you are editing. Changes to the X and Y values will move the radio button to the new location. This is a required field and defaults to the radio button's current position on the workspace. The X and Y values must be integers.
Dimensions	(read-only) The two text fields below the label Dimensions display the current <code>item-width</code> and <code>item-height</code> of the radio button that you are editing. You cannot edit this field.

Components of Edit Radio Button Dialog

Component	Description
Size	(required) The radio buttons below the label Size indicate the current size (small, medium, large) of the radio button that you are editing. You can change the size of the radio button that you are editing by selecting the radio button whose label indicates the desired size. When you apply this change, GUIDE regenerates all the radio buttons grouped by the same selection box in the new size without changing the buttons' other attributes.
Change label size	Applies the current selected size (small, medium, large) to all labels on radio buttons in the selection box.
Button spacing	(required) Displays the current number of workspace units used for the vertical spacing of the radio buttons. Changes to this value will cause the entire grouping of radio buttons to be re-displayed. The leader button (top-most) maintains its X/Y positioning. The remaining buttons are spaced accordingly.
Style	(required) The radio buttons below the label Style indicate the current window style (Motif or Window) of the radio button that you are editing. Users can change the window style by selecting the radio button whose label indicates the desired style. When this change is applied, GUIDE regenerates all the radio buttons grouped by the same selection-box in the new style without changing the buttons' other attributes.

Components of Edit Radio Button Dialog

Component	Description
State	<p>(required) The two sets of radio buttons below the label State indicate:</p> <ul style="list-style-type: none"> • Whether the radio button is enabled or disabled. • Whether the radio button is on or off. <p>Users can change the state by selecting the radio button whose label indicates the desired state. When this change is applied, the radio button is updated to reflect its new state.</p>
Run conclude method immediately	<p>The conclude method specified for this radio button is run as soon as a user clicks on the button. If you do not select the Run conclude method immediately option, the radio button is not concluded until a conclude method is run on the dialog (typically through an OK or Apply push button).</p>
Callback	<p>The callback procedure associated with this radio button. The callback is run when a user selects the radio button. The default callback is <code>uil-do-nothing</code>, which does nothing. For most purposes, radio buttons do not require their callbacks to perform any processing. However, if you need to perform special processing when a user selects a radio button, you can write a customized callback and use it in place of <code>uil-do-nothing</code>. For information about callbacks, see Methods, Actions, and Callbacks.</p>
Source Object	<p>Opens the Edit Source Object & Attribute dialog. The Edit Source Object & Attribute dialog enables you to specify the source object and attribute for the radio button that you are editing. For information about this dialog, see Edit Source Object & Attribute Dialog.</p>

Components of Edit Radio Button Dialog

Component	Description
Target Object	Opens the Edit Target Object & Attribute dialog. The Edit Target Object & Attribute dialog enables you to specify the target object and attribute for the radio button that you are editing. For information about this dialog, see Edit Target Object & Attribute Dialog .
Methods	<p>Opens the Edit Methods dialog, in which you can create customized methods for the radio button.</p> <p>For information about how to create customized methods, see Methods, Actions, and Callbacks.</p>

Summary of Radio Box Menu Choices

Radio Box Menu Choices

Menu Choice	Description
table	Shows the attribute table of the radio box.
transfer	Places the radio box on the mouse, so that you can drop it on a different workspace. The radio buttons in the radio box are transferred with it.
name	Opens an editor allowing a name to be entered or changed.
rotate/reflect	Opens a menu of rotation and reflection options: 90 clockwise, 90 counterclockwise, 180, left-right reflection, up-down-reflection.
change size	Opens a G2 dialog providing change size options.
color	Opens a series of G2 menus enabling you to change the colors of regions of the radio box. GUIDE provides the Configuration Editor as the recommended way to change icon regions. For information about this editor, see Using The GUIDE Configuration Editor .

Radio Box Menu Choices

Menu Choice	Description
lift to top	Displays radio box on top of another object on a workspace.
drop to bottom	Drops the radio box behind another object on a workspace.
describe	Shows a description of the object and its relations.
describe configuration	Displays a description of the object's current configuration.
edit array	Opens the Edit Array dialog, in which you can edit the member list of radio buttons in the radio box.
edit configuration	Posts GUIDE's Configuration Editor for selecting, deleting, copying, or editing configurations
configure	Applies the current configuration to the radio box.
delete.	Opens a confirmation dialog to the user and, if confirmed, deletes the radio box.
initialize	Calls the initialization method of the radio box. This menu choice is visible only when the radio box is enabled.
move	Opens the Move Object dialog, which enables you to move the object precisely. This menu choice is visible only when the dialog is enabled.
clone.	Makes a copy of the radio box and places it next to the original radio box on the workspace. This menu choice is visible only when the radio box is enabled.
enable./disable.	Enables or disables the radio box and the radio buttons that it contains.
resize box to fit buttons	Resizes the radio box to fit correctly around the radio buttons that the radio box contains. This user menu choice is useful if you add, delete, or modify radio buttons in the radio box.

Radio Box Menu Choices

Menu Choice	Description
hide/show selection box	Makes the radio box invisible or visible. Does not affect the visibility of the radio buttons in the radio box.
edit radio box	Opens the Edit Radio Box dialog, in which you can edit attributes of this radio box.

Summary of Radio Button Menu Choices

Menu Choice	Description
table	Shows the G2 table for the radio button.
transfer	Places the radio button on the mouse so that you can transfer it to a different workspace.
name	Opens an editor allowing a name to be entered or changed.
rotate/reflect	Opens a menu of rotation and reflection options.
change size	Opens a G2 dialog providing change size options.
color	Opens a series of dialogs that enable you to select colors for the radio button.
lift to top	Displays radio button on top of other objects on the workspace.
drop to bottom	Drops the radio button behind other objects on the workspace.
describe	Shows a description of the object and its relations. The description includes the radio button's name, module assignment, and relations to other objects.
describe configuration	Displays a description of the object's current configuration.
create subworkspace	Creates a subworkspace for this radio button.

Menu Choice	Description
edit configuration	Opens the GUIDE Configuration Editor, which enables you to select, delete, copy, or edit configurations.
configure	Applies the current configuration to the radio button.
delete.	Opens a confirmation dialog to the user and, if confirmed, deletes the radio button.
initialize	Calls the initialization method of the radio button. This menu choice is visible only when the radio button is enabled.
move	Opens the Move Object dialog, which you can use to position the radio button precisely. For information about this dialog, see the <i>G2 Reference Manual</i> .
clone.	Creates a copy (clone) of the radio button and places it at the bottom of the radio box that contains the original button.
disable./enable.	Disables or enables the radio button. Users cannot use disabled radio buttons.
select	Simulates the effect of selecting the radio button in a play mode.
edit radio box	Opens the Edit Radio Box dialog for editing attributes of the radio box that contains this radio button. For information about how to use this dialog, see Editing Radio Boxes .
edit radio button	Opens the Edit Radio Button dialog. For information about how to use this dialog, see Editing Radio Buttons .

Check Buttons

Describes how to create and edit groups of check buttons, in which users can select any number of choices.

Introduction **201**

Editing Check Boxes **205**

Editing Check Buttons **208**

Summary of Check Box Menu Choices **212**

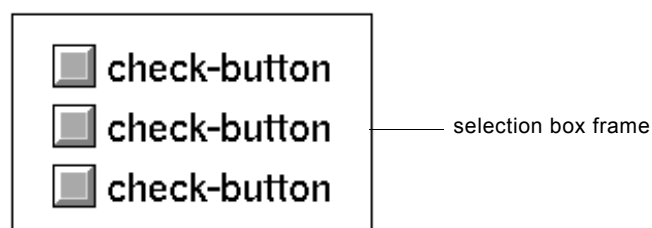
Summary of Check Button Menu Choices **214**



Introduction

Check buttons are used in groups of two or more buttons. In a group of check buttons, any number of buttons can be selected or unselected at the same time.

Each group of check buttons is managed by a check box. By default, the check box is surrounded by a selection box frame:



You can show and hide the selection box frame, using the **hide selection box** and **show selection box** choices on the user menu of the check box.

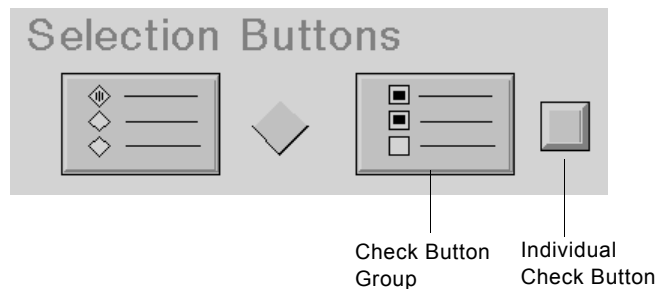
You can edit attributes of check boxes and check buttons to modify their appearance and behavior.

Selecting Motif or Windows Style Buttons

You can create Motif or Windows style buttons. You select the style of buttons that you want to create in the GUIDE Control Panel. For information about how to do this, see [Steps for Building a Master Dialog](#).

Adding Check Buttons to a Master Dialog

The GUIDE palette includes separate icons for check boxes and for individual check button:



Note In the icon for check boxes, two buttons are shown selected. This distinguishes the check box icon from radio box icon to its left, in which only one button is selected.

Check buttons cannot be used outside of check boxes. To add check buttons to a master dialog or workspace, you first add a check box. To do this, select the icon for check boxes on the GUIDE palette and drop the check box on the subworkspace of the master dialog or on the workspace. By default, a check box contains three check buttons.

To add a check button to an existing check box, select the icon for individual check buttons on the GUIDE palette and drop it on or near the check box. The button is automatically added to the check box when you drop it.

You can also add check boxes and check buttons to a master dialog or workspace by selecting the following choices from the GUIDE menu bar:

- Item > GUIDE Objects > uil-check-box
- Item > GUIDE Objects > Buttons > uil-check-button

The object that you select becomes attached to your cursor. You can drop it on the subworkspace of a master dialog or on a workspace.

You can delete a check button from a check box by selecting the check button and choosing **delete.** from its menu.

Moving Check Buttons

You can move an entire check box by dragging the topmost button. The topmost button is called the **lead button.**

You can rearrange the buttons in a check box by dragging individual buttons up or down within the check box. You can add additional buttons to check boxes in any position. The lead button, when moved to a new location within the check box, relinquishes its lead to the new topmost button.

Resizing Check Buttons

You can change the size of a check button by selecting a size (small, medium, or large) in the Edit Check Button dialog.

To resize a check button, choose **change size** from its menu. When you select **change size** or **change min size**, a black border appears around the check button, and a G2 dialog of resizing options appears on the workspace. For information about how to use the G2 resizing dialogs, see the *G2 Reference Manual.*

Updating and Concluding Check Buttons

Check buttons can be updated with a single value or with an array or list of values. When a check button is updated with a single value, the check button is selected if its **on-value** matches the update value. When a check button is updated with an array or list of values, the check button is selected if its **on-value** matches any one of the values in the array or list.

How check buttons conclude their values depends on whether all the check buttons in the check box have the same target object and target attribute:

- If all the check buttons in the check box have the same target object and target attribute, only the currently selected check buttons in the check box conclude their values. In this case, the check buttons conclude their **on-value** to the target attribute.
- If at least one check button in the check box specifies a different target attribute from the other check buttons, each check button concludes its **on-value** (if the button is selected) or its **off-value** (if the button is not selected) to the target attribute of the target object specified for that check button.

Specifying Labels for Check Buttons

By default, the labels of Motif-style check buttons created in GUIDE 5.0 or higher are contained in the icons of the buttons. Because the label is part of the icon, a Motif-style button and its label move and remain together whenever you drag either of them.

However, Windows-style check buttons use separate label objects for labels, as in previous versions of GUIDE/UIL.

For both Motif and Windows style check buttons, you specify label text in the Edit Check Button dialog. In this dialog, you also specify whether you want a check button to resize itself to fit its label. For information about how to use the Edit Check Button dialog, see [Editing Check Buttons](#).

Upgrading Check Buttons Created with Previous Versions of GUIDE/UIL

GUIDE/UIL supports check buttons with separate label objects created in versions of GUIDE/UIL prior to 5.0. It does not automatically convert them to place their labels in their icons. For information about how to convert existing Motif-style check buttons to the GUIDE/UIL 5.0 styles, see Chapter 26, “Upgrading Guide Applications” in the *G2 GUIDE User's Guide G2 Utilities Version 5.0* manual.

Using Label Objects for Check Button Labels

Dialogs that contain Motif-style buttons created with GUIDE/UIL 5.0 or higher launch more quickly than dialogs that contain buttons with separate label objects. However, if you want to use separate label objects for button text, as in previous versions of GUIDE/UIL, set the `uil-use-icon-text-for-buttons` parameter to `false`. You can set this parameter using the following menu choice from the GUIDE menu bar:

Tools > GUIDE 50r0 Migration Tools > Create 50r0 Buttons

When Create 50r0 Buttons is not selected, GUIDE/UIL creates a separate object to contain the label for each button that you create.

You can also set the `uil-use-icon-text-for-buttons` parameter using a `conclude` statement such as the following in an action button:

```
conclude that uil-use-icon-text-for-buttons is false
```

Caution Application code that accesses text in separate label objects will not work if you use the GUIDE/UIL 5.0-style buttons that include button and text within a single object.

Do not attempt to extract text from buttons using relations. Instead, you can use the UIL procedure `uil-get-label-text` to return the `uil-text` object that provides the text of another object's label.

Editing Check Boxes

The Edit Check Box dialog enables you to edit the size, label, location, and other properties of check boxes.

To open the Edit Check Box dialog, click the edit box and choose **edit check box** from its menu. You can also open the Edit Check Box dialog by choosing **edit check box** from the menu of any check button in the check box.

The Edit Check Box dialog looks like this

The screenshot shows the 'Edit Check Box' dialog box with the following fields and options:

- Class:** none
- Name:** NONE
- Label:** (empty)
- Id:** (empty)
- Label offset from buttons:**
 - X: 5
 - Y: 25
- Box offset from buttons:**
 - X: 10
 - Y: 5
 - Height: 114
 - Width: 178
- State:**
 - Enabled
 - Disabled
- Buttons:** Source Object..., Target Object..., Methods...
- Bottom Buttons:** OK, Apply, Cancel

The following table describes the components of the Edit Check Box dialog:

Components of Edit Check Box Dialog

Component	Description
Class	(read-only) Displays the class of check box that you are editing.
Name	(required) Displays the current name of the check box that you are editing. Changing the displayed value updates the name of the check button. This field does not require a value. Its contents, if any, must be a valid symbolic value.
Label	(optional) Specify the text of the label that appears with the check box.
Id	(optional) The ID of the check box. Specify an ID for the check box if you need to manipulate the check box, using a UIL procedure that references it by ID.
Label offset from buttons	Specify the horizontal (X) and vertical (Y) offset of the label from the check box. The X and Y values must be integers.
Box offset from buttons	Specify the horizontal (X) and vertical (Y) offset of the check box from the lead check button. The X and Y values must be integers.
State	Select the Enabled or Disabled button to enable or disable the check box and all the check buttons that it contains.
Source Object	Opens the Edit Source Object & Attribute dialog. The Edit Source Object & Attribute dialog enables you to specify the source object and attribute for the check button that you are editing. For information about this dialog, see Edit Source Object & Attribute Dialog .

Components of Edit Check Box Dialog

Component	Description
Target Object	Opens the Edit Target Object & Attribute dialog. The Edit Target Object & Attribute dialog enables you to specify the target object and attribute for the check button that you are editing. For information about this dialog, see Edit Target Object & Attribute Dialog .
Methods	Opens the Edit Methods dialog, in which you can create customized methods for the check box. For information about how to create customized methods, see Methods, Actions, and Callbacks .

Editing Check Buttons

To open the Edit Check Button dialog, click the check button that you want to edit and choose **edit check button** from the check button menu. The Edit Check Button dialog looks like this:

Edit Checkbutton

Class: UIL-CHECK-BUTTO ... **Name:** TOP-MOD-UIL-CB-112001- ...

Label: check-button **Id:**

On value: CHECK-BUTTON **Off value:** CHECK-BUTTON

Position: **Dimensions:** **Size:**

X: 559 Width: 20 Small

Y: 508 Height: 20 Medium

Large

Button spacing: 5 **Style:** **State:**

Run conclude method immediately Motif™ Enabled On

Windows™ Disabled Off

Callback: uil-do-nothing

The following table describes the components of the Edit Check Button dialog:

Components of Edit Check Button Dialog

Component	Description
Class	(read-only) Displays the class of check button that you are editing.
Name	<p>(required) Displays the current name of the check button that you are editing. Changing the displayed value updates the name of the check button.</p> <p>This field must contain a valid symbolic name. All check buttons must have a name.</p> <p>Although the name is used primarily as a grouping mechanism (the selection box that groups selection buttons knows the name of each button in the group), it may be changed here for convenience.</p>
Label	(optional) Displays the label of the check button that you are editing. Changing the label causes a new label to be generated for the check button with the new text. This is a non-required field, but if entered, it must be a valid text entry. Quotation marks are not required. If included, they become part of the label.
Id	(optional) Displays the ID of the check button that you are editing. Changing the ID updates the ID of the check button in its table.
On value	(optional) Displays the current on-value for the check button that you are editing. This is the value of the check button when it is selected. Update and conclude methods require this field to have a value. The default value is the symbol check-button.
Off value	(optional) Displays the current off-value for the check button that you are editing. This is the value of the check button when it is not selected. Update and conclude methods require this field to have a value. The default value is the symbol check-button.

Components of Edit Check Button Dialog

Component	Description
Position	(required) The two edit box fields below the label Position display the current item-x-position and item-y-position of the check button that you are editing. Changes to the X and Y values will move the check button to the new location. This is a required field and defaults to the check button's current position on the workspace. The X and Y values must be integers.
Dimensions	(read only) The two text fields below the label Dimensions display the current item-width and item-height of the check button that you are editing. You cannot edit this field.
Size	(required) The buttons below the label Size indicate the current size (small, medium, or large) of the check button that you are editing. You can change the size of the check button that you are editing by selecting the button whose label indicates the desired size. When you apply this change, GUIDE regenerates all the check buttons grouped by the same selection box in the new size without changing the buttons' other attributes.
Change label size	Applies the currently selected size (small, medium, or large) to the labels of the checks buttons in this selection box.
Button spacing	(required) Displays the current number of workspace units used for the vertical spacing of the check buttons. Changes to this value causes the entire grouping of check buttons to be re-displayed. The leader button (top-most) maintains its X/Y positioning. The remaining buttons are spaced accordingly.

Components of Edit Check Button Dialog

Component	Description
Style	(required) The buttons below the label Style (Motif, Windows) indicate the current window style of the check button that you are editing. You can change the window style by selecting the button whose label indicates the desired style. When you apply this change, GUIDE regenerates all the check buttons in the same selection box in the new style, without changing the buttons' other attributes.
State	(required) The buttons below the label State (Enabled, Disabled, On, Off), indicate the current state of the check button that you are editing. You can change the state by selecting the button whose label indicates the desired state. When you apply this change, the check button is updated to reflect its new state.
Run conclude method immediately	(optional) The conclude method specified for this check button is run as soon as a user clicks on the button. If you do not select the Run conclude method immediately option, the check button is not concluded until a conclude method is run on the dialog (typically through an OK or Apply push button).
Callback	The callback procedure associated with this check button. The callback is run when a user selects the check button. The default callback is <code>uil-do-nothing</code> , which does nothing. For most purposes, check buttons do not require their callbacks to perform any processing. However, if you need to perform special processing through a check button, you can write a customized callback and use it in place of <code>uil-do-nothing</code> . For information about callbacks, see Methods, Actions, and Callbacks .

Components of Edit Check Button Dialog

Component	Description
Source Object	Opens the Edit Source Object & Attribute dialog. The Edit Source Object & Attribute dialog enables you to specify the source object and attribute for the check button that you are editing. For information about this dialog, see Edit Source Object & Attribute Dialog .
Target Object	Opens the Edit Target Object & Attribute dialog. The Edit Target Object & Attribute dialog enables you to specify the target object and attribute for the check button that you are editing. For information about this dialog, see Edit Target Object & Attribute Dialog .
Methods	<p>Opens the Edit Methods dialog, in which you can create customized methods for the check button.</p> <p>For information about how to create customized methods, see Methods, Actions, and Callbacks.</p>

Summary of Check Box Menu Choices

Menu Choice	Description
table	Shows the attribute table of the check box
transfer	Places the check box on the mouse, so that you can drop it on a different workspace. The check buttons in the check box are transferred with it.
name	Opens an editor allowing a name to be entered or changed
rotate/reflect	Opens a menu of rotation and reflection options: 90 clockwise, 90 counterclockwise, 180, left-right reflection, up-down-reflection.
change size	Posts a G2 dialog providing change size options

Menu Choice	Description
color	Opens a series of G2 menus enabling you to change colors of icon regions. GUIDE provides the Configuration Editor as the recommended way to change colors. For information about this dialog, see Using The GUIDE Configuration Editor .
lift to top	Displays check box on top of another object on a workspace.
drop to bottom	Drops the check box behind another object on a workspace.
disable/enable	Disables or enables the check box.
describe	Shows a description of the object and its relations.
describe configuration	Displays a description of the object's current configuration.
edit array	Opens the Edit Array dialog, in which you can edit the member list of check buttons in the check box.
edit configuration	Posts GUIDE's configuration editor for selecting, deleting, copying, or editing configurations
configure	Applies the current configuration to the check box.
delete.	Posts a confirmation dialog to the user and, if confirmed, deletes the check box.
initialize	Calls the initialization method of the check box. This menu choice is visible only when the check box is enabled.
move	Opens the Move Object dialog, which enables you to move the object precisely. This menu choice is visible only when the dialog is enabled.
clone.	Makes a copy of the check box and places it next to the original check box on the workspace. This menu choice is visible only when the check box is enabled.

Menu Choice	Description
disable./enable.	Enables or disables the check box and the check buttons that it contains.
resize box to fit buttons	Resizes the check box to fit correctly around the check buttons that the check box contains. This user menu choice is useful if you add, delete, or modify check buttons in the check box.
hide/show selection box	Makes the check box invisible or invisible. Does not affect the visibility of the check buttons in the check box.
edit check box	Opens the Edit Check Box dialog, in which you can edit attributes of the check box.

Summary of Check Button Menu Choices

Menu Choice	Description
table	Shows the attribute table of the check button.
transfer	Places the check button on the mouse so that you can transfer it to a different workspace.
name	Opens an editor allowing a name to be entered or changed.
rotate/reflect	Opens a menu of rotation and reflection options: 90 clockwise, 90 counterclockwise, 180, left-right reflection, up-down-reflection.
change size	Opens a G2 dialog providing change size options.
color	Opens a series of G2 menus allowing icon regions to have their colors altered. GUIDE provides the Configuration editor as a preferred means of changing icon regions.
lift to top	Displays check button on top of another object on a workspace.
drop to bottom	Drops the check button behind another object on a workspace.

Menu Choice	Description
describe	Shows a description of the object and its relations. The description includes the check button's name, module assignment, and relations to other objects.
describe configuration	Displays a description of the object's current configuration.
create subworkspace	Creates a subworkspace for this check button.
edit configuration	Opens the GUIDE Configuration Editor, which enables you to select, delete, copy, or edit configurations.
configure	Applies the current configuration to the check button.
delete.	Opens a confirmation dialog to the user and if confirmed, deletes the check button.
initialize	Calls the initialization method of the check button. This menu choice is visible only when the check button is enabled.
move	Opens the Move Object dialog, which you can use to position the check button precisely. For information about this dialog, see the <i>G2 Reference Manual</i> .
select	Simulates the effect of selecting the check button in a play mode.
clone.	Makes a copy of the check button and places it at the bottom of the check box that contains the original button.
disable/enable	Disables a check button. Prevents the select menu choice from being shown or run on the button. Also changes the colors of the button to reflect its disabled state

Menu Choice	Description
edit check box	Opens the Edit Check Box dialog, in which you can edit attributes of the check box that contains this check button.
edit check button	Opens Edit Check Button dialog for editing the various attributes of the check button. For information about this editor, see Editing Check Buttons .

Toggle Buttons

Describes how to create and edit toggle buttons, which represent two mutually exclusive choices.

Introduction **217**

Editing Toggle Buttons **220**

Summary of Toggle Button Menu Choices **225**



Introduction

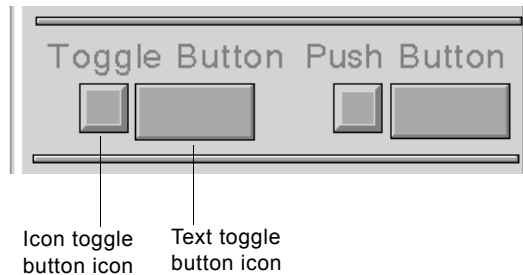
A **toggle button** can represent two mutually exclusive values. One value is chosen when the toggle button is selected, and the other value is chosen when the toggle button is unselected. These values are stored in attributes named **on-value** (button is selected) and **off-value** (button is not selected).

Selecting Motif or Windows Style Buttons

You can create Motif or Windows style buttons. You select the style of buttons that you want to create in the GUIDE Control Panel. For information about how to do this, see [Steps for Building a Master Dialog](#).

Adding Toggle Buttons to a Master Dialog

The GUIDE palette provides icons for two kinds of toggle buttons, icon toggle buttons and text toggle buttons:



Icon toggle buttons and text toggle buttons behave exactly the same; they differ from each other only in appearance:



You can add toggle buttons to a master dialog by clicking on either of these icons and dragging to the subworkspace of the master dialog. You can also add toggle buttons by selecting either of these choices from the GUIDE menu bar:

- Item > GUIDE Objects > Buttons > uil-icon-toggle-button
- Item > GUIDE Objects > Buttons > uil-text-toggle-button

The icon for the class of toggle button that you select becomes attached to your cursor. You can drop the button on the subworkspace of the master dialog that you are editing.

Updating and Concluding Toggle Buttons

When a conclude method is run on a toggle button, the current value of the toggle button (**on-value** if the button is selected or **off-value** if it is unselected) is concluded to the target attribute of the toggle button's target object.

When an update method is run on a toggle button, the toggle button is selected if it receives a value that matches its **on-value**, and is unselected if it receives a value that matches its **off-value**. If it receives any other value, the selection state of the toggle button is not affected.

For example, the **on-value** of a toggle button can be set to **running** and its **off-value** can be set to **stopped**. The initiating object of the toggle button is an object

representing a tank that has an attribute named `tank-state`. The toggle button has the following attribute values:

Toggle Button Attribute	Value
<code>uil-event-target-object</code>	<code>initiating-object</code>
<code>uil-event-target-attribute</code>	<code>tank-state</code>
<code>uil-event-source-object</code>	<code>initiating-object</code>
<code>uil-event-source-attribute</code>	<code>tank-state</code>

When the update method is run on the toggle button, the value of the `tank-state` attribute of the tank is automatically reflected in the state of the toggle button. If the value of the attribute `tank-state` is `stopped`, then the toggle button is left unselected. If the value of `tank-state` is `running`, then the toggle button is selected.

When the conclude method is run on the toggle button, the state of the toggle button is concluded into the `tank-state` attribute. If the value of toggle button is selected, then its `on-value` is concluded into the `tank-state` attribute of the initiating object. If the toggle button is unselected, then its `off-value` is concluded into `tank-state`.

Specifying Labels for Toggle Buttons

By default, the labels of Motif-style toggle buttons created in GUIDE 5.0 or higher are contained in the icons of the buttons. Because the label is part of the icon, a Motif-style button and its label move and remain together whenever you drag either of them.

However, Windows-style toggle buttons use separate label objects for labels, as in versions of GUIDE/UIIL prior to 5.0.

For both Motif and Windows style toggle buttons, you specify label text in the Edit Toggle Button dialog. In this dialog, you also specify whether you want a toggle button to resize itself to fit its label. For information about how to use the Edit Toggle Button dialog, see [Editing Toggle Buttons](#).

Upgrading Toggle Buttons Created with Previous Versions of GUIDE/UIIL

GUIDE/UIIL supports toggle buttons with separate label objects created in versions of GUIDE/UIIL prior to 5.0. It does not automatically convert them to place their labels in their icons. For information about how to convert existing Motif-style toggle buttons to the GUIDE/UIIL 5.0 styles, see Chapter 26, "Upgrading Guide Applications" in the *G2 GUIDE User's Guide G2 Utilities Version 5.0* manual.

Using Label Objects for Toggle Button Labels

Dialogs that contain Motif-style buttons created with GUIDE/UIL 5.0 or higher launch more quickly than dialogs that contain buttons with separate label objects. However, if you want to use separate label objects for button text, as in previous versions of GUIDE/UIL, set the `uil-use-icon-text-for-buttons` parameter to `false`. You can set this parameter using the following menu choice from the GUIDE menu bar:

Tools > GUIDE 50r0 Migration Tools > Create 50r0 Buttons

When `Create 50r0 Buttons` is not selected, GUIDE/UIL creates a separate object to contain the label for each button that you create.

You can also set the `uil-use-icon-text-for-buttons` parameter using a `conclude` statement such as the following in an action button:

```
conclude that uil-use-icon-text-for-buttons is false
```

Caution Application code that accesses text in separate label objects will not work if you use the GUIDE/UIL 5.0-style buttons that include button and text within a single object.

Do not attempt to extract text from buttons using relations. Instead, you can use the UIL procedure `uil-get-label-text()` to return the `uil-text` object that provides the text of another object's label.

Editing Toggle Buttons

You can edit the appearance and behavior of toggle buttons using the `Edit Toggle Button` dialog, `Edit Source Object & Attribute` dialog, `Edit Target Object & Attribute` dialog, and `Edit Methods` dialog.

Edit Toggle Button Dialog

To open the Edit Toggle Button dialog, click the toggle button that you want to edit and choose **edit toggle button** from the toggle button menu. The Edit Toggle Button dialog looks like this:

The following table describes the components of the Edit Toggle Button dialog:

Components of Edit Toggle Button Dialog

Component	Description
Class	(read-only) Displays the class of toggle button that you are editing.
Name	(optional) Displays the current name of the toggle button that you are editing. Changing the displayed value updates the name of the toggle button. This field does not require a value. Its contents, if any, must be a valid symbolic name.

Components of Edit Toggle Button Dialog

Component	Description
Id	(optional) Displays the ID of the toggle button that you are editing. Changing the ID updates the ID of the toggle button in its table.
Label	(optional) Displays the label of the toggle button that you are editing. Changing the label will cause a new label to be generated for the toggle button with the new text. This is a non-required field, but if entered, it must be a valid text entry. Quotation marks are not required. If included, they will become part of the label.
Toggle Label	(optional) Displays the toggle-label of the toggle button that you are editing. The text used for the toggle label is displayed when the toggle button is toggled on. This is a non-required field, but if entered, it must be a valid text entry. Quotation marks are not required. If included, they will become part of the label.
Off value	(optional) Displays the current off-value for the toggle button that you are editing. This is the value of the toggle button when it is not selected. The update and conclude methods for dialogs require this field to have a value. The default value is the symbol <code>off</code> .
On value	(optional) Displays the current on-value for the toggle button that you are editing. This is the value of the toggle button when it is selected. The update and conclude methods for dialogs require this field to have a value. The default value is the symbol <code>on</code> .

Components of Edit Toggle Button Dialog

Component	Description
Position	(required) The two edit box fields below the label Position display the current item-x-position and item-y-position of the toggle button that you are editing. Changes to the X and Y values will move the toggle button to the new location. This is a required field and defaults to the toggle button's current position on the workspace. The X and Y values must be integers.
Dimensions	(read only) The two text fields below the label Dimensions display the current item-width and item-height of the toggle button that you are editing. You cannot edit this field.
Size	(required) The radio buttons below the label Size indicate the current size (small, medium, or large) of the toggle button that you are editing. You can change the size of the toggle button that you are editing by selecting the radio button whose label indicates the desired size. When this change is applied, GUIDE regenerates the toggle button in the new size without changing the button's other attributes.
Style	(required) The radio buttons below the label Style (Motif, Windows) indicate the current window style of the toggle button that you are editing. Users can change the window style by selecting the radio button whose label indicates the desired style. When this change is applied, GUIDE regenerates the toggle button in the new style without changing the button's other attributes.
State	The Enabled and Disabled radio buttons below the label State indicate the current state of the toggle button that you are editing. You can change the state by selecting the radio button whose label indicates the desired state. When this change is applied, the toggle button is updated to reflect its new state.

Components of Edit Toggle Button Dialog

Component	Description
Callback	The callback procedure associated with this toggle button. The callback is run when a user clicks on the toggle button. The default callback is <code>uil-do-nothing</code> , which does nothing. For most purposes, toggle buttons do not require their callbacks to perform any processing. However, if you need to perform special processing through a toggle button, you can write a customized callback and use it in place of <code>uil-do-nothing</code> . For information about callbacks, see Methods, Actions, and Callbacks .
Resize button to fit label	If this option is selected (the default), GUIDE/UII automatically resizes the toggle button to fit the label text that you specify for the button. If this option is not selected, the button is not resized automatically.
Run conclude method immediately	If this option is selected, the value of this button is concluded to its target object whenever a user clicks on the button to toggle it on or off. If this option is not selected, the value of the button is not concluded until a conclude method is run on the dialog that contains the button.
Source Object	Opens the Edit Source Object & Attribute dialog. The Edit Source Object & Attribute dialog enables you to specify the source object and attribute for the toggle button that you are editing. For information about this dialog, see Edit Source Object & Attribute Dialog .

Components of Edit Toggle Button Dialog

Component	Description
Target Object	Opens the Edit Target Object & Attribute dialog. The Edit Target Object & Attribute dialog enables you to specify the target object and attribute for the toggle button that you are editing. For information about this dialog, see Edit Target Object & Attribute Dialog .
Methods	<p>Opens the Edit Methods dialog, in which you can create customized methods for the toggle button.</p> <p>For information about how to create customized methods, see Methods, Actions, and Callbacks.</p>

Summary of Toggle Button Menu Choices

Menu Choice	Description
table	Shows the G2 table for the toggle button.
transfer	Places the toggle button on the mouse and transfers it to a different workspace.
name	Opens an editor allowing a name to be entered or changed.
rotate/reflect	Opens a G2 dialog providing rotation and reflection options: 90 clockwise, 90 counterclockwise, 180, left-right reflection, up-down-reflection.
change size	Opens a G2 dialog providing change size options.
color	Opens a series of G2 menus allowing icon regions to have their colors altered. GUIDE provides the Configuration Editor as a preferred means of changing the colors of icon regions.
lift to top	Displays toggle button on top of another object on a workspace.

Menu Choice	Description
drop to bottom	Displays toggle button behind another object on a workspace.
describe	Shows a description of the object and its relations.
describe configuration	Displays a description of the button's current configuration.
create subworkspace	Creates and shows a subworkspace for the button.
edit configuration	Opens the Configuration Editor for selecting, deleting, copying, or editing configurations.
configure	Applies the current configuration to the button. This menu choice is visible only when the button is disabled.
delete.	Opens a confirmation dialog to the user and, if confirmed, deletes the button.
initialize	Calls the initialization method of the toggle button. This menu choice is visible only when the toggle button is enabled.
move	Opens the Move Object dialog, which enables you to move the object precisely. This menu choice is visible only when the toggle button is enabled.
select	Executes the uil-handler-method specified on the button. The handler method subsequently executes the callback specified for the button.
clone.	Makes a copy of the button and places it next to the original button on the workspace.
disable./enable.	Disabling the toggle button prevents the select menu choice from being shown or run on the button. The colors of the button change to reflect its disabled state.
edit toggle button	Opens the Edit Toggle Button dialog, in which you can edit the attributes of the toggle button.

Edit Boxes, Combo Boxes, and Spin Controls

Describes how to create and customize edit boxes, combo boxes, and spin controls.

Introduction **227**

Editing Edit Boxes **232**

Background Color and Text Color Dialogs **247**

Combo Boxes **247**

Spin Control Boxes **250**

Summary of Edit Box, Combo Box, and Spin Control Menu Choices **252**

Summary of Spin Control Box Menu Choices **253**

Summary of Combo Box Menu Choices **255**



Introduction

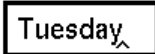
Edit boxes are fields in which an application can display textual information to users, and users can input textual information to the application.

The GUIDE palette provides icons for four different kinds of edit boxes:

- Edit boxes that can contain only a single line.
- Scrollable edit boxes that can contain more than one line of text. Edit boxes containing more than one line have scroll bars.

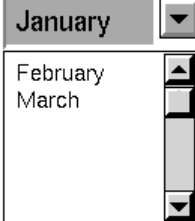
- Spin control edit boxes, which provide scrollable lists of values.
- Combo edit boxes, which are edit boxes combined with scrollable lists of text items.

The different kinds of edit boxes look like this:

Single Line Edit Box: 

Scrollable Edit Box: 

Spin Control Edit Box: 

Combo Edit Box: 

You can add edit boxes to a master dialog by clicking the icon for the class of edit box that you want to create and dropping it on the subworkspace of the master dialog. You can also add edit boxes by selecting the following choices from the GUIDE menu bar:

- Item > GUIDE Objects > Texts > uil-edit-box
- Item > GUIDE Objects > Texts > multi-line-uil-edit-box
- Item > GUIDE Objects > Texts > uil-spin-control
- Item > GUIDE Objects > Texts > uil-combo-box

Different sizes and styles of edit boxes are listed under each of these choices. Select the class of edit box that you want to add. The icon for that class of edit box becomes attached to your cursor. You can drop it on the subworkspace of the master dialog that you are editing.

For each edit box, you can specify the initial contents, the behavior of the editor for that edit box, validation criteria for data that users enter into the edit box, and the source and target objects and attributes of the edit box.

Setting the Initial Contents of Edit Boxes

You can specify initial contents for an edit box by choosing the `edit` user menu choice. This menu choice opens the editor on the edit box. The full, unformatted contents of the edit box are stored in the `message-contents` attribute of the edit

box. A formatted and clipped version of the text is stored in the `text` attribute of the edit box, which you cannot view or edit directly.

Edit Styles for Edit Boxes

You can specify different edit styles for edit boxes to control how the editor behaves when it is opened on the edit boxes, in all user modes.

An edit style can specify editing features such as the language in which the editor displays menus and prompts when it is opened on an edit box, and whether an edit box can display more than one line of text. Edit styles are instances of the class `uil-field-edit-style`.

GUIDE provides a number of useful system-defined edit styles. You can also create your own reusable edit styles.

Edit styles apply rules to edit boxes that govern the following:

- Aspects of editor behavior, such as whether the edit box can accept more than one line of text, whether the edit box is cleared at the beginning of an edit, and whether the editor displays edit buttons, menus, and grammatical prompts.
- The font size and color of text.
- The background color of the edit box.
- The default size of the editor.
- Whether the editor displays menus and buttons.
- The natural language, such as English, French, or German, in which the text of the label is displayed.

For information about how to edit an edit style, see [Creating and Editing an Edit Field Edit Style](#).

Validating the Contents of Edit Boxes

You can associate procedures known as **validation methods** with edit boxes to validate the edits that a user makes to the contents of the edit box. The procedure that performs the validation for an edit box is specified in the `uil-validation-method` attribute of the edit box. The default validation method for edit boxes is `uil-validate-grobj-method`.

The validation method for an edit box compares the contents of the edit box with an existing format, which is specified in the `uil-format-specification` attribute of the edit box. Each format specifies validation criteria for the contents of the edit box, such as the data type and format, minimum and maximum length, minimum and maximum value, and other criteria. The format also controls how quotation marks are displayed and how cases are handled.

You can specify that the validation method is run on the edit box when a user finishes editing it – for example, when a user presses the Tab or Return key, or clicks on some other edit box in the dialog. If you do not specify that validation is to occur immediately when a user finishes editing, the contents of the edit box are validated only when a validation method is run on the dialog – for example, when a user clicks a push button (typically, an OK or Apply button).

You select a format and validation options for an edit box through the `edit edit box` dialog. For information about the this dialog, see [Keyboard Navigation to Edit Boxes](#).

Because UIL cannot anticipate all types of validation that may be needed by the application, you can create your own validation procedures and functions to use in place of those provided. Supported types of validation are: data type checking, date and time formats, list member lookup, and range-checking.

How GUIDE Validates Edit Boxes

The validation method `uil-call-validate-method` validates the contents of edit boxes according to these rules:

- 1 If the value remains unchanged (that is, if the user has not altered the value but tabbed or aborted out of the edit), then the validation method is not run, even if it is specified.
- 2 The attribute `uil-validate-value-immediately`, if true, applies the validation method immediately upon exit from the edit session on the field.
- 3 When the validation method is run on a dialog, every edit-box in the `uil-dialog-gobj-list` of the dialog which specifies a validation method is validated. Each validation error, in turn is reported until all of the errors have been corrected or the dialog canceled.
- 4 If a validation failure occurs, the user must enter a value that satisfies the validation criteria before the next edit box is validated.
- 5 The validation method validates values using criteria specified in a format object of the class `uil-format-specification-object`. For information about formats, see [Formats and Validation Criteria](#).
- 6 You can specify your own validation procedures and functions in the format selected for validation.
- 7 Validation failures, by default, are handled by the built-in error handler. This error handler posts a dialog on the window that explains the reason for the validation failing. If information is available about the nature of the failure and the expected value or type of value, this information is displayed.
- 8 You can create and use a user-defined validation failure handler. This handler is used rather than the default handler.

A validation failure method can be referenced from the `uil-validation-failure` method attribute of a format. By default, the validation failure method of a format is unspecified. You can create a validation failure method using the GUIDE Method Help dialog. For information about how to use this dialog, see [Creating Callbacks, Methods, Procedures, Functions, and Actions Using the GUIDE Method Help Dialog](#).

Keyboard Navigation to Edit Boxes

Users can navigate from edit box to edit box in a dialog by pressing the Tab, Return, and abort (Ctrl + a) keys. Pressing these keys moves the editor forward in the sequence of edit boxes on the dialog, beginning with the edit box in the top left corner of the dialog and moving horizontally from left to right, and vertically from top to bottom. When the editor reaches the last edit box in the dialog, it can move forward by returning to the edit box in the top left corner of the dialog. Keyboard navigation is available only for edit boxes created from the system-defined class `uil-edit-box` or subclasses of `uil-edit-box`.

You can modify the behavior of the Tab, Return, and abort (Ctrl + a) keys to specify whether pressing these keys moves the editor ahead to the next edit box or to the first edit box in the next row. You can also specify that pressing the key exits the editor. For information about how to edit the behavior of these keys, see [Editor Behaviors Dialog](#). Users can also move the editor to another edit box on the same dialog or another dialog by clicking on the other edit box.

Disabling Keyboard Navigation to an Edit Box

If you do not want users to be able to access an particular edit box through keyboard navigation, you can set the `uil-allow-field-edit-of-this-message` attribute of the edit box to `false`. When this attribute is set to `false`, users can open the editor on the edit box only by clicking on the edit box. Setting this attribute to `true` enables users to open the editor on the edit box by pressing the tab, return, and abort key until the editor reaches the edit box.

Customizing Before and After Method Processing (Optional)

GUIDE/UII provides system-defined selection and unselection methods that perform special processing at the beginning and at the end of edit sessions on the edit boxes.

The selection method for an edit box is called when the edit box is selected. The unselection method is called when the edit session on the edit box is finished — for example, when you click another edit box, or when a user clicks on another edit box or clicks a key to navigate to another edit box.

The default selection method is `uil-edit-box-selection-method`, and the default unselection method is `uil-edit-box-unselection-method`. These methods provide support for opening and closing the editor on edit boxes.

Your application can use the default selection and unselection methods for most purposes. If you want to perform any additional, specialized processing on edit boxes when you open and close them for editing, you can write your own selection and unselection methods to use in place of the default methods.

Note Any customized selection method that you write should include a call to the lower-level procedure `uil-edit-box-selection`. Any customized unselection method should include a call to the lower-level procedure `uil-edit-box-unselection`. These lower-level procedures are required for starting and closing edit sessions on edit boxes.

Selection and unselection methods are available only for edit boxes and for edit boxes created from subclasses of the system-defined edit box classes. These methods are used in all user modes.

Updating and Concluding Edit Boxes

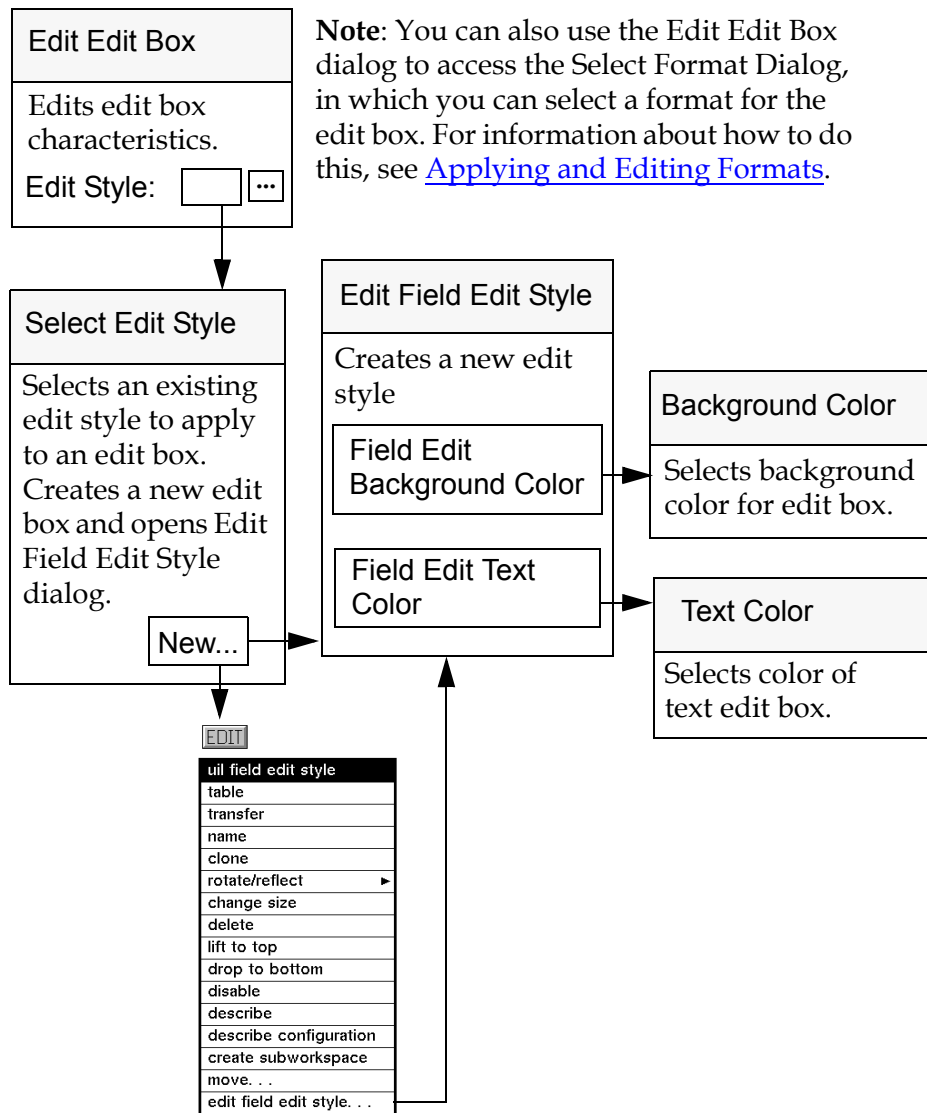
The text in edit boxes can be updated from and concluded to other objects. You can specify the source and target objects and attributes of edit boxes by selecting the Source Object and Target Object buttons in the Edit Edit Box dialog.

GUIDE/UIIL provides default methods for performing the update and conclude operations on edit boxes. Your application can use the default methods for most purposes. For information about how to create and use customized methods for updating and concluding edit boxes, see [Methods, Actions, and Callbacks](#).

Editing Edit Boxes

You can edit the appearance and behavior of edit boxes using the Edit Edit Box dialog, the Select Edit Style dialog, the Edit Field Edit Style dialog, and the Background Color and Text Color dialogs. The following figure illustrates these dialogs and how to access them.

Dialogs for Editing Edit Boxes and Edit Styles



Edit Edit Box Dialog

The Edit Edit Box dialog enables you to edit characteristics of an edit box such as its name, label, ID, value, position, state, and run-time options.

You can also select an existing edit style, or define a new one, through dialogs that you access from the Edit Edit Box dialog.

For more information about the Select Edit Style and Edit Field Edit Style dialogs, see the sections immediately following this one.

To open the Edit Edit Box dialog, click the edit box that you want to edit and choose edit edit box from the edit box menu. The Edit Edit Box dialog looks like this:

Edit Edit Box

Class: UIL-EDIT-BOX-MEDIU ... **Name:** UNSPECIFIED

Label: Label **Id:**

Value:

Position: **Dimensions:** **Size:** **State:**

X: 461 Width: 100 ◆ Small ◆ Enabled

Y: 559 Height: 33 ◆ Medium ◆ Disabled

◆ Large

Options:

Run validate method immediately Bounding box

Run conclude method immediately Clear field on edit

Maximum characters to display: ◆ Unlimited Allow field edit

◆ 10 Use target grammar

Format: unspecified ...

Edit Style: uil-edit-box-default-field-edit-style ...

Source Object... Target Object... Methods...

OK Apply Cancel

The following table describes the components of the Edit Edit Box dialog:

Components of Edit Edit Box Dialog

Component	Description
Class	(read-only) Displays the class of edit box that you are editing.
Name	(optional) Displays the current name of the edit box that you are editing. Changing the displayed value updates the name of the edit box. This field does not require a value. Its contents, if any, must be a valid symbolic name entry.
Label	(optional) Displays the current text of the edit box's label. You can edit the contents of this field to change the label text.
Id	(optional) Displays the ID of the edit box that you are editing. Changing the ID updates the id attribute of the edit box.
Value	<p>(optional) Displays the unformatted contents of the edit box that you are editing. You can edit the Value field to modify the edit box's unformatted contents. The unformatted contents are stored in the message-contents attribute of the edit box.</p> <p>If you specify a format for the edit box (through the Select Format dialog), this format is applied to the contents of the message-contents attribute, and the resulting formatted text is stored in the text attribute of the edit box. Any limitation on the contents of the edit box that you specify in the Maximum Characters to Display field (see below) is also applied to the formatted text in the text attribute. You cannot directly access or edit the text attribute.</p>

Components of Edit Edit Box Dialog

Component	Description
Position	(required) The two edit box fields below the label Position display the current item-x-position and item-y-position of the edit box that you are editing. Changes to the X and Y values will move the edit box to the new location. This is a required field and defaults to the edit box's current position on the workspace. The X and Y values must be integers.
Dimensions	(read-only) The two text fields below the label Dimensions display the current item-width and item-height of the edit box that you are editing. You cannot edit this field.
Size	(required) The radio buttons below the label Size indicate the current size (small, medium, or large) of the edit box that you are editing. You can change the size of the edit box by selecting the radio button whose label indicates the desired size. When this change is applied, GUIDE regenerates the edit box in the new size without changing the object's other attributes.
State	(required) The radio buttons below the label State indicate the current enabled or disabled state of the object that you are editing. Users can change the state by selecting the radio button whose label indicates the desired state. When this change is applied, the edit box is updated to reflect its new state.

Components of Edit Edit Box Dialog

Component	Description
Options	
Run validate method immediately	<p>If you select this option, the validation method for this edit box is run as soon as the user finishes editing the contents of the edit box. The validation method is specified in the <code>uil-validation-method</code> attribute of the edit box. This method applies validation criteria specified in the format that is currently selected for this edit box.</p> <p>If you do not select this option, the contents of the edit box are not validated until a validation method is run on the dialog, typically through an OK or Apply push button.</p>
Run conclude method immediately	<p>If you select this option, the conclude method for this edit box is run as soon as the user finishes editing the contents of the edit box. The conclude method is specified in the <code>uil-conclude method</code> attribute of the edit box.</p> <p>If you do not select this option, the contents of the edit box are not concluded until a conclude method is run on the dialog, typically through an OK or Apply push button.</p>

Components of Edit Edit Box Dialog

Component	Description
Maximum characters to display	<p>(required) Specifies the maximum number of characters that can be displayed in the edit box. Any characters in excess of this number are clipped. The default value is the 10 characters.</p> <p>Specifying a value in the Maximum characters to display field prevents the edit box from expanding beyond its original dimensions. This is particularly useful for maintaining dialog layout and item alignment.</p> <p>The full, unformatted text is maintained in the <code>message-contents</code> attribute of the edit box. The clipped and/or formatted text is placed in an attribute named <code>text</code>, which you cannot view or edit directly.</p>
Bounding box	<p>(optional) When this button is selected, GUIDE creates a border and wraps it around the edit box. The border is given a name and is referred to in the edit box's <code>uil-border-relation</code> attribute. If the edit box has a border and you turn this option off, the border is deleted and the <code>uil-border-relation</code> attribute is set to be the symbol <code>unspecified</code>.</p>
Clear field on edit	<p>(optional) If this button is selected, GUIDE clears out the text in the edit box whenever the edit box receives focus for editing. If the toggle button is unselected, then the text is not cleared from the edit box.</p>
Allow field edit	<p>(optional) If this button is selected, users can navigate to this edit box by pressing the tab, return, or abort key. Navigating to the edit box opens the field editor on it.</p> <p>If this button is not selected, users can open the field editor on this edit box only by clicking on the edit box.</p>

Components of Edit Edit Box Dialog

Component	Description
Use target grammar	If this button is selected, the editor for the edit box displays look-ahead grammatical prompts that are appropriate to target object of this edit box. This target grammar takes precedence over the grammar of the Edit Style specified for this edit box.
Format	<p>You can enter the name of an existing format to specify a format for this edit box. For information about how to use formats, see Formats and Validation Criteria.</p> <p>Clicking on the push button to the right of the Format edit box opens the Select Format dialog, in which you can select a format for this edit box. For more information about this dialog, see Creating Formats.</p>
Edit Style	<p>You can enter the name of an existing edit style to apply to this edit box. For information about how to use edit styles, see Edit Styles for Edit Boxes.</p> <p>Clicking on the push button to the right of the Edit Style edit box opens the Select Edit Style dialog, which enables you to choose an existing edit style for the edit box that you are editing. See the following section for more information about the Select Edit Style dialog.</p>
Source Object	Opens the Edit Source Object & Attribute dialog. The Edit Source Object & Attribute dialog enables you to specify the source object and attribute for the edit box that you are editing. For information about this dialog, see Edit Source Object & Attribute Dialog .

Components of Edit Edit Box Dialog

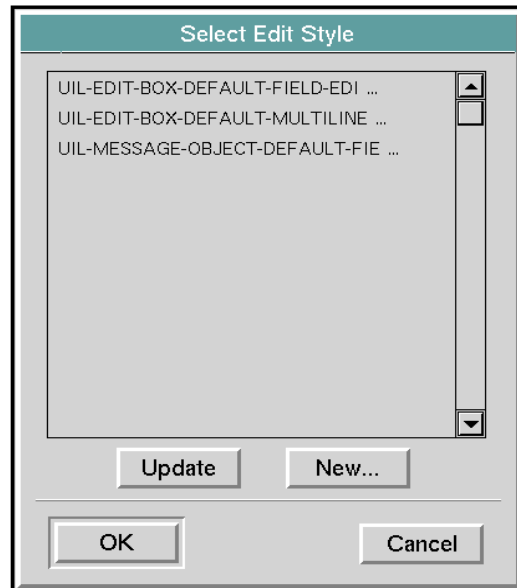
Component	Description
Target Object	Opens the Edit Target Object & Attribute dialog. The Edit Target Object & Attribute dialog enables you to specify the target object and attribute for the edit box that you are editing. For information about this dialog, see Edit Target Object & Attribute Dialog .
Methods	Opens the Edit Methods dialog, in which you can create customized methods for the edit box. For information about how to create customized methods, see Methods, Actions, and Callbacks .

Select Edit Style Dialog

The Select Edit Style dialog enables you to select an existing edit style to apply to an edit box. An edit style controls the appearance and behavior of the editor when you edit edit boxes to which the edit style is applied.

The Select Edit Style dialog also enables you to access the Edit Field Edit Style dialog, in which you can create a new edit style for an edit box.

To open the Select Edit Style dialog, click the Select Edit Style button in the Edit Edit Box dialog. The Select Edit Style dialog looks like this:



To select an edit style, select the name of the style in the scroll area of the dialog, and click OK.

To ensure that all existing edit styles are displayed in the list, click Update.

To create an edit style, click the New button. Clicking New opens the Edit Field Edit Style dialog. See the following section for information about this dialog.

You can also create an edit style by clicking on the Field Edit Style icon in the More Options palette and dropping it on the workspace where you to use the edit style. You can access this palette by clicking on the More Options button in the GUIDE palette.

Creating and Editing an Edit Field Edit Style

You can create an edit style in either of three ways:

- Select the following choice from the GUIDE menu bar:
Item > GUIDE Objects > Edit Style

The icon for the edit style becomes attached to your cursor, and you can drop it on any workspace.

- Open the Edit Field Edit Style dialog by clicking the New button in the Select Edit Style dialog. When you click the New button, an edit style is created and

the following icon is added to the workspace that contains the object that you are editing.



- To edit an existing edit style or create a new edit style, select **edit field edit style** from the edit button menu to open the Edit Field Edit Style dialog. The Edit Field Edit Style dialog looks like this:

The dialog box is titled "Edit Field Edit Style" and contains the following fields and controls:

- Name:** A text field containing "NONE".
- Attribute Settings:** A group of four checkboxes:
 - Edit in multi line mode
 - Clear before edit
 - Edit in place
 - Show edit buttons
 - Confirm cancelled edit
- Font size:** A group of four radio buttons:
 - Small
 - Medium
 - Large
 - Default
- Editor size:** A group of two radio buttons:
 - DEFAULT
 - Minimum
 - Maximum
- Grammar:** A group of controls:
 - Show grammar menus
 - Class:** A text field containing "false".
 - Defining class:** A text field containing "false".
 - Attribute:** A text field containing "false".
- Frame style name:** A text field containing "DEFAULT".
- Field edit background color:** A text field containing "DEFAULT" and a color selection button (three dots).
- Field edit text color:** A text field containing "DEFAULT" and a color selection button (three dots).
- Language:** A list box containing "English", "French", "German", and "Italian".

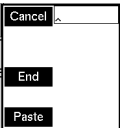

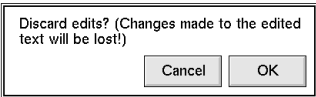
At the bottom of the dialog are three buttons: "OK", "Apply", and "Cancel".

The following table describes the components of the Edit Field Edit Style dialog:

Components of Edit Field Edit Style Dialog

Component	Description
Name	(required) Displays the current name of the edit style that you are editing. Changing the displayed value updates the name of the edit style. This field must contain a valid symbolic name entry.
Attribute Settings	
Edit in multi line mode	<p>Indicates whether the editor has scroll bars.</p> <p>When this button is selected, pressing Return adds a line-feed to the field, and the editor has scroll bars. Pressing Ctrl + Return or Tab closes edit.</p> <p>When this button is not selected, pressing Return terminates the editing session on this edit box. The editor does not have scroll bars.</p>
Clear before edit	Removes contents of edit box when a user starts to edit the edit box.
Edit in place	<p>Indicates where the editor appears when a user edits the field.</p> <p>When this button is selected, the editor appears directly on top of the edit box being edited.</p> <p>When this button is not selected, the editor appears in the upper left corner of the screen.</p>

Components of Edit Field Edit Style Dialog

Component	Description
Show edit buttons	<p>Indicates whether to show or hide the Cancel, End, and Paste buttons that appear along the left side of the editor in the field that you are editing.</p> <p>When this button is selected, the Cancel, End, and Paste buttons appear. For example:</p> <div style="text-align: center;">  </div> <p>When this button is not selected, the Cancel, End, and Paste buttons do not appear. For example:</p> <div style="text-align: center;">  </div>
Confirm cancelled edit	<p>Indicates whether or not to display a confirmation dialog when the user cancels an editing session on this field.</p> <p>When this button is selected, the following confirmation dialog appears when a user cancels an edit:</p> <div style="text-align: center;">  </div> <p>When this button is not selected, no confirmation dialog is displayed when a user cancels an editing session.</p>
Font size	<p>Specifies the size of the characters in the field. Select the button for the size that you want (small, medium, large, or default). The default size is the current font size of the object that you are editing.</p>
Editor size	<p>Specifies the size of the editor. You can specify the size of the editor in the field directly below the Editor size label. Or, you can determine editor size by selecting one of the following buttons:</p>

Components of Edit Field Edit Style Dialog

Component	Description
Minimum	The editor expands to size of the edit box that you are editing.
Maximum	The editor expands to display the entire contents of the edit box that you are editing. If the edit box has a multi-line format and contains several lines of text, the editor may expand to overlap other objects in the dialog.
Grammar	
Show grammar menus	Indicates whether or not to display look-ahead grammatical prompts in the editor. When this button is selected, the editor displays grammatical prompts as you edit. When this button is not selected, no grammatical prompts are displayed.
Class	The class of object for which grammar is enforced.
Defining class	The class that defines the attributes for which grammar is enforced. In most cases, this is the class specified in the preceding Class field. However, it can also be a parent class of the class specified in the preceding Class field.
Attribute	The attribute of the specified Class or Defining class for which grammar is enforced.
Frame style name	Displays the current frame-style (a built-in G2 4.0 class) to use when posting the editor. The frame style controls the appearance of the border that gets drawn around the edit field. This field expects a symbol that names a frame-style. The symbol default is also acceptable.

Components of Edit Field Edit Style Dialog

Component	Description
Field edit background color	<p>Displays the G2 color used for the background color of the editor displayed when a user edits the edit box.</p> <p>You can enter the name of any G2 color. You can also click the push button to the right of the field to display the Background Color dialog. In the Background Color dialog, you select one color and click OK.</p> <p>The symbol default is also acceptable.</p>
Field edit text color	<p>Displays the G2 color used for the text color of the editor displayed when a user edits the edit box.</p> <p>You can enter the name of any G2 color. You can also click the push button to the right of the field to display the Text Color dialog. In the Text Color dialog, you select one color and click OK.</p> <p>The symbol default is also acceptable.</p>
Language	<p>Enter the name of the natural language in which you want the field editor to be displayed.</p> <p>You can also select a language from the scroll area below this field. The name of the language that you select is displayed in the field above the scroll area.</p>

Specifying a Password-Style Block Font

You can specify a password-style block font for an edit box by applying to the edit box an edit style whose `uil-field-edit-use-block-font` attribute is set to `true`. When the block font is applied to the edit box, all characters that the user enters in the edit box appear as black rectangles. By default, the `uil-field-edit-use-block-font` attribute of edit styles is `false`.

To specify a password-style block font::

- 1 Open the attribute table of the edit style by selecting **table** from the menu of the edit style.
- 2 In the table, set the `uil-field-edit-use-block-font` attribute to `true`.

Background Color and Text Color Dialogs

The Background Color dialog enables you to select a background color for the edit boxes to which a particular edit style is applied. The Text Color dialog enables you to select the color of the text in these edit boxes.

To open the Background Color dialog, click the Field edit background color button in the Edit Field Edit Style dialog.

To open the Text Color dialog, click the Field edit text color button in the same dialog.

These dialogs are identical except for their titles. The Text Color dialog looks like this:

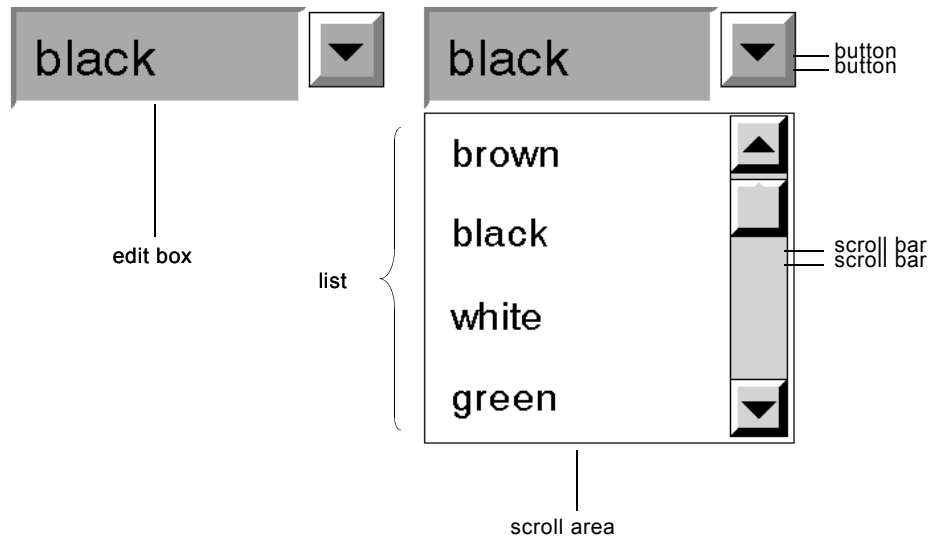


To choose a background color or text color, select the color in the scroll area of the dialog and click OK.

Combo Boxes

A Combo Box (uil-combo-box) is an edit box combined with a scrollable list of text items. Users can display the list by clicking the button in the upper right corner of the Combo Box. When a user selects an item in the list, the list is closed and the selected item appears in the edit box.

The following figure illustrates a Combo Box with its list hidden and shown:



Users can change the contents of the edit box part of a Combo Box in two ways:

- By clicking an item in the scrollable list. This causes the selected item to appear in the edit box, replacing the previous contents.

If you do not want users to be able to select messages in the scrollable list, specify `uil-combo-box-no-selection` as the method referenced from the `uil-message-unselection-method` attribute of the Combo Box.

- By entering characters or deleting them in the edit box directly.

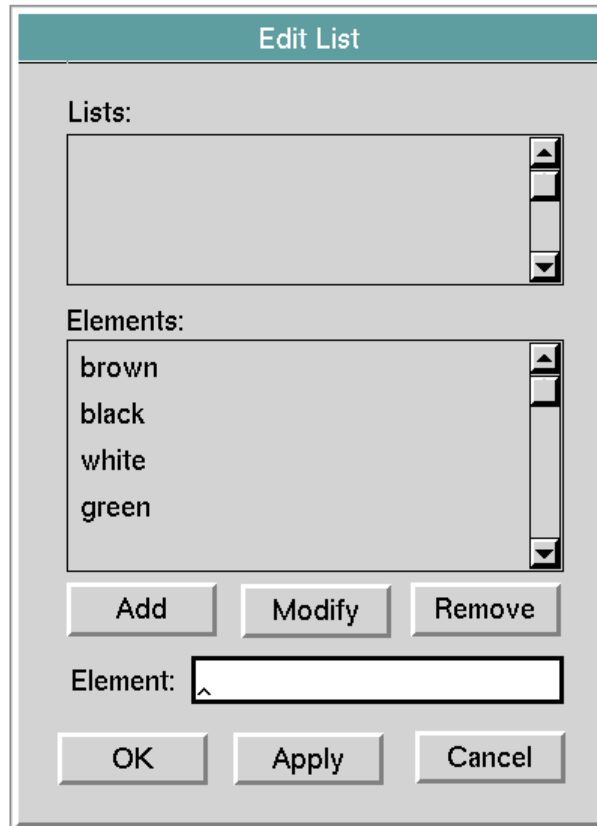
When a `conclude` method is run on a Combo Box, the value that is concluded is the value that currently appears in the edit box.

You cannot add a value to the Combo Box scrollable list by concluding a value to the Combo Box. You can, however, add a value to the Combo Box list programmatically. To do so, use the UIL procedure `uil-add-to-combo-box-list()` for individual values or the UIL procedure `uil-update-scroll-area-from-list()` for multiple values. For information about these procedures, see the *G2 GUIDE/UIL Procedures Reference Manual*.

Editing Combo Boxes

You can edit most attributes of a combo box through the Edit Edit Box dialog, which you can open by selecting `edit edit box` from the menu of the Combo Box.

You can edit the contents of the list in a Combo Box through the GUIDE/UIE Edit List dialog. To open the Edit List dialog, select edit combo box from the menu of Combo Box:



To add an item to the Combo Box list, enter the new item in the Elements field and click Add.

To change an item in the list, select the item in the Elements scroll area and edit it in the Element: field. Click Modify to apply your edit to that item.

To remove an item from the list, select the item and click Remove.

Note The Lists scroll area in the Edit List dialog is not used for editing Combo Boxes.

GUIDE/UIE provides procedures that manipulate and modify the default behavior of Combo Boxes. For information about these procedures, see the *G2 GUIDE/UIE Procedures Reference Manual*.

Spin Control Boxes

A spin control entry box (`uil-spin-control-entry-box`) is a specialized edit box that enables users to select one value from a range of values by scrolling a list of values up or down within the box:



Spin control boxes share most characteristics of other styles of edit boxes, including how their values can be updated and concluded.

Creating Spin Control Boxes

You can create spin control boxes in the following ways:

- Drag the icon for spin control boxes on the GUIDE palette to the workspace or dialog subworkspace where you want to add the spin control box.
- Select the following choice from the GUIDE menu bar:

Item > GUIDE Objects > Texts > uil-edit-box

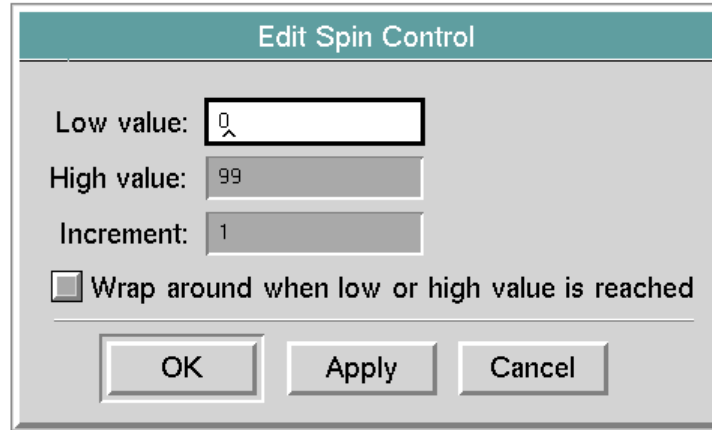
Under `uil-edit-box`, choose the size of spin control box that you want to add. The spin control box icon becomes attached to your cursor, and you can drop it wherever you want to add it.

- Add the spin control box programmatically, using the UIL procedure `uil-create-spin-control`. For information about this procedure, see *G2 GUIDE/UIL Procedures Reference Manual*.

Editing Spin Control Boxes

You can edit most attributes of a spin control box through the Edit Edit Box dialog, which you can open by selecting `edit edit box` from the menu of the Spin Control box.

You can edit the attributes of a spin control box that distinguish it from other edit boxes through the Edit Spin Control dialog. To open the Edit Spin Control dialog, select edit spin control from the menu of Spin Control Box:



The following table describes the components of the Edit Spin Control dialog:

Components of Edit Spin Control Dialog

Component	Description
Low Value	The lowest value that the spin control box can have. You can specify any valid G2 quantity.
High Value	The highest value that the spin control box can have. You can specify any valid G2 quantity.
Increment	The increment by which the value of the spin control box is increased or decreased when a user scrolls the spin control box. You can specify any valid G2 quantity.
Wrap around when low or high value is reached	<p>If the option is selected, the value of the spin control box wraps around when a user attempts to scroll beyond the low value or high value specified for the spin control box. For example, if a user is decrementing the value and scrolls beyond the low value, the first value scrolled to is the high value.</p> <p>If this option is not selected, scrolling stops when a user scrolls to either the low value or the high value.</p>

Summary of Edit Box, Combo Box, and Spin Control Menu Choices

Menu Choice	Description
table	Shows the attribute table of the edit box.
transfer	Places the edit box on the mouse and allows the user to move it to another workspace or subworkspace.
change min size	Enables you to change the minimum size of the edit box by dragging the edge of the edit box.
align text	Align edit box text left, right, or center.
color	Opens a series of G2 menus allowing icon regions to have their colors altered. GUIDE provides the Configuration Editor as a preferred means of changing colors of icon regions.
lift to top	Displays the edit box on top of another object on a workspace.
drop to bottom	Drops the edit box behind another object on a workspace.
describe	Shows a description of the object and its relations.
table of hidden attributes	Display the edit box's hidden attributes table.
show unsaved attributes	Display the table for the edit box with permanently changed attributes highlighted.
describe configuration	Display the inheritance of configurations for the edit box.
help on this item	Access help about edit boxes.
describe configuration	Displays a description of the edit box's current configuration.
edit configuration	Opens the Configuration Editor, which you can use to select, delete, copy, or edit configurations.

Menu Choice	Description
configure	Applies the current configuration to the edit box.
initialize	Calls the initialization method of the edit box. This menu choice is visible only when the edit box is enabled.
disable./enable.	Disabling the edit box prevents the select menu choice from being shown or run on the edit box. The colors of the edit box change to reflect its disabled state.
edit.	Opens a text editor for editing the message-contents of the edit box.
move	Opens the Move Object dialog, which enables you to position the edit box precisely.
clone.	Makes a copy of the edit box and places it next to the original edit box on the workspace.
delete.	Opens a confirmation dialog to the user and if confirmed, deletes the edit box.
edit edit box	Opens the Edit Edit Box dialog, in which you can edit attributes of the edit box.
validate	Calls the validation method for the edit box.

Summary of Spin Control Box Menu Choices

Menu Choice	Description
table	Shows the attribute table of the spin control box.
transfer	Places the spin control box on the mouse and allows the user to move it to another workspace or subworkspace.
change min size	Enables you to change the minimum size of the spin control box by dragging the edge of the spin control box.
align text	Align spin control box text left, right, or center.

Menu Choice	Description
color	Opens a series of G2 menus allowing icon regions to have their colors altered. GUI DE provides the Configuration Editor as a preferred means of changing colors of icon regions.
lift to top	Displays the spin control box on top of another object on a workspace.
drop to bottom	Drops the spin control box behind another object on a workspace.
describe	Shows a description of the object and its relations.
table of hidden attributes	Display the spin control box's hidden attributes table.
show unsaved attributes	Display the table for the spin control box with permanently changed attributes highlighted.
describe configuration	Display the inheritance of configurations for the spin control box.
help on this item	Access help about spin control boxes.
describe configuration	Displays a description of the spin control box's current configuration.
edit configuration	Opens the Configuration Editor, which you can use to select, delete, copy, or edit configurations.
configure	Applies the current configuration to the spin control box.
initialize	Calls the initialization method of the spin control box. This menu choice is visible only when the spin control box is enabled.
disable./enable.	Disabling the spin control box prevents the select menu choice from being shown or run on the spin control box. The colors of the spin control box change to reflect its disabled state.
edit.	Opens a text editor for editing the message-contents of the spin control box.

Summary of Combo Box Menu Choices

Menu Choice	Description
table	Shows the attribute table of the combo box.
transfer	Places the combo box on the mouse and allows the user to move it to another workspace or subworkspace.
change min size	Enables you to change the minimum size of the combo box by dragging the edge of the combo box.
align text	Align combo box text left, right, or center.
color	Opens a series of G2 menus allowing icon regions to have their colors altered. GUIDE provides the Configuration Editor as a preferred means of changing colors of icon regions.
lift to top	Displays the combo box on top of another object on a workspace.
drop to bottom	Drops the combo box behind another object on a workspace.
describe	Shows a description of the object and its relations.
table of hidden attributes	Display the combo box's hidden attributes table.
show unsaved attributes	Display the table for the combo box with permanently changed attributes highlighted.
describe configuration	Display the inheritance of configurations for the combo box.
help on this item	Access help about combo boxes.
describe configuration	Displays a description of the combo box's current configuration.
edit configuration	Opens the Configuration Editor, which you can use to select, delete, copy, or edit configurations.
configure	Applies the current configuration to the combo box.

Menu Choice	Description
initialize	Calls the initialization method of the combo box. This menu choice is visible only when the combo box is enabled.
disable./enable.	Disabling the combo box prevents the select menu choice from being shown or run on the combo box. The colors of the combo box change to reflect its disabled state.
edit.	Opens a text editor for editing the message-contents of the combo box.

Scroll Areas and Message Objects

Describes how to create and edit scroll areas and message objects.

Introduction **257**

Editing Scroll Areas **264**

Edit Message Dialog **273**

Multiple Column Scroll Areas **276**

Summary of Scroll Area Menu Choices **288**

Summary of Message Object Menu Choices **289**

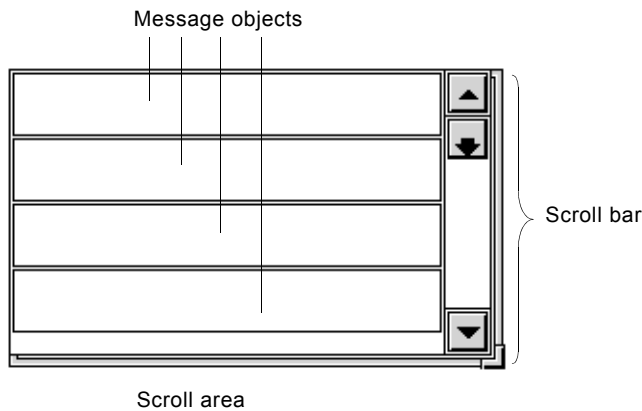


Introduction

A scroll area contains one or more message objects. Each message object contains a text value that can be updated from or concluded to other objects in your G2 application.

You can use a scroll area to represent a set of values that is larger than can be conveniently displayed in its entirety at one time. Users can scroll the contents of the scroll area up or down by clicking on the scroll bar of the scroll area.

The following figure illustrates a scroll area containing several message objects:



Note In order to create scroll areas, GUIDE requires that the top-level module of your application be named. GUIDE incorporates the name of the top-level module into scroll area names that it generates for internal use.

Adding Scroll Areas and Message Objects

The GUIDE palette provides two icons for scroll areas. One contains empty message objects by default, and one contains no message objects.

The empty scroll area is convenient to use if you intend to add all message objects to the scroll area programmatically, using the procedure `uil-add-message-to-list` or `uil-create-message`. For information about these procedures, see *G2 GUIDE/UII. Procedures Reference Manual*.

To add a scroll area to a workspace or dialog, select the icon for the scroll area that you want to use and drop it on the workspace or dialog subworkspace. You can add individual message objects to existing scroll areas, but you cannot add them to a dialog or workspace outside of any scroll area.

To add a message object to an existing scroll area, select the icon in the GUIDE palette labeled Message and drop it on or near the scroll area. The message object is added to the scroll area.

You can also add scroll areas and message objects by selecting the following choices from the GUIDE menu bar:

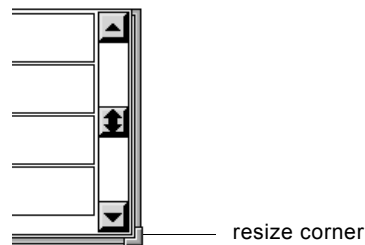
- Item > GUIDE Objects > Scroll Areas
- Item > GUIDE Objects > Texts > uil-message-object

Under the choice for scroll areas, you can choose either empty or populated scroll areas. The object that you select becomes attached to your cursor. You can drop it

on the subworkspace or the master dialog to which you want to add the scroll area or message object.

Resizing Scroll Areas and Message Objects

To resize a scroll area, move the pointer to the resize corner, which is the lower right corner of the scroll area. Then drag until the scroll area has the size and shape that you want. The following figure illustrates a scroll area and its resizing corner.



When you resize a scroll area, the message objects in the scroll area are automatically resized to the new size of the scroll area.

Note It is good practice to disable the resize corner when you finish adjusting the size of the scroll area, so that it does not appear at run time. For information about how to do this, see [Scroll Area Options Dialog](#).

Moving Scroll Areas and Message Objects

You can move a scroll area by moving the pointer to an edge of the scroll area and dragging it.

You can rearrange the message objects in a scroll area by dragging individual message objects up or down within the scroll area.

You can move message objects from one scroll area to another by dragging the message objects. When you drop a message object on the new scroll area, the message object is resized and reformatted according to the size and format of the new scroll area.

Updating and Concluding Scroll Areas

Scroll areas can be updated from source objects and can conclude their values into target objects. You can edit scroll areas to specify how their contents are updated and concluded.

For example, you can specify that the contents of all the message objects in a scroll area are cleared when the scroll area is updated with a list of values, or that message objects are selected if their values match any of the update values.

Similarly, you can specify that only selected message objects conclude their values when the scroll area is concluded, or that all message objects conclude their values.

For information about how to specify update and conclude options for scroll areas, see [Managing Message Size in Scroll Areas](#).

Specifying Formats for Message Objects

You can specify formats for individual message objects. Each format contains formatting information about the appearance of the text contained in the message object. The format also controls how quotation marks are displayed and how cases are handled. For information about how to select a format for a message object, see [Select Format Dialog](#).

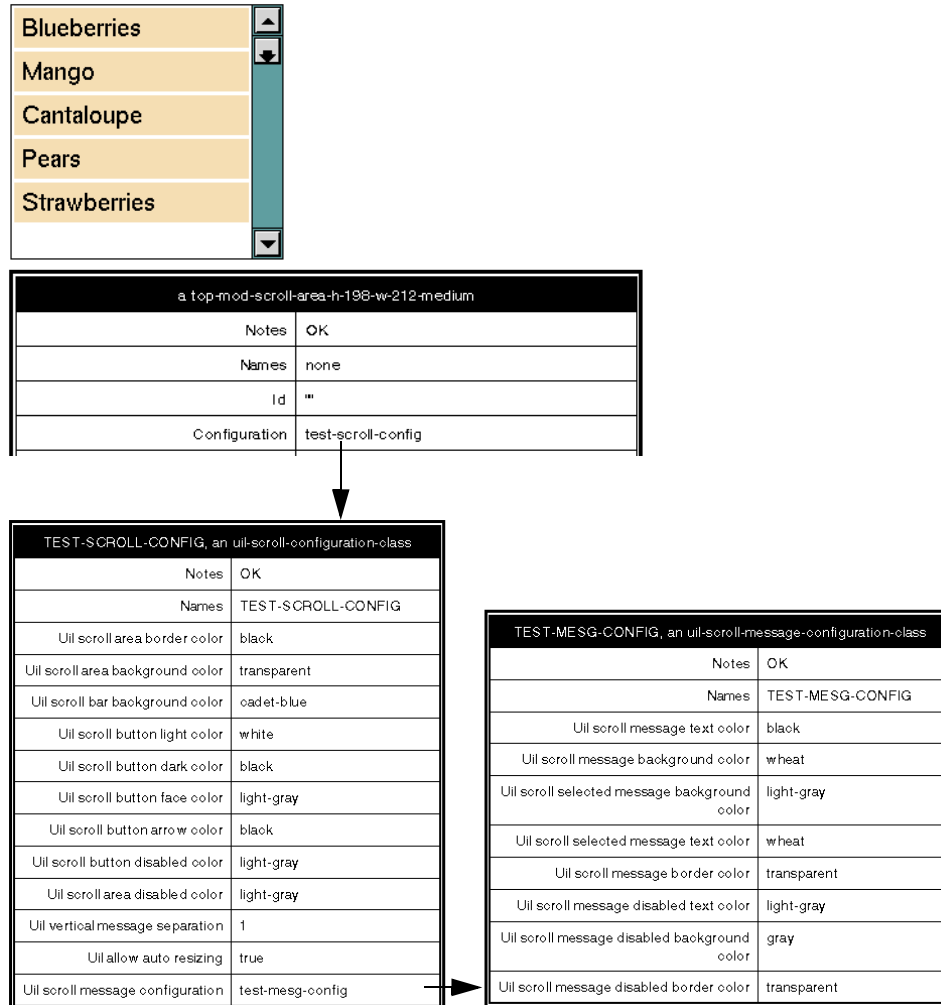
Specifying Configurations for Scroll Areas and Message Objects

Scroll areas and message object have separate configurations.

The configuration of a scroll area specifies the colors of foreground, background, scroll bar, buttons, and arrows on the scroll area. The configuration of the scroll area also has an attribute, `uil-scroll-message-configuration`, that specifies a default configuration for message objects in the scroll area.

If you add a message object whose own configuration is unspecified to a scroll area, the default message object configuration of the scroll area determines the colors of that message object. However, if you add a message object that specifies its own configuration, the message object's own configuration determines the colors of the message object.

The following figure illustrates a scroll area, the table of the scroll area's configuration, and the table of the default message object configuration referenced by the scroll area:



In the figure above, the configuration attribute of the scroll area references a scroll area configuration named `test-scroll-config`. This scroll area configuration references a default message object configuration named `test-mesg-config`. If you add a message object whose own configuration is unspecified to this scroll area, the colors of this message object are specified by `test-mesg-config` when you add it to the scroll area. If the message object specifies its own configuration, `test-mesg-config` has no effect on the colors of the message object.

Specifying Selection and Unselection Methods for Scroll Areas

Message objects can have their own selection and unselection methods. For information about how to specify methods for UIL controls, see [Methods, Actions, and Callbacks](#).

Managing Message Size in Scroll Areas

Scroll areas do not change size to accommodate long messages received by message objects at run time. Long messages can extend beyond the boundaries of the scroll area. You can do two things to prevent this from happening:

- Specify a maximum number of characters to display in message objects. If the message object receives a message that exceeds the maximum number of characters, the text is clipped to the maximum number and is annotated with ellipses (...).

You can specify a maximum number of characters to display on both the scroll area and on individual message objects. You set the Maximum number of characters to display option for scroll areas in the Scroll Area Options dialog, and for message objects in the Edit Message dialog. The setting of the Maximum number of characters to display option is stored in the attribute `uil-maximum-characters-to-display` attribute of scroll areas and message objects. The value of a message object's `uil-maximum-characters-to-display` attribute takes precedence over the value of a scroll area's `uil-maximum-characters-to-display` attribute.

- Allow the message objects in a particular scroll area to contain multiple line messages. When this option is selected, long messages in message objects that do not clip text can wrap to additional lines in the message object.

To allow multiple line messages in the message objects in a particular scroll area, select the Multi-lined messages option in the Scroll Area Options dialog. The setting of this option is stored in the `uil-allow-multiple-line` attribute of the scroll area. You cannot set the multiple line option for individual message objects.

For information about how to set the options for managing long messages, see [Scroll Area Options Dialog](#).

Specifying User-Defined Methods for Message Objects

When you add a message object to a scroll area, the message object inherits the methods that are specified as attributes of the scroll area, as well as values for other attributes such as `uil-maximum-characters-to-display` and configuration.

If you want to specify user-defined methods as attributes of the message objects in a scroll area, it is easiest to specify the user-defined methods as attributes of the scroll area *before* you add the message objects to the scroll area. The message objects that you subsequently add to the scroll area inherit the user-defined methods.

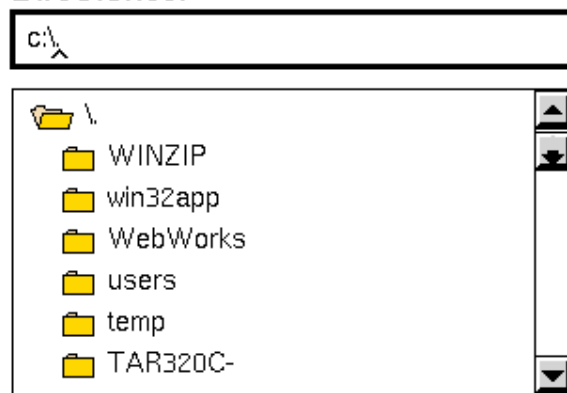
If you specify user-defined methods as attributes of a scroll area *after* you add message objects to it, the message objects do not inherit the new user-defined methods. In this case, you must edit the attributes of each message object individually to specify the user-defined methods.

Appending Items to Message Objects

To illustrate the purpose or content of message objects, you can append items to the message objects. The icons of the appended items appear next to the message objects.

For example, the following scroll area in the G2 Save Module dialog contains message objects with appended `g2-ui-file-open-icon` or `g2-ui-file-closed-icon` objects appended to them. Objects of these classes have icons that illustrate open or closed directories:

Directories:



To create a message object with an appended item:

- 1 Create a subclass of the `uil-appended-item` class. You append an instance of this class to the message object.
- 2 As a method of your subclass, specify `uil-on-single-click` or `uil-on-double-click` to specify that single-clicking or double-clicking on the icon causes the associated message object to be selected. For information about these UIL procedures, see the *G2 GUIDE/UIL Procedures Reference Manual*.

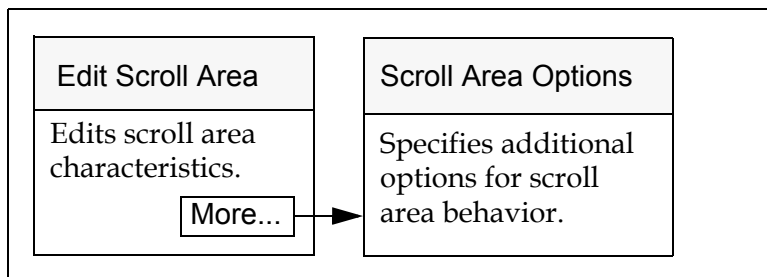
- 3 Assign an icon to the `uil-appended-item` class. This icon appears next to the message object to which the instance of your appended item subclass is appended.
- 4 Call the UIL procedure `uil-create-message-with-appended-item` to create a new message object with an instance of your appended item subclass appended to it, or `uil-append-item-to-message` to append an instance of your subclass to an existing message object.

For information about the UIL procedures that you can use to append items to message objects and manipulate appended items, see the *G2 GUIDE/UIL Procedures Reference Manual*.

Editing Scroll Areas

You can edit the appearance and behavior of scroll areas, using the Edit Scroll Area dialog and the Scroll Area Options dialog. The following figure illustrates these dialogs:

Dialogs for Editing Scroll Areas



Edit Scroll Area Dialog

The Edit Scroll Area dialog enables you to edit characteristics of a scroll area such as its name, ID, position, location, and other characteristics. To open the Edit Scroll Area dialog, click the scroll area that you want to edit and select **edit scroll area** from the scroll area menu. The Edit Scroll Area dialog looks like this:

The dialog box is titled "Edit Scroll Area" and contains the following fields and options:

- Class:** NONE
- Name:** UNSPECIFIED
- Id:** (empty field)
- Position:** X: 373, Y: 340
- Dimensions:** Width: 250, Height: 150
- State:**
 - Enabled
 - Disabled
- Ordering:**
 - Alphabetic
 - Chronological
 - Logbook
 - Priority
 - Unordered
- Font:**
 - Small
 - Medium
 - Large
- Update Style:**
 - Clear before update
 - Select matching items
 - Reuse messages
- Orientation:**
 - Scrollbar on left
 - Scrollbar on right
- Conclude Style:**
 - Selected messages
 - All messages
 - All including disabled
- Format:** unspecified

Buttons at the bottom: Source Object..., Target Object..., Methods..., More..., OK, Apply, Cancel.

The following table describes the components of the Edit Scroll Area dialog:

Components of Edit Scroll Area Dialog

Component	Description
Class	(read-only) Displays the class of the scroll area that you are editing.
Name	(optional) Displays the current name of the scroll area that you are editing. Changing the displayed value updates the name of the scroll area. This field does not require a value. Its contents, if any, must be a valid symbolic name.
Id	(optional) Displays the ID of the scroll area that you are editing. Changing the ID updates the id attribute of the scroll area.
Position	(required) The two edit box fields below the label Position display the current item-x-position and item-y-position of the scroll area that you are editing. Changes to the X and Y values will move the scroll area to the new location. This is a required field and defaults to the scroll area's current position on the workspace. The X and Y values must be integers.
Dimensions	(required) The two edit box fields below the label Dimensions display the current item-width and item-height of the scroll area that you are editing. You can resize the scroll area by changing the width and height values, while maintaining its X and Y position. The width and height values must be integers. This attribute defaults to the scroll area's current dimensions on the workspace.

Components of Edit Scroll Area Dialog

Component	Description
State	<p>(required) The radio buttons below the label State indicate the current state (enabled or disabled) of the object that you are editing.</p> <p>You can change the state by selecting the radio button whose label indicates the desired state. When this change is applied, the scroll area is updated to reflect its new state.</p>
Ordering	<p>(required) The radio buttons below the label Ordering indicate the current ordering method (alphabetic, chronological, logbook, priority, or unordered) of the scroll area that you are editing.</p> <p>You can change the ordering method of the scroll area by selecting the radio button whose label indicates the desired method. When this change is applied, GUIDE reorders the messages contained in the scroll area in the new order.</p>
Alphabetic	Sorts the messages in alphabetical order.
Chronological	The most recent messages are added to the bottom. Messages are time stamped with the current real-time when they are created.
Logbook	The most recent messages are added to the top.
Priority	Sorts the messages according to the value of the <code>message-display-priority</code> attribute of the message. You can set this attribute on individual message objects using the Edit Message dialog.
Unordered	Messages are displayed in the order that they were added to the scroll area.

Components of Edit Scroll Area Dialog

Component	Description
Font size	<p>The radio buttons below the label Font size indicate the current size font (small, medium, or large) used by the messages contained in the scroll area that you are editing.</p> <p>Users can change the size of the font used by selecting the radio button whose label indicates the desired size. When this change is applied, GUIDE regenerates the messages to display their new font size without changing the message objects' other attributes.</p>
Update Style	
Clear before update	Deleting any message objects that are currently in the scroll area before updating the scroll area.
Select matching items	Selects any message object in the scroll area whose value matches one of the values with which the scroll area is being updated.
Reuse messages	Recycles message objects in the scroll area. When an update method is run on the scroll area, existing messages objects are used to display the values with which the scroll area is being updated. If the scroll area contains more message objects than are needed to display the update values, the unused message objects are disabled and hidden.
Conclude Style	
Selected messages	Conclude values of all message objects that are currently selected and enabled.
All messages	Conclude values of all enabled message objects.
All including disabled	Conclude values of all message objects, including disabled message objects.

Components of Edit Scroll Area Dialog

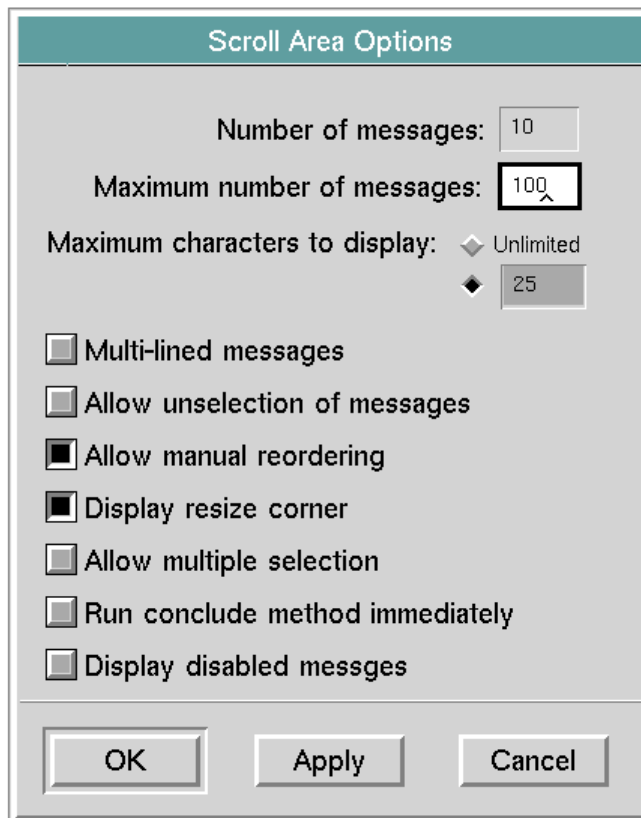
Component	Description
Orientation	<p>The Scrollbar on left and Scrollbar on right radio buttons below the label Orientation indicate the scroll bar's orientation on the scroll area that you are editing. Users can change the scroll bar's orientation by selecting the radio button whose label indicates the desired orientation. When this change is applied, GUIDE displays the scroll area with the new scroll bar orientation.</p>
Format	<p>Clicking the button to the right of the Format field opens the Select Format dialog, in which you can select a format to apply to message objects in the scroll area. For information about formats and the Select Format dialog, see Formats and Validation Criteria.</p> <p>Note: If you specify formats for individual message objects within the scroll area, these formats override the format that you select for the scroll area as a whole through the Edit Scroll Area dialog.</p>
Source Object	<p>Opens the Edit Source Object & Attribute dialog. The Edit Source Object & Attribute dialog enables you to specify the source object and attribute for the scroll area that you are editing. For information about this dialog, see Edit Source Object & Attribute Dialog.</p>
Target Object	<p>Opens the Edit Target Object & Attribute dialog. The Edit Target Object & Attribute dialog enables you to specify the target object and attribute for the scroll area that you are editing. For information about this dialog, see Edit Target Object & Attribute Dialog.</p>

Components of Edit Scroll Area Dialog

Component	Description
Methods	<p>Opens the Edit Methods dialog, in which you can create customized methods for the scroll area.</p> <p>For information about how to create customized methods, see Methods, Actions, and Callbacks.</p>
More	<p>Opens the Scroll Area Options dialog. The dialog contains additional choices for the scroll area that you are editing. See the following section for information about the Scroll Area Options Dialog.</p>

Scroll Area Options Dialog

To open the Scroll Area Options dialog, click the More button in the Edit Scroll Area dialog. The Scroll Area Options dialog looks like this:



The following table describes the components of the Scroll Area Options dialog:

Components of Scroll Area Options Dialog

Component	Description
Number of messages	(read only) Displays the total number of messages currently in the scroll area that you are editing.
Maximum number of messages	(required) Specifies the maximum number of messages allowed in the scroll area that you are editing. When the number of messages exceeds the maximum number, the excess messages are removed from the scroll area. Messages are removed in the reverse of the ordering method specified under Ordering in the Edit Scroll Area dialog. For example, if Alphabetic is the currently selected ordering method, messages are removed in reverse alphabetic order. If Logbook is the ordering method, messages are removed from the bottom of the list, because the Logbook ordering method adds message to the top.
Maximum number of characters to display	Displays the current number of characters displayed before clipping. The clipped text is placed in the text attribute. By default this attribute is 25 characters. You can use this feature to prevent message objects from extending beyond the edges of the scroll area.
Multi-lined messages	Allow multi-line messages in the scroll area. A long message wraps to additional lines.
Allow unselection of messages	Allow users to unselect messages by clicking on them.
Allow manual reordering	When this button is selected, application developers and users can change the order of the messages in the scroll area by dragging individual messages up or down.

Components of Scroll Area Options Dialog

Component	Description
Display resize corner	<p>Displays the resizing corner on the lower right corner of the scroll area. You can drag this handle to change the size and shape of the scroll area.</p> <p>You can disable the resize corner if you do not want end users to be able to change the size of the scroll area.</p>
Allow multiple selection	<p>Allow users to select more than one message in the scroll area.</p>
Run conclude method immediately	<p>If this option is selected, the contents of the scroll area conclude to the scroll area target object whenever there is a change to the contents of the scroll area. If this option is not selected, the contents of the scroll area are not concluded until a conclude method is run on the dialog that contains the scroll area.</p>
Display disabled messages	<p>If this option is selected, disabled message objects in the scroll area are greyed out but displayed. If this option is not selected, disabled message objects in the scroll area are invisible.</p>

Edit Message Dialog

To open the Edit Message dialog, click the message object that you want to edit and choose **edit message** from the message object's menu. The Edit Message dialog looks like this:

The following table describes the components of the Edit Message dialog:

Components of Edit Message Dialog

Component	Description
Class	(read-only) Displays the class of message object that you are editing.
Name	(optional) Displays the current name of the message object that you are editing. Changing the displayed value updates the name of the message object. This field does not require a value. Its contents, if any, must be a valid symbolic name.

Components of Edit Message Dialog

Component	Description
Id	<p>(optional) Displays the ID of the message object that you are editing. Changing the ID updates the id attribute of the message object.</p>
Message Value	<p>(optional) Displays the unformatted contents of the message object that you are editing. You can edit the Message Value field to modify the message object's unformatted contents. The unformatted contents are stored in the <code>message-contents</code> attribute of the message object.</p> <p>If you specify a format for the message object (through the Select Format dialog), this format is applied to the contents of the <code>message-contents</code> attribute, and the resulting formatted text is stored in the <code>text</code> attribute of the message object. Any limitation on the length of the message object that you specify in the Maximum characters to display field (see below) is also applied to the formatted text in the <code>text</code> attribute. You cannot directly access or edit the <code>text</code> attribute.</p>
Position	<p>(required) The two fields below the label Position display the current <code>item-x-position</code> and <code>item-y-position</code> of the message object that you are editing. Changes to the X and Y values will move the message object to the new location. This is a required field and defaults to the message object's current position on the workspace. The X and Y values must be integers.</p> <p>If you specify a position that is outside the scroll area that contains the message object, the message object appears at this position. To bring the message object back into the scroll area, you can move the scroll area.</p>
Dimensions	<p>(read-only) The two text fields below the label Dimensions display the current <code>item-width</code> and <code>item-height</code> of the message object that you are editing. You cannot edit this field.</p>

Components of Edit Message Dialog

Component	Description
State	(required) The Enabled and Disabled radio buttons below the label State indicate the current state of the message object that you are editing. Users can change the state by selecting the radio button whose label indicates the desired state. When this change is applied, the message object is updated to reflect its new state.
Options	
Message priority	<p>Displays an integer value for the current priority of the message object that you are editing. The value is stored in the <code>message-display-priority</code> attribute of the message object.</p> <p>Message objects in a scroll area are ordered by the values in their <code>message-display-priority</code> attributes if you specify Priority as the ordering method for the scroll area in the Edit Scroll Area dialog.</p>
Maximum characters to display	<p>Displays the current number of characters displayed before clipping. By default this attribute is the symbol <code>unlimited</code>.</p> <p>Setting the Maximum characters to display can prevent a message object from expanding beyond its original dimensions. This is particularly useful for maintaining dialog layout and item alignment. The full, unformatted text is maintained in the <code>message-contents</code> attribute of the message object. The clipped and/or formatted text is placed in the <code>text</code> attribute (which is not visible to the user).</p>

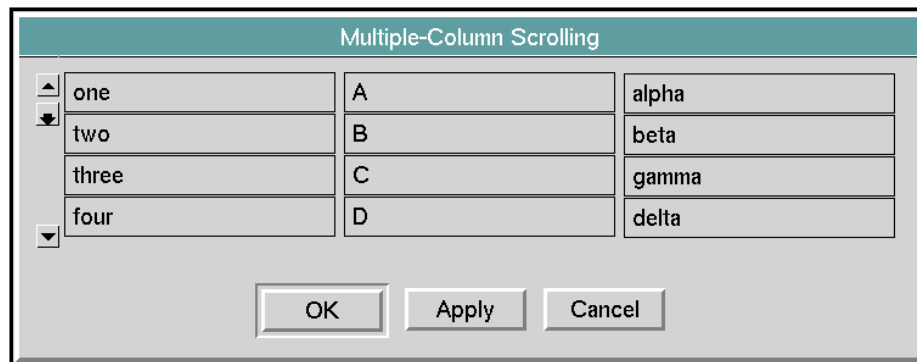
Components of Edit Message Dialog

Component	Description
Format	<p>You can enter the name of an existing format to specify a format for this message object. For information about how to use formats, see Formats and Validation Criteria.</p> <p>Clicking on the button to the right of the Format edit box opens the Select Format dialog, in which you can select a format for this message object. For more information about how to select a format, see Select Format Dialog.</p>
Source Object	<p>Opens the Edit Source Object & Attribute dialog. The Edit Source Object & Attribute dialog enables you to specify the source object and attribute for the message object that you are editing. For information about this dialog, see Edit Source Object & Attribute Dialog.</p>
Target Object	<p>Opens the Edit Target Object & Attribute dialog. The Edit Target Object & Attribute dialog enables you to specify the target object and attribute for the message object that you are editing. For information about this dialog, see Edit Target Object & Attribute Dialog.</p>
Methods	<p>Opens the Edit Methods dialog, in which you can create customized methods for the message object.</p> <p>For information about how to create customized methods, see Methods, Actions, and Callbacks.</p>

Multiple Column Scroll Areas

For some purposes, your application may need to display sets of related information in a multiple-column scroll area. Users need to scroll all the columns simultaneously.

Using GUIDE, you can construct multiple-column scroll areas such as the three-column scroll area in the following dialog:



In the dialog shown above, users can scroll all three columns simultaneously using the scroll bar on the left side of the left column. For example, if a user scrolls the left scroll area so that **two** is at the top, the other scroll areas are scrolled so that **B** and **beta** move to the top.

Note You can access a working example of a multiple-column scroll area through the GUIDE Examples workspace. To access this example, click **UIL Examples** in the GUIDE Help dialog, and go to the **Scroll Area Examples** workspace.

Creating a Multiple-Column Scroll Area

To create a multiple-column scroll area:

- 1 Add a scroll area to the dialog, using the GUIDE palette.
- 2 If you want the scroll bar for the multiple-column scroll area to be on the left side, move the scroll bar to the left side.

If you want the scroll bar for the multiple-column scroll area to be on the right side, you can leave the scroll bar in its default position (right side).

To move the scroll bar to the opposite side of the scroll area, choose **left right reflection.** from the scroll area's menu.

- 3 Make one copy of the scroll area to represent each additional column in the multiple-column scroll area.

To make a copy, select **clone.** from the user menu of the scroll area.

- 4 Move the scroll areas next to each other, and align them vertically.

The scroll bar for the multiple-column scroll area should be either on the left side or the right side of the contiguous scroll areas.

- 5 For your own convenience, edit the IDs of the scroll areas to identify them as scroll areas used together for multiple-column scrolling.

In this example, the three scroll areas have the IDs `my-sa-1`, `my-sa-2`, `my-sa-3`, from left (`my-sa-1`) to right.

To edit a scroll area, choose **edit scroll area** from its user menu. This opens the Edit Scroll Area dialog, in which you specify the ID for each scroll area.

Note Use the Edit Scroll Area dialog to edit the IDs of the scroll areas. Do not edit the IDs directly in the attribute tables.

- 6** Edit the configuration of each cloned scroll area to hide its scroll bar and border. To do this:
 - a** Select **edit configuration** from the user menu of the cloned scroll area. This opens the GUIDE Configuration Editor.
 - b** In the GUIDE Configuration Editor, select `uil-default-scroll-configuration`, and click the Copy button.
 - c** A dialog appears prompting you to enter a name for the new configuration. Enter a name such as `multiple-column-scrolling-config`, and click OK.

This creates a copy of the standard configuration `uil-default-scroll-configuration`. You can now edit this configuration to make it hide the scroll bar and borders.
 - d** Edit the new configuration to make the scroll bar and borders transparent.

To do this, select the name of the new configuration in the GUIDE Configuration Editor. Then click the Edit button. This opens another dialog named GUIDE Configuration Editor.
 - e** In the second GUIDE Configuration Editor dialog, set the colors of the following attributes to **transparent**:
 - `uil-scroll-area-border-color`
 - `uil-scroll-area-background-color`
 - `uil-scroll-bar-background-color`
 - `uil-scroll-button-light-color`
 - `uil-scroll-button-dark-color`
 - `uil-scroll-button-face-color`
 - `uil-scroll-button-arrow-color`

To set an attribute to **transparent**, click it. This opens a dialog in which you can select a color for that particular attribute. Select **transparent** and click OK.

When you have set all these attribute to **transparent**, click OK in the second GUIDE Configuration Editor.

- f** In the first GUIDE Configuration Editor, make sure that the new configuration is selected – in this example, **multiple-column-scrolling-config** – and click OK.

This applies the new configuration to the cloned scroll area. The scroll bar and borders on the cloned scroll area disappear.

- 7** Write customized methods for the scroll area whose scroll bar you intend to use to scroll all the columns.

These methods perform all scrolling and message selection and unselection operations on the multiple-column scroll areas. You can write methods to reference from the following attributes of the scroll area:

```
uil-message-selection-method
uil-message-unselection-method
uil-scroll-increment-method
uil-scroll-increment-line-method
uil-scroll-decrement-method
uil-scroll-decrement-line-method
uil-scroll-to-position-method
```

For example, you can create methods as shown in the following scroll area attribute table:

Uil message selection method	my-multi-scroll-message-selection
Uil message unselection method	my-multi-scroll-message-unselection
Uil scroll increment method	my-multi-scroll-increment
Uil scroll decrement method	my-multi-scroll-decrement
Uil scroll increment line method	my-multi-scroll-increment-line
Uil scroll decrement line method	my-multi-scroll-decrement-line
Uil scroll to position method	my-multi-scroll-to-position

- 8** For the scroll areas with hidden scroll bars and borders, write empty placeholder methods for all scrolling, message selection, and message unselection operations.

Reference these empty methods from the appropriate attributes of the scroll areas with hidden scroll bars. Using these empty methods guarantees that nothing happens if a user accidentally scrolls with a hidden scroll bar.

For example, you can create empty methods as shown in the following scroll area attribute table:

Uil message selection method	my-multi-scroll-message-selection-noop
Uil message unselection method	my-multi-scroll-message-unselection-noop
Uil scroll increment method	my-multi-scroll-increment-noop
Uil scroll decrement method	my-multi-scroll-decrement-noop
Uil scroll increment line method	my-multi-scroll-increment-line-noop
Uil scroll decrement line method	my-multi-scroll-decrement-line-noop
Uil scroll to position method	my-multi-scroll-to-position-noop

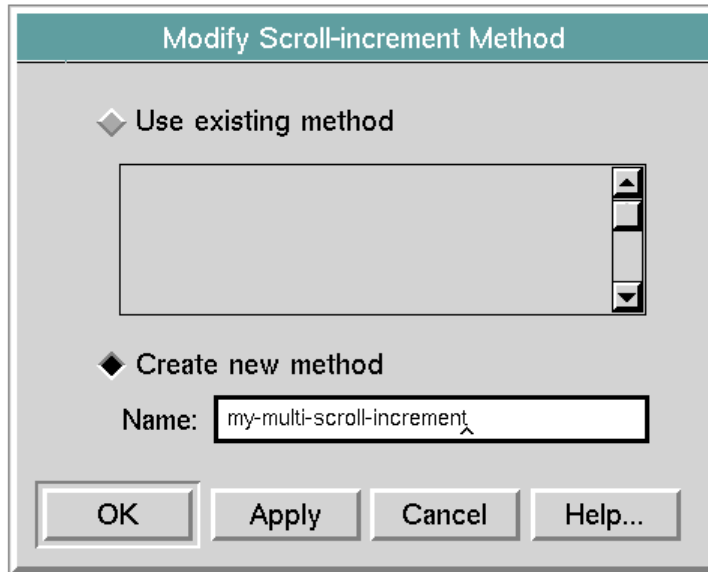
Creating Methods Required by Multiple-Column Scroll Areas

To create the methods required for a multiple-column scrolling area:

- 1 Select **edit scroll area** from the menu of the left scroll area to open the Edit Scroll Area dialog.
- 2 In the Edit Scroll Area dialog, click the **Methods** button to open the Edit Methods dialog for scroll areas:

- 3 Click the button for the kind of method that you want create to open a dialog in which you can specify a name for the method.

For example, if you want to create a scroll increment method, click that button. The Modify Scroll-increment Method dialog appears:



In the Modify Scroll-increment Method dialog, select the Create new method option and enter the name of the new method in the Name field. When you click OK, the new method is created.

- 4 Open the attribute table of the new method.

The table contains the default definition of the method:

MY-MULTI-SCROLL-INCREMENT, a procedure	
Notes	OK
Authors	none
Tracing and breakpoints	default
Class of procedure invocation	none
Default procedure priority	6
Uninterrupted procedure execution limit	use default
<pre>my-multi-scroll-increment (SA: class ui-scroll-area) begin {- The ui-scroll-increment-method is called whenever the user clicks on the down-arrow of a scroll-bar. SA is the scroll area whose up-arrow was clicked on No value is returned by this method. -} end</pre>	

- 5 Edit the body of the new method to make it perform the operations required for multiple-column scrolling.

The following sections illustrate the methods that you must create for multiple-column scrolling.

Caution Do not edit the arguments of the new method. Each method referenced by the attribute of an object has a set of arguments to which GUIDE passes values automatically when the method is invoked. The method cannot function properly if you alter these arguments.

Scroll Increment Method

The scroll increment method is called whenever a user clicks on the scroll bar below the scroll thumb. The scroll increment method is referenced by the `uil-scroll-increment-method` attribute of the scroll area.

The scroll increment method has the following required argument signature:

```
guide-sample-scroll-increment(scroll-area: class uil-scroll-area)
```

For example, the following customized scroll increment method, `my-multi-scroll-increment`, is referenced from the `uil-scroll-increment-method` attribute of the scroll area used to scroll the entire multi-column scroll area:

```
my-multi-scroll-increment (SA: class uil-scroll-area)
SA-2: class uil-scroll-area;
SA-3: class uil-scroll-area;
WS: class kb-workspace;
begin
  call uil-scroll-increment (SA);

  WS = the workspace of SA;
  SA-2 = call uil-get-grobj-from-id-on-dlg-or-wksp ("my-sa-2", WS);
  SA-3 = call uil-get-grobj-from-id-on-dlg-or-wksp ("my-sa-3", WS);

  call uil-display-message-list (SA-2, the current-message-pointer of SA);
  call uil-display-message-list (SA-3, the current-message-pointer of SA);
end
```

The method `my-multi-scroll-increment` does the following things:

- It scrolls the scroll area upward when a user clicks on the down arrow on the scroll bar. To do this, the method calls the UIL procedure `uil-scroll-increment`.
- It scrolls two other scroll areas in the multi-column scroll area. The IDs of these scroll areas are `my-sa-2` and `my-sa-3`. To scroll `my-sa-2` and `my-sa-3`, the method calls the UIL procedure `uil-display-message-list` to set the current message pointers of `my-sa-2` and `my-sa-3` to be equal to the current message pointer of the first scroll area. The result is that the three scroll areas are

scrolled simultaneously and to the same position when a user clicks on the down arrow.

Note The `current-message-pointer` referenced by `uil-display-message-list` points to the top message currently displayed in the scroll area.

Scroll Decrement Method

The scroll decrement method is called whenever a user clicks on the scroll bar above the scroll thumb. The scroll decrement method is referenced by the `uil-scroll-decrement-method` attribute of the scroll area.

For example, the following customized scroll decrement method, `my-multi-scroll-decrement`, is referenced from the `uil-scroll-decrement-method` attribute of the scroll area used to scroll the entire multi-column scroll area:

```
my-multi-scroll-decrement (SA: class uil-scroll-area)
SA-2: class uil-scroll-area;
SA-3: class uil-scroll-area;
WS: class kb-workspace;
begin
  call uil-scroll-decrement (SA);

  WS = the workspace of SA;
  SA-2 = call uil-get-grobj-from-id-on-dlg-or-wksp ("my-sa-2", WS);
  SA-3 = call uil-get-grobj-from-id-on-dlg-or-wksp ("my-sa-3", WS);

  call uil-display-message-list (SA-2, the current-message-pointer of SA);
  call uil-display-message-list (SA-3, the current-message-pointer of SA);
end
```

The `multi-column-scroll-decrement` method is identical to the customized scroll increment method, except that it calls `uil-scroll-decrement` rather than `uil-scroll-increment`.

Scroll Decrement Line Method and Scroll Increment Line Method

The scroll decrement line method is called whenever a user clicks on the scroll up arrow on the scroll bar. This method causes the message objects in the scroll area to move down one line.

The scroll increment line method is called whenever a user clicks on the scroll down arrow on the scroll bar. This method causes the message objects in the scroll area to move up one line.

The contents of these methods can be the same as the contents of the scroll decrement method and scroll increment method, except that they call `uil-scroll-decrement-line` (rather than `uil-scroll-decrement`) and `uil-scroll-increment-line` (rather than `uil-scroll-increment`).

Scroll to Position Method for Multiple-Column Scrolling

The scroll to position method is called whenever a user drags the scroll thumb. The scroll to position method is referenced by the `uil-scroll-to-position-method` attribute of the scroll area.

The scroll to position method has the following required argument signature:

```
guide-sample-scroll-to-position(scroll-area: class uil-scroll-area)
```

For example, the following customized scroll to position method, `my-multi-scroll-to-position`, is referenced from the `uil-scroll-to-position-method` attribute of the scroll area used to scroll the entire multi-column scroll area:

```
my-multi-scroll-to-position (SS: class uil-scroll-segment)
SA-1: class uil-scroll-area;
SA-2: class uil-scroll-area;
SA-3: class uil-scroll-area;
WS: class kb-workspace;
begin
  call uil-scroll-to-position (SS);

  WS = the workspace of SS;
  SA-1 = call uil-get-grobject-from-id-on-dlg-or-wksp ("my-sa-1", WS);
  SA-2 = call uil-get-grobject-from-id-on-dlg-or-wksp ("my-sa-2", WS);
  SA-3 = call uil-get-grobject-from-id-on-dlg-or-wksp ("my-sa-3", WS);

  call uil-display-message-list (SA-2, the current-message-pointer of SA-
    1);
  call uil-display-message-list (SA-3, the current-message-pointer of SA-
    1);
end
```

The method `my-multi-scroll-to-position` does the following things:

- It scrolls the scroll area to a particular position when a user drags the scroll thumb. To do this, the method calls the UIL procedure `uil-scroll-to-position`.
- It scrolls two other scroll areas in the multi-column scroll area. To do this, it calls the UIL procedure `uil-display-message-list` to set the current message pointers of the two other scroll areas to be equal to the current message pointer of the first scroll area. The result is that the three scroll areas are scrolled simultaneously and to the same position.

Selection and Unselection Methods for Multiple-Column Scrolling

The selection method of a scroll area is called whenever a user clicks on a message object within the scroll area. The unselection method is called whenever a user deselects the message object by clicking on something else. These methods are referenced by the `uil-message-selection-method` and `uil-message-unselection-method` attributes of the scroll area.

Note Specify the customized selection and unselection methods for the scroll area before you add message objects to the scroll message. The message objects that you subsequently add to the scroll area inherit the message selection and unselection methods as well as other attributes of the scroll area.

The message selection method has the following required argument signature:

```
guide-sample-message-selection
  (message-object: class uil-message-object,
   scroll-area: class uil-scroll-area, window: class g2-window)
```

The message unselection method has the following required argument signature:

```
guide-sample-message-unselection
  (message-object: class uil-message-object,
   scroll-area: class uil-scroll-area, window: class g2-window)
```

For example, the following customized message selection method, `my-multi-scroll-message-selection`, is referenced from the `uil-message-selection-method` attribute of the scroll area used to scroll the entire multi-column scroll area:

```
my-multi-scroll-message-selection (MO: class uil-message-object,
  SA: class uil-scroll-area, W: class g2-window)
SA-2: class uil-scroll-area;
SA-3: class uil-scroll-area;
Msg: item-or-value;
WS: class kb-workspace;
Index: integer;
begin
  call uil-message-selection (MO, SA, W);

  Index = call uil-get-index-of-scroll-message (SA, MO);

  WS = the workspace of SA;
  SA-2 = call uil-get-grobject-from-id-on-dlg-or-wksp ("my-sa-2", WS);
  SA-3 = call uil-get-grobject-from-id-on-dlg-or-wksp ("my-sa-3", WS);

  Msg = call uil-get-message-from-index (SA-2, Index);
  if (Msg exists and Msg is an item) then
    call uil-message-selection (Msg, SA-2, W);

  Msg = call uil-get-message-from-index (SA-3, Index);
  if (Msg exists and Msg is an item) then
    call uil-message-selection (Msg, SA-3, W);
end
```

The method `my-multi-scroll-message-selection` method does the following things:

- It selects the message object whenever a user clicks on it. To do this, it calls the UIL procedure `uil-message-selection`.
- It selects the message objects in the other two scroll areas that are at the same position as the message object that the user clicks on. To do this, it:
 - Calls the UIL procedure `uil-get-index-of-scroll-message` to get the index of the message object that the user clicks on.
 - Calls `uil-get-gobj-from-id-on-dlg-or-wksp` to get the other two scroll areas.
 - Calls the UIL procedure `uil-get-message` to get the message objects in the other two scroll areas that have the index returned by `uil-get-index-of-scroll-message`.
 - If message objects with this index exist in the other two scroll areas, it calls `uil-message-selection` to select these messages.

The message unselection method, `my-multi-scroll-message-unselection`, is identical to `my-multi-scroll-message-selection`, except that it calls `uil-message-unselection`, rather than `uil-message-selection`:

```
my-multi-scroll-message-unselection (MO: class uil-message-object,  
  SA: class uil-scroll-area, W: class g2-window)  
SA-2: class uil-scroll-area;  
SA-3: class uil-scroll-area;  
Msg: item-or-value;  
WS: class kb-workspace;  
Index: integer;  
begin  
  call uil-message-unselection (MO, SA, W);  
  
  Index = call uil-get-index-of-scroll-message (SA, MO);  
  
  WS = the workspace of SA;  
  SA-2 = call uil-get-gobj-from-id-on-dlg-or-wksp ("my-sa-2", WS);  
  SA-3 = call uil-get-gobj-from-id-on-dlg-or-wksp ("my-sa-3", WS);  
  
  Msg = call uil-get-message-from-index (SA-2, Index);  
  if (Msg exists and Msg is an item) then  
    call uil-message-unselection (Msg, SA-2, W);  
  
  Msg = call uil-get-message-from-index (SA-3, Index);  
  if (Msg exists and Msg is an item) then  
    call uil-message-unselection (Msg, SA-3, W);  
end
```


Empty Methods for Scroll Areas with Hidden Scroll Bars

For the scroll areas whose scroll bars are hidden, you must specify empty methods for the following attributes to ensure that nothing happens when users accidentally click the hidden scroll bars:

```
uil-message-selection-method  
uil-message-unselection-method  
uil-scroll-increment-method  
uil-scroll-increment-line-method  
uil-scroll-decrement-method  
uil-scroll-decrement-line-method  
uil-scroll-to-position-method
```

Each empty method must have the required argument signature for its particular kind of method. However, the body of each empty method contains no code. For example, the empty scroll increment method looks like this:

```
my-multi-scroll-increment-noop (SA: class uil-scroll-area)  
begin  
end
```

Note In this example, clicking on message objects in the scroll areas with hidden scroll bars has no effect, because the selection and unselection methods of these scroll areas are empty. However, you may want to be able to select all message objects in a row by clicking on message objects in the scroll areas with hidden scroll bars.

To do this, create a selection method and an unselection method for each scroll area with a hidden scroll bar to select and unselect all message objects at the same position as the message object that the user clicks on. You can model these selection and unselection methods on the methods used for the scroll area with the visible scroll bar, as shown in [Specifying Selection and Unselection Methods for Scroll Areas](#).

Summary of Scroll Area Menu Choices

Menu Choice	Description
table	Shows the attribute table of the scroll area.
go to subworkspace	Opens a subworkspace that contains any message objects that are in the scroll area but are not currently displayed.
transfer	Places the scroll area on the mouse and transfers it to a different workspace.
name	Opens an editor allowing a name to be entered or changed.
rotate/reflect	Opens a menu of rotation and reflection options.
change size	Opens a G2 dialog in which you can change the size of the scroll area.
lift to top	Displays the scroll area on top of another object on a workspace.
drop to bottom	Drops the scroll area behind another object on a workspace.
describe	Shows a description of the object and its relations.
edit array	Opens the Edit Array dialog. Do not attempt to use this dialog to editing scroll areas.
edit list	Opens the Edit List dialog, which you can use to add message objects to or delete message objects from the scroll area.
edit configuration	Opens the Configuration Editor, in which you can select, delete, copy, or edit configurations.
configure	Applies the current configuration to the scroll area.
delete.	Posts a confirmation dialog to the user and if confirmed, deletes the scroll area.
initialize	Calls the initialization method of the scroll area. This menu choice is visible only when the scroll area is enabled.

Menu Choice	Description
move	Opens the Move Object dialog, which enables you to position the scroll area precisely.
clone.	Makes a copy of the scroll area and places it next to the original scroll area on the workspace.
disable./enable.	Disabling the scroll area prevents the select menu choice from being shown or run on the messages contained in the scroll area. The colors of the scroll area change to reflect its disabled state.
left right reflection	Reorients the scroll-area's scroll bar from left to right, or vice versa.
edit scroll area	Invokes the Edit Scroll Area dialog, in which you can edit the various attributes of the scroll area.

Summary of Message Object Menu Choices

Menu Choice	Description
table	Shows the attribute table for the message object.
transfer	Places the message object on the mouse so that you can move it to a different workspace.
color	Posts a series of G2 menus allowing icon regions to have their colors altered. GUIDE provides a Configuration Editor as a preferred means of changing the colors of icon regions.
lift to top	Displays the message object on top of another object on a workspace.
drop to bottom	Drops the message object behind another object on a workspace.
describe	Shows a description of the object and its relations.
describe configuration	Displays a description of the message object's current configuration.

Menu Choice	Description
edit configuration	Opens the Configuration Editor, which you can use to select, delete, copy, or edit configurations.
configure	Applies the current configuration to the message object.
initialize	Calls the initialization method of the message object. This menu choice is visible only when the message object is enabled.
edit.	Opens a GUIDE editor for modifying message contents
clone.	Makes a copy of the message object and places it below the original message object in the scroll area.
disable./enable.	Disabling the message object prevents the select menu choice from being shown or run on the message object. The colors of the message object change to reflect its disabled state
delete.	Posts a confirmation dialog to the user and if confirmed, deletes the message object.
select	Calls <code>uil-select-message-on-window</code> and highlights the message.
edit message	Open the Edit Message dialog, in which you can edit attributes of the message object.

Sliders

Describes how to use and edit Sliders.

Introduction **291**

Summary of Slider Menu Choices **296**

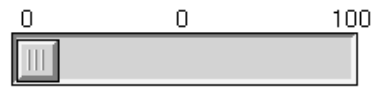


Introduction

A slider is a graphical object that enables you to display and select numeric values by moving a pointer, called a **thumb**, along a horizontal or vertical track.

GUIDE provides a variety of different styles of sliders. Sliders of all styles behave the same; they differ from each other only in appearance:

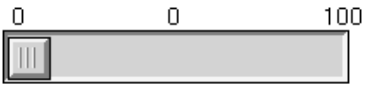
Slider Styles



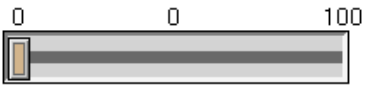
`uil-slider`



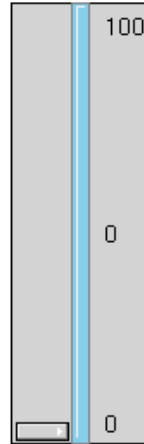
`uil-slider-thin`



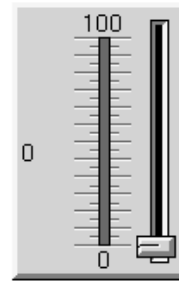
`uil-slider-regular`



`uil-slider-special`



`uil-slider-vertical`



`uil-slider-vertical
-with-scale`



`uil-slider-horizontal`

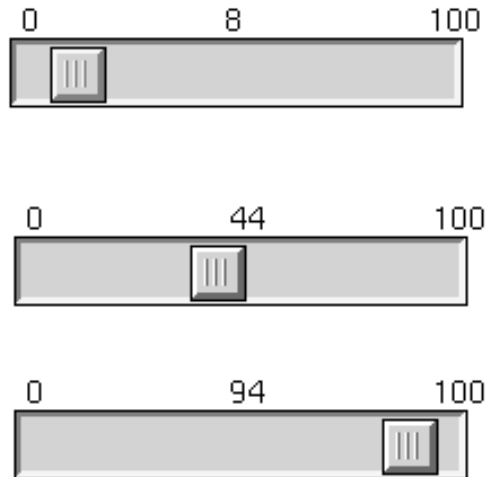
Using Sliders

Every slider has a thumb whose position determines the current value of the slider. Three numeric values appear with each slider, representing the minimum value, current value, and maximum value of the slider.

Every slider can have a source object and a target object. When a slider is updated with the value of its source object, the current value of the slider is changed to this value, and the thumb is moved to the position indicated by the value. When a slider is concluded, its current value is sent to its target object.

The following figure illustrates a regular slider (`uil-slider`) whose minimum and maximum values are 0 and 100, and whose thumb is shown at three different positions:

Slider Showing Different Current Values



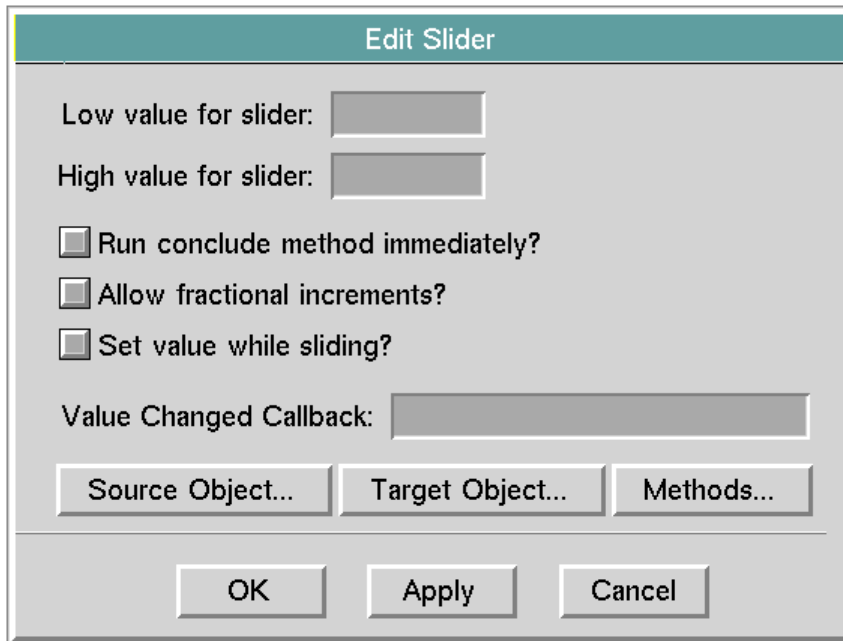
Creating Sliders

You can create sliders by choosing `Tools > Slider` from the GUIDE menu bar and choosing one of the styles of sliders available: `uil-slider`, `uil-slider-thin`, `uil-slider-regular`, `uil-slider-special`, `uil-slider-vertical`, `uil-slider-vertical-with-scale`, and `uil-slider-horizontal`.

You can also create sliders programmatically, using the UIL procedure `uil-create-slider`. For information about this procedure, see the *G2 GUIDE/UIL Procedures Reference Manual*.

Editing Sliders

The Edit Slider dialog enables you to specify the minimum and maximum values of a sliders, as well as other aspects of its behavior. To open this dialog, click the slider and choose `edit slider` from its menu. The Edit Slider dialog looks like this:



The following table lists and describes the components of the Edit Slider dialog:

Components of Edit Slider Dialog

Component	Description
Low value for slider	The smallest possible value that the slider can represent, shown at the left end of a horizontal slider or the bottom of a vertical slider.
High value for slider	The largest possible value that the slider can represent, shown at the right end of a horizontal slider or the top of a vertical slider.

Components of Edit Slider Dialog

Component	Description
Run conclude method immediately?	If this choice is toggled on, the conclude method of the slider is run immediately after any change to the current value of the slider. If this choice is toggled off, the conclude method is not run until some other action causes it to be run, such as an action that causes all the values in a dialog containing the slider to be concluded.
Allow fractional increments?	If this option is selected, the current value of the slider can have three decimal places. If this option is not selected, the slider can have only one decimal place.
Set value while sliding?	If this option is selected, moving the thumb causes the current value of the slider to change continuously. If this option is not selected, the current value of the slider is not changed until the user moving the thumb releases the mouse button.
Value Changed Callback	In this edit box you can specify a callback procedure that is invoked automatically whenever the thumb is moved or the value of the slider is changed. By default, no value change callback is specified. For information about how to write callback procedures, see Creating Methods, Actions, and Callbacks .
Source Object	Opens the Edit Source Object & Attribute dialog. The Edit Source Object & Attribute dialog enables you to specify the source object and attribute for the slider that you are editing. For information about this dialog, see Edit Source Object & Attribute Dialog .

Components of Edit Slider Dialog

Component	Description
Target Object	Opens the Edit Target Object & Attribute dialog. The Edit Target Object & Attribute dialog enables you to specify the target object and attribute for the check button that you are editing. For information about this dialog, see Edit Target Object & Attribute Dialog .
Methods	<p>Opens the Edit Methods dialog, in which you can create customized methods for the check box.</p> <p>For information about how to create customized methods, see Methods, Actions, and Callbacks.</p>

Summary of Slider Menu Choices

Menu Choice	Description
table	Shows the attribute table for the slider.
transfer	Places the slider on the mouse so that you can move it to a different workspace.
name	The name of the slider object.
change size	Enables you to change the size and shape of the slider by dragging its edges.
color	Opens dialogs in which you can modify the colors of different regions of the slider.
edit configuration	Opens the Configuration Editor, which you can use to select, delete, copy, or edit configurations that apply to the slider.
delete.	Posts a confirmation dialog to the user and if confirmed, deletes the slider.
move.	Opens the Move Object dialog, which you can use to position the slider precisely on the dialog subworkspace or workspace that contains it.
clone.	Clones the slider object.

Menu Choice	Description
disable./enable.	Disabling the slider prevents the select menu choice from being shown or run on the slider. The colors of the slider change to reflect its disabled state
rotate.	Rotates the slider by 90 degrees.
edit slider	Open the Edit Slider dialog, in which you can edit attributes of the slider.

Text Objects

Describes how to create and edit text objects, which display read-only text.

Introduction **299**

Setting the Initial Contents of Text Objects **300**

Updating the Contents of Text Objects **300**

Specifying Formats for Text Objects **300**

Editing Text Objects **301**

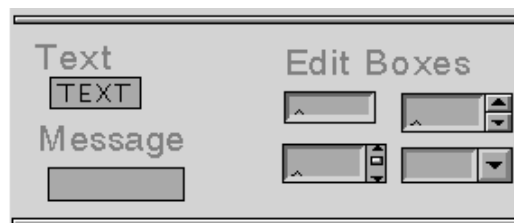
Summary of Text Object Menu Choices **304**



Introduction

Text objects only display text. Users cannot edit the contents of text objects.

The GUIDE palette includes the following icon for text objects:



You can add a text object to a master dialog or workspace by clicking the text object icon and dragging it to the subworkspace of the master dialog or to the

workspace. You can also add a text object by selecting the following choice from the GUIDE menu bar:

Item > GUIDE Objects > Texts > uil-text

The text icon object becomes attached to your cursor. You can drop it on the subworkspace of a master dialog or on a workspace.

Setting the Initial Contents of Text Objects

You can specify the initial contents of a text object by choosing **edit.** from the menu of the text object. In the editor, you can edit the unformatted contents of the text object, which is stored in the **message-contents** attribute of the text object. The following figure illustrates a text object when the editor is open on it, and after it has been edited:



You can edit a text object's size, position, state, and other properties, using the Edit Text dialog. For information about how to use this dialog, see [Editing Text Objects](#).

Updating the Contents of Text Objects

Your application can update the contents of text objects. The method that updates a text object is specified in the **uil-update-method** attribute of the text object.

You cannot conclude the value of a text object.

Specifying Formats for Text Objects

You can specify formats for text objects. Each format contains formatting information about the appearance of the text contained in the text object. The format also controls how quotation marks are displayed and how cases are handled.

You can specify a maximum number of characters to display in a text object. Setting a maximum number of characters can prevent text objects from growing too large.

You can select a format and specify a maximum number of characters for a text object using the Edit Text dialog. For information about how to use this dialog, see the following section.

Editing Text Objects

You can edit the appearance and behavior of text objects using the Edit Text dialog, the Select Format dialog, the Edit Source Object & Attribute dialog, the Edit Target Object & Attribute dialog, and the Edit Method dialog.

Edit Text Dialog

The Edit Text dialog enables you to edit attributes of a text object and to access other dialogs that you use to edit the appearance and behavior of the text object.

To open the Edit Text dialog, click the text object that you want to edit and choose edit text from the text object's menu. The Edit Text dialog looks like this:

The screenshot shows the 'Edit Text' dialog box with the following fields and options:

- Class:** NONE
- Name:** UNSPECIFIED
- Id:** (empty)
- Value:** Text
- Position:** X: 430, Y: 549
- Dimensions:** Width: 52, Height: 33
- Size:** Small, Medium, Large (all selected)
- State:** Enabled, Disabled (both selected)
- Maximum characters to display:** Unlimited
- Bounding box:** (unchecked)
- Format:** unspecified
- Buttons:** Source Object..., Methods..., OK, Apply, Cancel

The following table describes the components of the Edit Text dialog:

Components of Edit Text Dialog

Component	Description
Class	(read-only) Displays the class of text object that you are editing.
Name	(optional) Displays the current name of the text object that you are editing. Changing the displayed value updates the name of the text object. This field does not require a value. Its contents, if any, must be a valid symbolic name.
Id	(optional) Displays the ID of the text object that you are editing. Changing the ID updates the id attribute of the text object.
Value	<p>(optional) Displays the unformatted contents of the text object that you are editing. You can edit the Value field to modify the text object's unformatted contents. The unformatted contents are stored in the <code>message-contents</code> attribute of the text object.</p> <p>If you specify a format for the text object (through the Select Format dialog), this format is applied to the contents of the <code>message-contents</code> attribute, and the resulting formatted text is stored in the <code>text</code> attribute of the text object. Any limitation on the length of the contents that you specify in the Maximum characters to display field (see below) is also applied to the formatted text in the <code>text</code> attribute. You cannot directly access or edit the <code>text</code> attribute.</p>
Position	(required) The two edit box fields below the label Position display the current <code>item-x-position</code> and <code>item-y-position</code> of the text object that you are editing. Changes to the X and Y values move the text object to the new location. This is a required field and defaults to the text object's current position on the workspace. The X and Y values must be integers.

Components of Edit Text Dialog

Component	Description
Dimensions	(read-only) The two text fields below the label Dimensions display the current item-width and item-height of the text object that you are editing. You cannot edit this field.
Size	(required) The radio buttons below the label Size indicate the current size (small, medium, or large) of the text object that you are editing. You can change the size of the text object that you are editing by selecting the radio button whose label indicates the desired size. When this change is applied, GUIDE regenerates the text object in the new size without changing the text object's other attributes.
State	(required) The radio buttons below the label State (Enabled , Disabled), indicate the current enabled or disabled state of the object that you are editing. Users can change the state by selecting the radio button whose label indicates the desired state. When this change is applied, the text object is updated to reflect its new state.
Maximum characters to display	<p>(required) Specifies the maximum number of characters that can be displayed in the text object. Any characters in excess of this number are clipped. The default value is the symbol unlimited.</p> <p>Specifying a value in the Maximum characters to display field prevents the text field from expanding beyond its original dimensions. This is particularly useful for maintaining dialog layout and item alignment.</p> <p>The full, unformatted text is maintained in the message-contents attribute of the text object. The clipped and/or formatted text is placed in the text attribute, which users cannot access directly.</p>

Components of Edit Text Dialog

Component	Description
Bounding box	When this button is selected, GUIDE creates a border and wraps it around the text object. The border is given a name and is referred to in the text object's <code>uil-border-relation</code> attribute. If the toggle button is unselected and the text object has a border, the border is deleted and the <code>uil-border-relation</code> attribute is set to be the symbol unspecified.
Format	(optional) You can enter the name of an existing format to specify a format for this text object. For information about how to use formats, see Formats and Validation Criteria . To select from a list of existing formats, click the button to the right of the Format field. This opens the Select Format dialog, in which you can select a format for this text object. For more information about this dialog, see Select Format Dialog .
Source Object	Opens the Edit Source Object & Attribute dialog. The Edit Source Object & Attribute dialog enables you to specify the source object and attribute for the text object that you are editing. For information about this dialog, see Edit Source Object & Attribute Dialog . Note: Text objects cannot conclude their values and for this reason do not have target objects.

Summary of Text Object Menu Choices

Menu Choice	Description
table	Shows the attribute table for the text object.
transfer	Places the text object on the mouse and transfers it to a different workspace.
change min size	Opens a G2 dialog providing change size options.

Menu Choice	Description
color	Opens a series of G2 menus that you can use to change the colors of icon regions. GUIDE provides the Configuration Editor as the recommended way to change icon regions. For information about this dialog, see Using The GUIDE Configuration Editor .
lift to top	Displays the text object on top of another object on a workspace.
drop to bottom	Drops the text object behind another object on a workspace.
describe	Shows a description of the object and its relations.
edit configuration	Opens a GUIDE editor for selecting, deleting, copying, or editing configurations.
configure	Applies the current configuration to the text object.
initialize	Calls the initialization method of the text object. This menu choice is visible only when the text object is enabled.
edit.	Opens a G2 editor for modifying the text object's value.
move	Opens the Move Object dialog, which enables you to position the text box precisely.
clone.	Makes a copy of the text object and places it next to the original text on the workspace.
disable.	Changes the colors of the text object to reflect its disabled state.
delete.	Opens a confirmation dialog to the user and if confirmed, deletes the text object.
create border for text	Places a border around the text object.
edit text	Opens the Edit Text dialog, in which you can edit attributes of the text object.

Title Bars, Borders, and Separators

Describes how to use title bars, border, and separators to provide your user interface with visual definition.

Introduction **307**

Title Bars **307**

Borders **309**

Separators **313**

Summary of Title Bar Menu Choices **314**

Summary of Border Menu Choices **315**

Summary of Separator Menu Choices **317**



Introduction

You can use title bars, borders, and separators to provide your user interface with visual definition. These objects do not have values that can be concluded or updated.

Title Bars

A **title bar** is a text object that you can add to a workspace or dialog subworkspace to identify the workspace or subworkspace to users.

To add a title bar to a workspace or dialog subworkspace, select the icon labeled Title on the GUIDE palette and drop it on the workspace or dialog subworkspace.

When you drop a title bar on a dialog subworkspace, GUIDE automatically generates an anchored border for the subworkspace, if the subworkspace does not have a border. GUIDE resizes the title bar to span the width of the subworkspace and resizes the border to surround the entire subworkspace.

When you drop a title bar on a workspace, GUIDE automatically generates an anchored border and resizes the title bar and border to enclose all the objects that happen to be on the workspace.

A workspace or dialog subworkspace can have only one title bar. Dropping title bars on workspaces or dialog subworkspaces that already have title bars has no effect.

Using the Hide Button on Title Bars

Beginning in GUIDE/UII 5.0, title bars include hide buttons that users can click to hide the dialog or workspace that contains the title bar. By default, the hide buttons are disabled and invisible.

To make the hide buttons on Title Bars usable:

- 1 Apply the configuration `uil-dialog-title-configuration-with-title-button` to the title bar.

You can also apply a user-defined configuration that makes the hide button visible.

For information about how to use configurations, see [Specifying the Colors of UII Objects](#).

- 2 Set the attribute `uil-title-button-behavior-enabled` of the title bar to `true`.
- 3 Set the `uil-title-button-callback` attribute of the dialog title to the name of a procedure, or leave the default procedure `uil-title-button-callback`.

`uil-title-button-callback()` does the following:

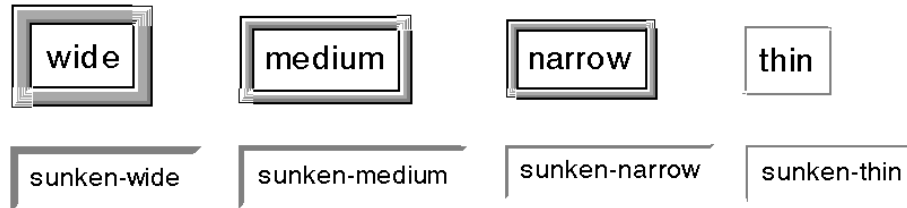
- If the dialog title is on a workspace and there is a Cancel button on the workspace, it selects the Cancel button. If there is no Cancel button on the workspace, it hides the workspace.
- If the dialog title is on a dialog and there is a Cancel button on the dialog, it selects the Cancel button. If there is no Cancel button on the dialog, it inserts the cancel actions (`uil-unsimulate-play-mode`, `uil-hide-dialog`, and `uil-release-dialog`) into the dialog's event queue.

Any callback that you specify for the `uil-title-button-callback` attribute must have the following argument signature:

title: class `uil-dialog-title`, *window*: class `g2-window`

Borders

The GUIDE palette provides icons for eight different border styles:



You can add stand-alone borders as decoration, or attach borders to edit boxes, text objects, dialog subworkspaces, or workspaces. Edit boxes, text objects, dialog subworkspaces, and workspaces are the only objects that can have borders.

Adding Borders

You can add a border to an object in either of two ways:

- Select the icon on the GUIDE palette for the border style that you want to use and drag it to the object.
- Select the following choice from the GUIDE Men Bar:

Item > GUIDE Objects > Borders

Under the Borders choice, you can select from all the different styles of borders. The icon for the border that you select becomes attached to your cursor, and you can drop it on the object to which you want to add a border.

The following table lists steps that you must follow to add borders to objects of different classes:

Adding Borders to Different Classes of Objects

Object	Steps for Adding Border to Object
Edit box or text object	<ol style="list-style-type: none">1 Drop border on or near object to which you want to add border.2 Choose snap to nearest text from the border menu. The border wraps around the edit box or text object.
Dialog subworkspace	<ol style="list-style-type: none">1 Drop the border on an empty area within the subworkspace.2 Choose anchor to workspace edge from the border menu. The border expands to surround the entire subworkspace.
Workspace	<ol style="list-style-type: none">1 Drop the border on an empty area within the workspace.2 Choose anchor to workspace edge from the border menu. The border expands to surround all the objects on the workspace.

Note To access a border's menu, click the lower left corner of the border.

If you want to detach a border from a workspace or dialog subworkspace, choose **release border from workspace edge** from the border menu. You can then resize the border and move it anywhere within the workspace or subworkspace.

You can also add borders programmatically through a series of calls to UIL procedures. For information about UIL procedures that manipulate borders, see the chapter on borders in the *G2 GUIDE/UIL Procedures Reference Manual*.

Deleting Borders

To delete a border, choose **delete.** from the border's menu.

Deleting an object with an anchored border deletes the border as well. However, you can delete the border without deleting the object that the border surrounds.

Moving Objects with Borders

When you move or transfer an object with a border, the border is automatically moved or transferred with the object.

You can move a border by grabbing the upper right corner or the lower left corner with the mouse.

Resizing Borders

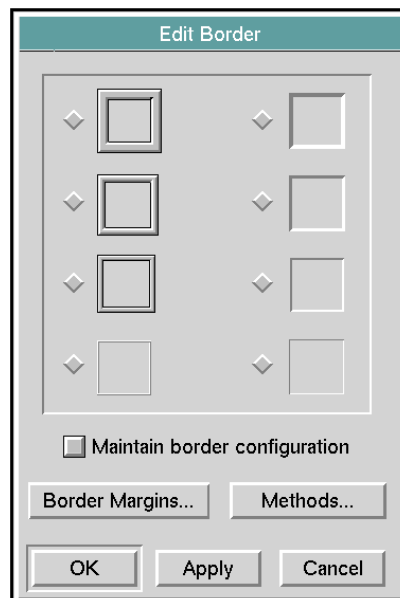
You can resize a border by dragging its upper right or lower left corner.

After you resize a UIL object that has a border, move the UIL object slightly. This causes the border to reposition itself with respect to it the UIL object. The border does not reposition itself properly until you move the UIL object.

Edit Border Dialog

You can use the Edit Border dialog to edit characteristics of stand-alone borders and borders anchored to workspaces or dialog subworkspaces. You cannot use the Edit Border dialog to edit borders anchored to edit boxes or text objects.

To open the Edit Border dialog, click the border that you want to edit and choose edit border from the border menu. The Edit Border dialog looks like this:



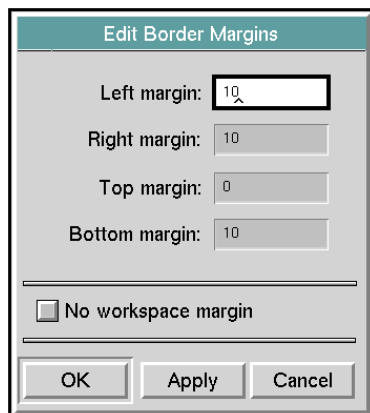
The following table describes the components of the Edit Border dialog:

Components of Edit Border Dialog

Component	Description
border styles (see previous figure)	Select one of the border styles illustrated in the Edit Border dialog by selecting the radio button to the left of the style that you want to use. In the figure above, the border style in the top left corner of the Edit Border dialog is selected.
Maintain border configuration	Selecting this button applies the configuration of the original border to the new border.
Border Margins	Opens the Edit Border Margins dialog, which enables you to specify the size of the margins around the border. See the following section for information about the Edit Border Margins dialog.
Methods	Opens the Edit Methods dialog, in which you can create customized methods for the border. For information about how to create customized methods, see Methods, Actions, and Callbacks .

Edit Border Margins Dialog

To open the Edit Border Margins dialog, click the Edit Border Margins button in the Edit Border dialog. The Edit Border Margins dialog looks like this:



The following table describes the components of the Edit Border Margins dialog:

Components of Edit Border Margins Dialog

Component	Description
Left margin	Displays the number of workspace units used for the left margin. This is a non-required field, but if modified must contain a positive whole number value.
Right margin	Displays the number of workspace units used for the right margin. This is a non-required field, but if modified must contain a positive whole number value.
Top margin	Displays the number of workspace units used for the top margin. This is a non-required field, but if modified must contain a positive whole number value.
Bottom margin	Displays the number of workspace units used for the bottom margin. This is a non-required field, but if modified must contain a positive whole number value.
No workspace margin/ Border anchored to workspace	<p>When this button is selected, its label is Border anchored to workspace. Selecting this button anchors the border to the workspace and activates automatic shrink-wrap for the workspace whenever items are added or repositioned.</p> <p>When this button is not selected, its label is No workspace margin, and the border acts like any other item on the workspace.</p>

Separators

You can use separators to group and distinguish features of your user interface. You can add separators to workspaces and to master dialog subworkspaces.

The G2 GUIDE palette provides icons for separators of four different thicknesses. To add a separator to a workspace or dialog subworkspace, select the icon that you want to use and drop it where you want to add the separator.

You can also add a separator by selecting the following choice from the GUIDE menu bar:

Item > GUIDE Objects > Separators

Under the Separators menu choice, you select the style of separator that you want to add. The icon for the separator that you select becomes attached to your cursor, and you can drop it on any workspace or master dialog subworkspace.

You can move and resize a separator dragging either one of its ends.

Separators have menu choices for operations such as deleting, moving, cloning, and rotating the separator. To access the menu, click the left end point of the separator.

The **rotate.** menu choice rotates the separator by 90 degrees. This enables you to change the orientation of the separator from horizontal to vertical, or from vertical to horizontal.

Summary of Title Bar Menu Choices

Menu Choice	Description
table	Shows the attribute table of the title bar.
transfer	Places the title bar on the mouse and transfers it to a different workspace.
change min size	Opens a G2 dialog providing change size options.
color	Opens a series of G2 menus allowing icon regions to have their colors altered. GUIDE provides the Configuration editor as a preferred means of changing icon regions.
lift to top	Displays the title bar on top of another object on a workspace.
drop to bottom	Drops the title bar behind another object on a workspace.
describe	Shows a description of the object and its relations.
lift workspace to top	Lifts the title bar and its workspace to the top.
edit configuration	Opens a GUIDE editor for selecting, deleting, copying, or editing configurations.

Menu Choice	Description
configure	Applies the current configuration to the title bar.
initialize	Calls the initialization method of the title bar. This menu choice is visible only when the title bar is enabled.
edit.	Opens an editor allowing the current text of the title bar to be modified.
move	Opens the Move Object dialog, which enables you to move the object precisely. This menu choice is visible only when the title bar is enabled.
create border for text	Places a border around the dialog title.
clone.	Makes a copy of the title bar and places it next to the original title bar on the workspace.
disable./enable.	Disables or enables the title bar.
delete.	Opens a confirmation dialog to the user and if confirmed, deletes the title bar.

Summary of Border Menu Choices

Menu Choice	Description
table	Shows the attribute table for the border.
name	Opens an editor allowing a name to be entered or changed.
rotate/reflect	Opens a menu of rotation and reflection options.
change size	Opens a G2 dialog that you can use to change the size of the border.
color	Post a series of G2 menus enabling you to change the color of the icon. GUIDE provides the Configuration Editor as the recommended way to change icon regions. For information about this dialog, see Using The GUIDE Configuration Editor .

Menu Choice	Description
lift to top	Displays the border on top of other objects on a workspace.
drop to bottom	Drops the border behind other objects on a workspace.
describe	Shows a description of the object and its relations.
edit configuration..	Opens GUIDE's configuration editor for selecting, deleting, copying, or editing configurations.
configure	Applies the current configuration to the border.
move	Opens the Move Object dialog, which enables you to position the edit box precisely.
clone.	Makes a copy of the border object and places it next to the original border on the workspace.
delete.	Opens a confirmation dialog to the user and if confirmed, deletes the border.
anchor to workspace edge/release border from workspace edge	Attaches the border object to the workspace while enforcing its margins, or releases border.
snap to nearest text	Wrap the border around the edit box or text object.
edit border	Opens the Edit Border dialog, in which you can edit the attributes of the object.

Note A border anchored to a workspace containing a title bar can not be deleted as long as the title bar exists. You must delete the title bar before you delete the border.

Summary of Separator Menu Choices

Menu Choice	Description
table	Shows the attribute table for the separator.
move	Attaches the separator to the mouse and drops it when the user clicks the mouse.
name	Opens an editor allowing a name to be entered or changed.
color	Opens a series of G2 menus allowing icon regions to have their colors altered. GUIDE provides the Configuration Editor as a preferred means of changing the colors of icon regions.
describe	Shows a description of the separator and its relations.
describe configuration	Displays a description of the separator's current configuration.
edit configuration	Opens the GUIDE Configuration Editor, in which you can select, delete, copy, or edit configurations.
configure	Applies the current configuration to the separator.
delete.	Opens a confirmation dialog to the user and if confirmed, deletes the separator.
move	Opens the Move Object dialog, which enables you to move the object precisely. This menu choice is visible only when the separator is enabled.
clone.	Makes a copy of the separator and places it next to the original separator on the workspace.
rotate.	Rotates the line separator 90 degrees.

Navigation Buttons and Other Tools

Describes how to add navigation buttons, help buttons, print workspaces, and the GUIDE garbage pail to workspaces.

Introduction **320**

Navigation Buttons **320**

Edit Navigation Button Dialog **322**

The Print Workspace Button **324**

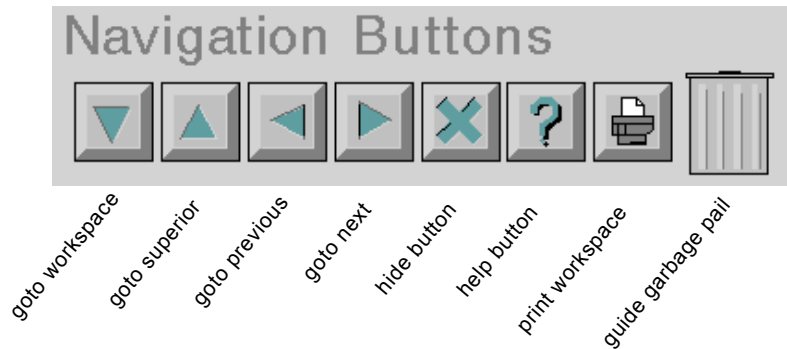
The GUIDE Garbage Pail **324**

Summary of Navigation Button Menu Choices **325**



Introduction

The GUIDE palette includes the following icons for navigation buttons and other tools that you can drop onto the workspaces of your G2 applications:



You can add navigation buttons to a workspace by dragging these icons to the workspace. You can also add navigation buttons by selecting the GUIDE menu bar choice:

Item > GUIDE Objects > Buttons > uil-navigation button

A list of all classes of navigation buttons appears under the uil-navigation button menu choice. Select the class of navigation button that you want to add. The icon for that class of navigation button becomes attached to your cursor. You can drop the button on any workspace.

Navigation Buttons

The GUIDE palette contains icons that you can use to add navigation buttons to the workspaces of your G2 application. To add a navigation button to a workspace, click the icon for the kind of navigation button that you want to add and drop it on the workspace.

Classes of Navigation Buttons

GUIDE supports the following classes of navigation buttons:

Navigation Button Class	Description
uil-goto-workspace-button	Navigates to a named workspace (if defined), to the subworkspace of this button (if one exists), or to a subworkspace that it creates.
uil-goto-superior-button	Navigates to the item that is superior to the workspace of this button, and then displays the workspace of that item.
uil-goto-previous-button	Navigates to a named workspace (if defined), to the subworkspace of this button (if one exists), or to a subworkspace that it creates.
uil-goto-next-button	Navigates to a named workspace (if defined), to the subworkspace of this button (if one exists), or to a subworkspace that it creates.
uil-hide-button	Hides the workspace that contains this button.
uil-help-button	Navigates to a workspace on which you can place help information.

Note It is not recommended practice to use navigation buttons to open dialogs. A dialog that is opened by a navigation button does not have full GUIDE support for field editing, automatic updating and concluding of values, and other essential features of the GUIDE dialog system.

When GUIDE is not loaded, you can create navigation buttons by selecting New Object from the KB Workspace menu.

Modules Supporting Navigation Buttons

The uil module supports full navigation button functionality.

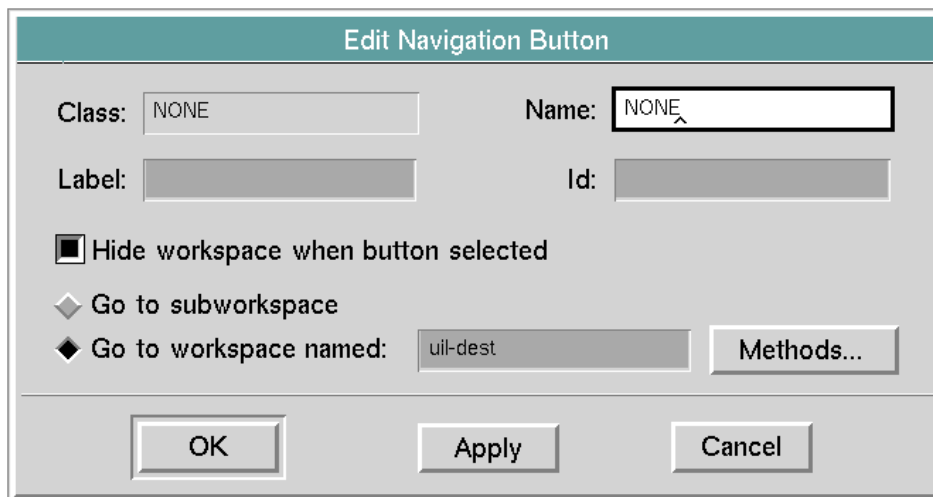
The uilroot module provides navigation buttons and some of the procedures, class definitions, and functions that support the full navigation functionality of navigation buttons. It does not support some features of navigation buttons, such

as labels, or the ability to edit navigation buttons by means of the Edit Navigation Button dialog.

You can use `uilroot` with G2 applications that do not require the full navigation button functionality. When you work with just the `uilroot` loaded, support for navigation buttons is limited to creation, configuration, activation and deletion.

Edit Navigation Button Dialog

To open the Edit Navigation Button dialog, click the navigation button that you want to edit and select `edit navigation button` from the navigation button menu. The Edit Navigation Button dialog looks like this:



The dialog box is titled "Edit Navigation Button". It features a teal header bar. Below the header, there are four input fields: "Class" (containing "NONE"), "Name" (containing "NONE"), "Label", and "Id". Below these fields are three radio button options: "Hide workspace when button selected" (which is checked), "Go to subworkspace", and "Go to workspace named:". The "Go to workspace named:" option includes a text field containing "uil-dest" and a "Methods..." button. At the bottom of the dialog are three buttons: "OK", "Apply", and "Cancel".

The following table describes the components of the Edit Navigation Button dialog:

Components of Edit Navigation Button Dialog

Component	Description
Class	(read-only) Displays the class of navigation button that you are editing.
Name	Displays the current name of the navigation button that you are editing. Changing the displayed value updates the name of the navigation button. This field is not required. Its contents, if any, must be a valid symbolic name.

Components of Edit Navigation Button Dialog

Component	Description
Label	Displays the label of the navigation button that you are editing. Changing the label causes a new label to be generated for the navigation button with the new text. This is a non-required field, but if entered, it must be a valid text entry. Quotation marks are not required. If included, they will become part of the label.
Id	(optional) Displays the ID of the navigation button that you are editing. Changing the ID updates the id of the navigation in its table.
Hide workspace when button selected	If this option is selected, the workspace that contains the navigation button is hidden when a user clicks the navigation button. If this option is not selected, the workspace is not hidden when a user clicks the button.
Go to subworkspace	Select this option if you want the button to navigate to the subworkspace of the workspace that contains the button. This option is disabled for <code>uil-goto-superior-button</code> and <code>uil-hide-button</code> navigation buttons.
Go to workspace named	If this option is selected, you can enter the name of a workspace to which you want the button to navigate. This option is disabled for <code>uil-goto-superior</code> and <code>uil-hide-button</code> navigation buttons.
Methods	Opens the Edit Methods dialog, in which you can create customized methods for the navigation buttons. For information about how to create customized methods, see Methods, Actions, and Callbacks .

The Print Workspace Button

Clicking on a print workspace button prints the workspace or dialog subworkspace where the print workspace button resides.

To add a print workspace button to a workspace, click the print workspace icon in the GUIDE palette and drop it on the workspace. You cannot use the print workspace button on a dialog subworkspace.

The print workspace button prints the workspace on a default printer using default settings. To specify these settings, choose the following from the G2 Main Menu:

System Tables > Printer Setup

This opens the Printer-Setup table, in which you can specify default printer settings.

Note To enable users to select a printer and specify printing options for printing a workspace, you can add a Print button to the workspace. Clicking on the Print button opens the Printer Setup Dialog. In this dialog, users can specify options for printing the workspace that contains the Print button.

To add a Print button to a workspace, click the Print Workspace Dialog icon in the More Options palette and drop it on the workspace.

You can also print a workspace by choosing:

KB Workspace > Print

This menu choice opens a series of dialogs in which you can specify options for printing the workspace. For information about how to use these dialogs, see [Printing GUIDE Workspaces](#).

The GUIDE Garbage Pail

You can delete dialogs and UIL controls by dragging and dropping them on a GUIDE Garbage Pail.

To add a GUIDE Garbage Pail to a workspace, click the GUIDE Garbage Pail icon in the GUIDE palette and drop it on the workspace.

Caution You cannot recover dialogs or UIL controls after you drop them on the GUIDE Garbage Pail.

Summary of Navigation Button Menu Choices

Menu Choice	Description
table	Shows the attribute table for the navigation button.
go to subworkspace	Shows the subworkspace of the button.
transfer	Places the button on the mouse and transfers it to a different workspace.
name	Opens an editor allowing a name to be entered or changed.
rotate/reflect	Opens a menu providing rotation and reflection options.
change size	Opens a G2 dialog providing change size options.
color	Opens a series of menus enabling you to change the size of the icon. GUIDE provides the Configuration Editor as the recommended way to change the colors of icon regions. For information about this dialog, see Using The GUIDE Configuration Editor .
lift to top	Displays the navigation button on top of another object on a workspace.
drop to bottom	Drops the navigation button behind an other object on a workspace.
describe	Shows a description of the object and its relations
edit configuration	Opens the GUIDE Configuration Editor, in which you can select, delete, copy, or edit configurations.
configure	Applies the current configuration to the navigation button.
delete.	Opens a confirmation dialog to the user and if confirmed, deletes the navigation button.

Menu Choice	Description
initialize	Calls the initialization method of the navigation button. This menu choice is visible only when the navigation button is enabled.
move	Opens the Move Object dialog, which enables you to position the navigation button precisely.
clone.	Makes a copy of the navigation button and places it next to the original button on the workspace.
disable./enable.	Disabling the navigation button prevents the select menu choice from being shown or run on the button. The color of the button is changed to reflect its disabled state.
select	Executes the handler method specified on the button.
edit navigation button	Opens the Edit Navigation Button dialog, in which you can edit the attributes of the navigation button.

Advanced Features

Chapter 19: Formats and Validation Criteria

Describes how to create and edit reusable formats for edit boxes, message objects, and text objects.

Chapter 20: Specifying Source and Target Objects

Describes how to specify the objects from which the graphical objects on a dialog are updated and to which the graphical objects conclude their values.

Chapter 21: Creating Temporary Storage Objects

Describes how to create and use temporary storage objects, which you can use when you process data while it is being updated into or concluded from a dialog.

Chapter 22: Methods, Actions, and Callbacks

Describes how to create and use UIL methods, actions, and callbacks.

Chapter 23: Help Dialog

Describes the GUIDE help facility.

Chapter 24: Creating Custom UIL Subclasses

Describes how to create customized subclasses of system-defined UIL classes provided with GUIDE.

Chapter 25: Specifying the Colors of UIL Objects

Describes how to create reusable objects called configurations, which specify the colors of the different regions of the graphical components in your user interface.

Chapter 26: Upgrading GUIDE Applications

Describes how to modify dialogs and other components of a user interface created with earlier versions of GUIDE, to take advantages of the features introduced in newer versions.

Formats and Validation Criteria

Describes how to create and edit reusable formats for edit boxes, message objects, and text objects.

Introduction **329**

Creating Formats **331**

Applying and Editing Formats **333**



Introduction

Formats are reusable objects that specify how text is formatted in edit boxes, message objects, and text objects. Formats can also establish criteria for validating data that users enter into edit boxes.

You can associate a format any number of different objects to which you want to apply the formatting or validating criteria specified by that format. All formats are instances of the class `uil-format-specification-class`.

Formatting Rules for Edit Boxes, Message Objects, and Text Objects

For edit boxes, message objects, and text objects, a format can specify formatting rules that govern:

- Quotation marks
- Capitalization

Validation Criteria for Edit Boxes

For edit boxes only, a format can specify validation rules that govern:

- Data format type, such as symbol, text, quantity, and so on
- Minimum and maximum text values
- Minimum and maximum text length
- Default values
- Date and time formats
- Legal values for the edit box

If you apply a format to a text object (through the Select Format dialog), the format is applied to the contents of the `message-contents` attribute of the text object. The resulting formatted text is stored in an attribute named `text`, which you cannot view or access directly. Any limitation on the length of the text object that you specify in the Maximum characters to display field of the Select Format dialog is also applied to the formatted text in the `text` attribute.

GUIDE provides a number of useful predefined formats. You can also create your own customized formats.

Creating Customized Validation Procedures or Functions

You can write your own validation procedures or functions to handle application-specific validations. By selecting the format type Procedure Call or Function Call in the Edit Format Specification dialog, you can substitute your own customized procedure or function call for the validation method provided by GUIDE. You enter the procedure or function to call in the Validation Method Name field in the Edit Format Specification dialog.

Data validation occurs immediately at the conclusion of an edit if the `uil-validate-immediately` attribute on the edit box is set to `true`. Validation is also triggered when the Apply or OK button is clicked and the button includes the validation method (`uil-call-validation-method`) among its actions.

You can create new validation methods, using the GUIDE Method Help dialog. For information about how to use this dialog, see [Creating Methods, Actions, and Callbacks](#).

Creating Formats

You can create formats in the following ways:

- Open the More Options palette by clicking the More Options button in the GUIDE palette. In the More Options palette, click the Format Specification icon and drop it on the workspace where you want to add the format.

To edit the new format, choose **edit format specification** from its menu. This menu choice opens the Edit Format Specification dialog, in which you can specify formatting information and validation criteria in the new format.

- Click the New button in the Select Format dialog. Clicking on the New button creates a new format and opens the Edit Format Specification dialog.

You can open the Select Format dialog by clicking on the button to the right of the Format field in the Edit Text dialog, the Edit Message dialog, or the Edit Edit Box dialog.

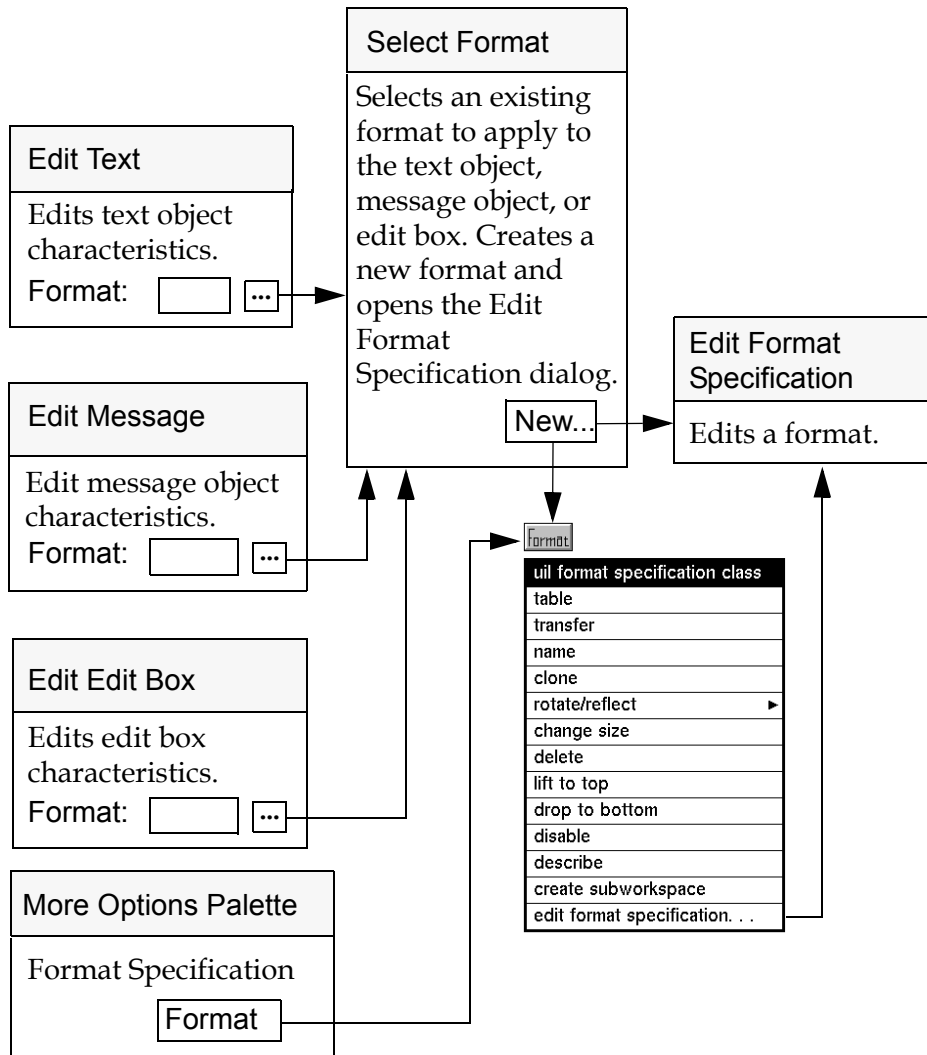
- Selecting the following choice from the GUIDE menu bar:

Item > GUIDE Objects > Format Specification

The icon for the format specification becomes attached to your cursor, and you can drop it on any workspace.

The following figure illustrates how you access the Select Format and Edit Format Specification dialogs:

Dialogs for Editing Formats



Applying and Editing Formats

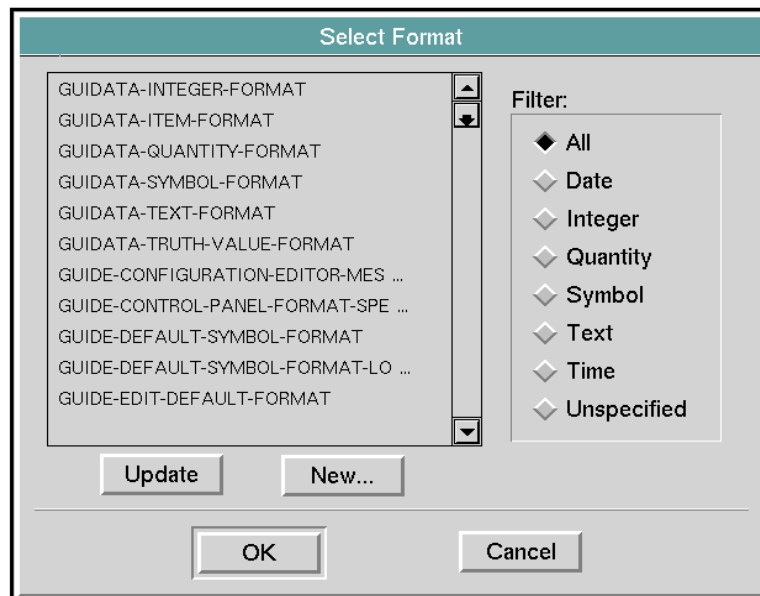
You can apply an existing format to an edit box, message object, or text object using the Select Format dialog.

Select Format Dialog

The Select Format dialog enables you to apply an existing format to a text object, message object, or edit box.

You can also use the Select Format dialog to access the Edit Format Specification dialog, in which you can create a new format for a text object, message object, or edit box. For more information about the Edit Format Specifications dialog, see the section immediately following this section.

To open the Select Format dialog, click the button to the right of the Format field in the Edit Text dialog, the Edit Message dialog, or the Edit Edit Box dialog. The Select Format dialog looks like this:



The following table describes the components of the Select Format dialog:

Components of Select Format Dialog

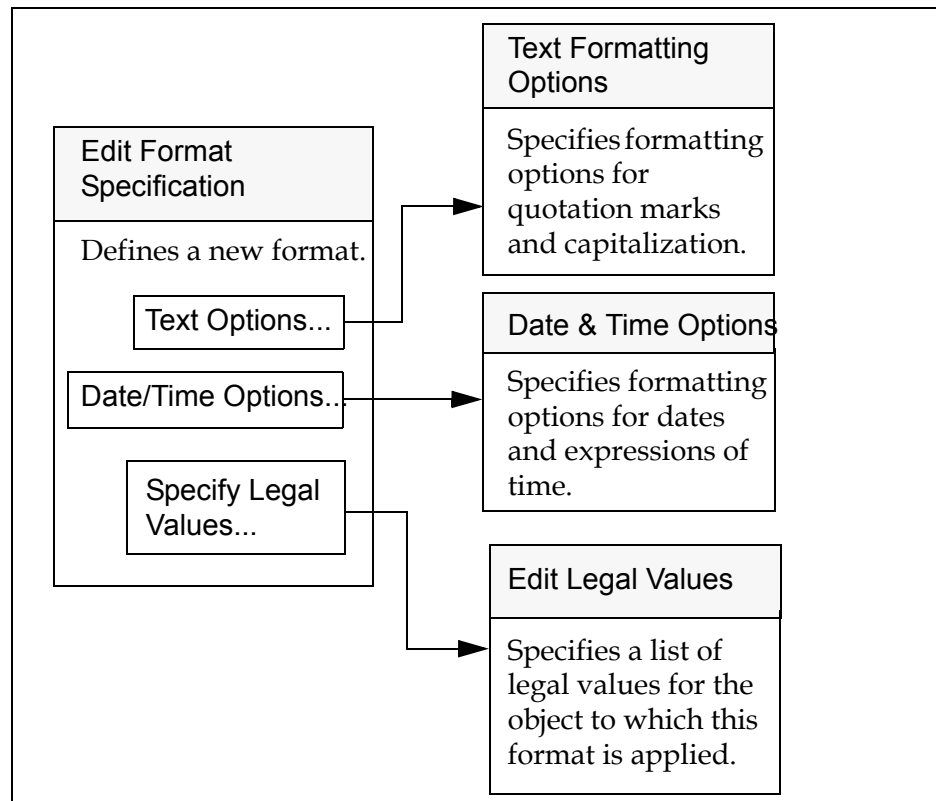
Component	Description
Filter	Specifies which of the existing formats are displayed in the scroll area on the left side of the Select Format dialog.
All	Display all existing formats.
Date	Display formats for dates.
Integer	Display formats for integers.
Quantity	Display formats for quantity values.
Symbol	Display formats for symbols.
Text	Display formats for text.
Time	Display formats for expressions of time.
Unspecified	Displays format with the default format type unspecified.
OK	Applies your selection.
Update	Refreshes the scroll area to include any new formats of the specified Filter type.
New	Clicking the New button creates a new format and opens the Edit Format Specification dialog, in which you can edit the new format. For information about how to edit a format, see the following section.

Edit Format Specification Dialog

The Edit Format Specification dialog enables you to define a format that you can apply to text objects, message objects, and edit boxes.

To define a format, you use the Edit Format Specification together with other dialogs that you can access from the Edit Format Specification dialog. The following figure illustrates the dialogs for editing formats.

Dialogs for Editing Formats



You can open the Edit Format Specification dialog to create a new format or to edit an existing format.

- To create a new format, open the Edit Format Specification dialog by clicking the New button in the Select Format dialog. When you click the New button, a format is created and the following icon is added to the workspace that contains the item that you are editing:



- To edit an existing format, choose **edit format specification** from the menu of that format to open the Edit Format Specification dialog.

The Edit Format Specification dialog looks like this:

Edit Format Specification

Name:

Format Type:

- Unspecified
- Symbol
- Text
- Integer
- Quantity
- Date
- Time
- Procedure Call
- Function Call

Validation Method Name:

Validation Failure Handler:

Default Value Handling:

- Unspecified
- Use 1st value in value array
- Use type as basis
- Use attribute name
- Allow none as valid option
- Revert value on error

Field Limiters:

Minimum Value:

Maximum Value:

Minimum Length:

Maximum Length:

The following table describes the components of the Edit Format Specification dialog:

Components of Edit Format Specification Dialog

Component	Description
Name	(required) Displays the current name of the format that you are editing. Changing the displayed value updates the name of the format. The Name field must contain a valid symbolic name entry.
Format Type	(required) The radio buttons below the label Format Type specify the data type of the format that you are editing. You can change the data type by selecting the radio button for the data type that you want to use.
Unspecified	No formatting activity will occur.
Symbol	<p>Entries into the item that you are editing must be symbolic.</p> <p>If you select Symbol, the Specify Legal Values button becomes enabled. Click the Specify Legal Values button if you want to open the Edit Legal Values dialog to specify a list of legal values that any UIL object to which this format is applied can contain. For information about this dialog, see Edit Legal Values Dialog.</p>

Components of Edit Format Specification Dialog

Component	Description
Text	<p>Entries made into the item that you are editing must be textual.</p> <p>If you select Text, the Text Options and Specify Legal Values buttons become enabled.</p> <ul style="list-style-type: none">• Click the Text Options button if you want to open the Text Formatting Options dialog, in which you can specify additional formatting for text. For information about this dialog, see Text Formatting Options Dialog.• Click the Specify Legal Values button if you want to open the Edit Legal Values dialog to specify a list of legal values that any UIL object to which this format is applied can contain. For information about this dialog, see Edit Legal Values Dialog.
Integer	<p>Entries made into objects to which this format applies must be whole numbers.</p> <p>If you select Integer, the Specify Legal Values button becomes enabled. Click the Specify Legal Values button if you want to open the Edit Legal Values dialog to specify a list of legal values that any UIL object to which this format is applied can contain. For information about this dialog, see Edit Legal Values Dialog.</p>

Components of Edit Format Specification Dialog

Component	Description
Quantity	<p>Entries made into objects to which this format applies must be quantitative. If you select Quantity, the Float Options and Specify Legal Values buttons become enabled.</p> <ul style="list-style-type: none"> • Click the Float Options button if you want to open the Float Formatting Options dialog to specify the format of floating point values that any UIL object to which this format is applied can represent. For information about this dialog, see Float Formatting Options Dialog. • Click the Specify Legal Values button if you want to open the Edit Legal Values dialog to specify a list of legal values that any UIL object to which this format is applied can contain. For information about this dialog, see Edit Legal Values Dialog.
Date	<p>Entries made into objects to which this format applies must be dates conforming to the currently specified date format.</p> <p>To specify a format for dates, click the Date/Time Options button, which becomes enabled when you select the Date format type. This opens the Date & Time Options dialog, which you use to specify additional formatting for date and time values. For information about this dialog, see Date & Time Options Dialog.</p>
Time	<p>Entries made into objects to which this format applies must conform to the time format that has been selected.</p> <p>When you select Time, the Date/Time Options button is enabled. Click the Date/Time Options button if you want to open the Date & Time Options dialog to specify additional formatting for date and time values. For information about this dialog, see Date & Time Options Dialog.</p>

Components of Edit Format Specification Dialog

Component	Description
Procedure Call	Entries made into objects to which this format applies are validated by the procedure that you specify in the Validation Method Name edit box.
Function Call	Entries made into objects to which this format applies are validated by the function that you specify in the Validation Method Name edit box.
Validation Method Name	<p>Indicates the current user-defined validation method specified for the format that you are editing. Enter the name of the procedure or function that you want to use to validate entries into any UIL object to which this format is applied.</p> <p>The Validation Method Name edit box is enabled only when the Procedure Call or Function Call button is selected.</p>
Validation Failure Handler	Indicates the user-defined validation failure method for the format that you are editing. Enter the name of the procedure or function that you want to use to use to handle failures to validate any UIL object to which this format is applied.
Default Value Handling	<p>Specifies a default value for the edit box. If any of these options are selected and no value is placed into the edit box by an update method, then the value will be placed according to the default declaration indicated here.</p> <p>The radio buttons below the label Default Value Handling are enabled for the Unspecified, Symbol, Text, Quantity, Integer, Date, and Time formats. When you specify a procedure call or function call to validate formats, default values are handled by that procedure or function.</p>
Unspecified	No default value is provided.

Components of Edit Format Specification Dialog

Component	Description
Use 1st value in value array	The default value is the value listed first in the list of legal values. You specify a list of legal values using the Edit Legal Values dialog.
Use type as basis	Provides a default value based on the data type selected under Format Type. The default values provided with the data types are: <ul style="list-style-type: none"> • G2 (for Symbol) • "G2" (for Text) • 0.0 (for Quantity) • 0 (for Integer) • Today's date in the currently specified format (for Date) • The current time in the currently specified format (for Time)
Use attribute name	The default value is the symbol used as the attribute name in the source object, if any.
Field Limiters	<p>The edit boxes below the label Field Limiters specify restrictions on the range and text length of data entries.</p> <p>The edit boxes are enabled only for the Unspecified, Symbol, Text, Quantity, Integer, Date, and Time formats. When you specify a procedure call or function call to validate formats, field limiters are specified by that procedure or function.</p>
Minimum Value	Contains the current minimum value accepted by the validation method.
Maximum Value	Contains the current maximum value accepted by the validation method.
Minimum Length	Contains the current minimum length of text accepted by the validation method.
Maximum Length	Contains the current maximum length of text accepted by the validation method.

Components of Edit Format Specification Dialog

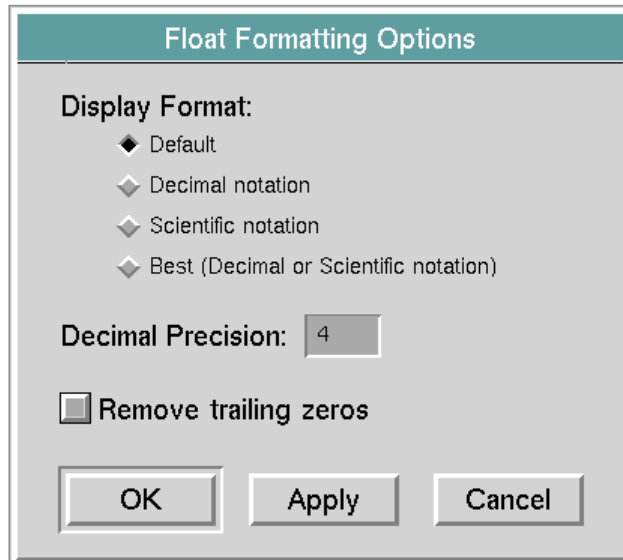
Component	Description
Allow none as valid option	If set to true, recognizes none as a valid entry.
Revert value on error	If the new value cannot be validated by the criteria specified in this format, replace the new value with the last successfully validated value.

Float Formatting Options Dialog

The Float Formatting Options dialog enables you to specify the style of any floating point value represented by the UIL object to which this format is applied.

To open the Float Formatting Options dialog, click the Float Options button in the Edit Format Specification dialog. This button is enabled only when the Quantity button is clicked.

The Float Formatting Options dialog looks like this:



The following table lists and describes the components of the Float Formatting Options dialog:

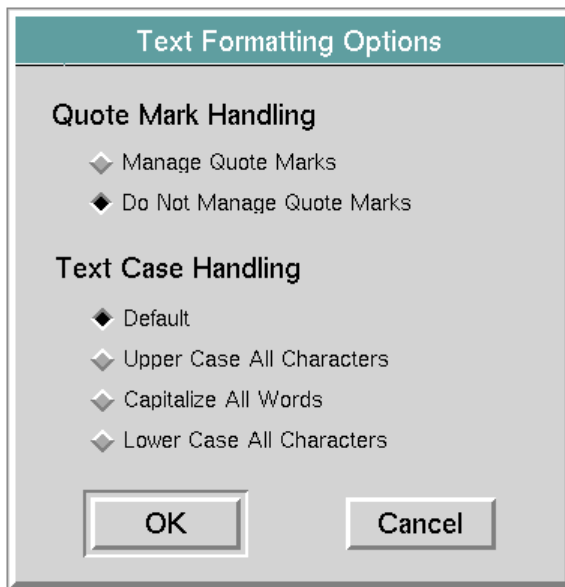
Components of Float Formatting Dialog

Component	Description
Display Format	The buttons below this label provide you with a choice of styles for representing the floating point value. One of these buttons is always selected.
Default	Format the data according to the default format specified by G2.
Decimal notation	Represent the floating point value in decimal notation.
Scientific notation	Represent the floating point value in scientific notation.
Best (Decimal or Scientific notation)	Allow GUIDE to determine the better display format for the data to which this format is applied.
Decimal Precision	Specify the decimal precision of the floating point value. This option is enabled only when the Decimal notation or Scientific notation button is selected.
Remove Trailing Zeros	Remove trailing zeros from the floating point value. This option is enabled only when the Decimal notation or Scientific notation button is selected.

Text Formatting Options Dialog

To open the Text Formatting Options dialog, go to the Edit Format Specification dialog and select the Text radio button. Then click the Text Options button.

The Text Formatting Options dialog looks like this:



The following table describes the components of the Text Formatting Options table:

Components of Text Formatting Options Dialog

Component	Description
Quote Mark Handling	The radio buttons below the label Quote Mark Handling indicate how quotation marks are handled by the format that you are editing.
Manage Quote Marks	Selecting this button ensures that quotation marks are not displayed in the edit box even when the data value contains them. If quotation marks are present or are required by the data in an object, they are replaced automatically before the value is concluded back to this object.
Do Not Manage Quote Marks	Selecting this button ensures that the data placed into the edit box is preserved exactly as it is found. Quotation marks are not added or subtracted from the data.
Text Case Handling	The radio buttons below the label Text Case Handling indicate how text case is handled by the format that you are editing.

Components of Text Formatting Options Dialog

Component	Description
Default	No alteration are made to the text before it is displayed in the edit box.
Upper Case All Characters	Every character in the string of text is capitalized before it is displayed in the edit box.
Capitalize All Words	Every word in the string of text is capitalized before it is displayed in the edit box.
Lower Case All Characters	Every character in the string of text is made lower case before it is displayed in the edit box.

Edit Legal Values Dialog

The Edit Legal Values dialog enables you to specify a list of legal values associated with the format that you are editing. GUIDE uses this list to validate the contents of objects to which the format is applied.

To open the Edit Legal Values dialog, click the Specify Legal Values button in the Edit Format Specification dialog. The Edit Legal Values dialog looks like this:



The following table describes the components of the Edit Legal Values dialog:

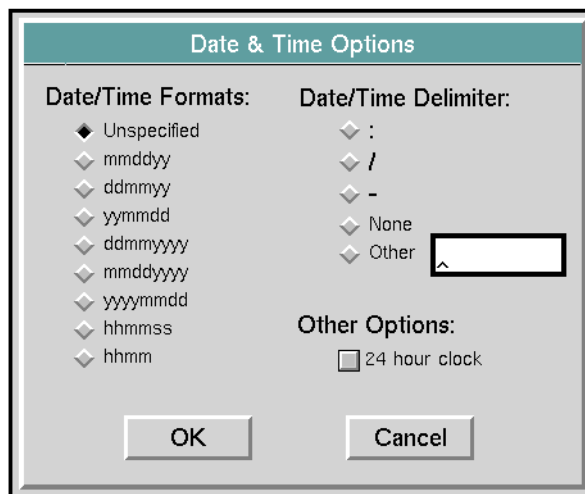
Components of Edit Legal Values Dialog

Component	Description
Legal Values	Lists the array of current legal values.
Remove Selection(s)	Deletes all selected messages from the Legal Values scroll area.
Enter new legal value	Enter new values into the Legal Values scroll area. When you finish entering a value, click Return. The value that you entered is added to the list of legal values in the Legal Values scroll area.

Date & Time Options Dialog

The Date & Time Options dialog enables you to specify additional formatting options for formats with the Date or Time format type, as specified in the Edit Format Specification dialog.

To open the Date & Time Options dialog, click the Date/Time Options button in the Edit Format Specification dialog. The Date & Time Options dialog looks like this:



The following table describes the components of the Date & Time Options dialog:

Components of Date & Time Options Dialog

Component	Description
Date/Time Formats	<p>The radio buttons below the label Date/Time Formats indicate the current date or time format for the format that you are editing.</p> <p>These formats are used with a delimiter that you select under Date/Time Delimiter. The delimiter is used to separate fields in the selected date or time format. For example, selecting the date and time format mmddy with a delimiter of / means that a valid entry must resemble 10/15/93.</p>
Unspecified	No format is enforced.
mmddy	Entries made into the edit box must resemble 05/31/94 or 053194 (no delimiter).
ddmmyy	Entries made into the edit box must resemble 31/05/94 or 310594 (no delimiter).
yymmdd	Entries made into the edit box must resemble 94/05/31 or 940531 (no delimiter).
ddmmyyyy	Entries made into the edit box must resemble 31/05/1994 or 31051994 (no delimiter).
mmddyyyy	Entries made into the edit box must resemble 05/31/1994 or 05311994 (no delimiter).
yyyymmdd	Entries made into the edit box must resemble 1994/05/31 or 19940531 (no delimiter).
hhmmss	Entries made into the edit box must resemble 10/25/30 or 102530 (no delimiter).
hhmm	Entries made into the edit box must resemble 9/30 .

Components of Date & Time Options Dialog

Component	Description
Date/Time Delimiter	<p>The radio buttons below the label Date/Time Delimiter specify the delimiter used with the date or time format that you select under Date/Time Formats above.</p> <p>The following examples illustrate the supported delimiters used with a value in mmddy format.</p> <p style="margin-left: 20px;">: 05:31:94</p> <p style="margin-left: 20px;">/ 05/31/94</p> <p style="margin-left: 20px;">- 05-31-94</p> <p style="margin-left: 20px;">None 053194</p> <p style="margin-left: 20px;">Other You can specify any alphanumeric character to use as the delimiter. Enter the character in the edit box to the right of the label Other.</p>
Other Options	
24 hour clock	<p>Entries made into the edit box must be in military format.</p> <p>Example: 17:45:10 (ten seconds past 5:45 pm)</p>

Specifying Source and Target Objects

Describes how to specify the objects from which the graphical objects on a dialog are updated and to which the graphical objects conclude their values.

Introduction **349**

Edit Source Object & Attribute Dialog **352**

Edit Target Object & Attribute Dialog **355**

Updating from and Concluding to Embedded Objects **359**



Introduction

Most UIL controls can send values to and receive values from other objects. For example, a user can enter values for the make, model, year, and color of a car into edit boxes in a dialog, and then send the values of these edit boxes to a G2 object holding the data for that car. The object can, in turn, send these values back to the edit boxes in the dialog for display and editing.

The object to which a UIL control sends a value is called a **target object**. The UIL control sends the value to a particular attribute of the target object, which is called a **target attribute**. A UIL control sends a value to the target attribute of its target object when a conclude method is run on the UIL control.

The object from which a UIL control receives a value is called a **source object**. The particular attribute of the source object from which this value comes is called the **source attribute**. A UIL control receives a value from the source attribute of its source object when an update method is run on the UIL control.

You can specify source and target objects and attributes for UIL controls using GUIDE editors, which are described in this section. The source and target objects

and attributes of UIL controls are specified in attributes of the UIL controls named `uil-event-source-object`, `uil-event-source-attribute`, `uil-event-target-attribute`, and `uil-event-target-object`.

Specifying Source and Target Objects

You can use the following kinds of objects as source objects for a UIL control:

- The G2 object that launched the dialog containing the UIL control. This object is known as the **initiating object** of the dialog.

The initiating object must be an instance of a user-defined class. The user-defined class must have at least one class-specific attribute whose value can be reflected in the state of the UIL control.

- The temporary storage object for the dialog that contains the UIL control that you are editing. For information about how to use temporary storage objects, see [Creating Temporary Storage Objects](#).
- Any named G2 object.

You can use the following kinds of objects as target objects:

- The object that launched that dialog containing the UIL control the you are editing.
- The temporary storage object for the dialog that contains the UIL control that you are editing.
- The destination object for the dialog that contains the UIL control that you are editing. This object is specified at runtime by a call to the UIL procedure `uil-set-destination-for-dialog`. You can specify only one destination object for a dialog. For information about `uil-set-destination-for-dialog`, see the *G2 GUIDE/UIL Procedures Reference Manual*.
- Any named G2 object.

For detailed descriptions of the different kinds of source and target objects that you can specify, see [Edit Source Object & Attribute Dialog](#) and [Edit Target Object & Attribute Dialog](#).

Note For some purposes, you may want to perform additional processing on the values when they are concluded to a target object or retrieved from a source object. For information about how to do this, see [Creating Temporary Storage Objects](#).

Source and Target Objects of Different UIL Controls

Except for text objects, all UIL controls that have values or state can have both source and target objects. Text objects can have only source objects. This is because text objects display read-only text and cannot send their values to other objects.

Push buttons have target objects, even though push buttons do not have values that can be concluded to or updated from object attributes. The target object of a push button is a dialog on which a set of actions is run whenever a user clicks on the push button.

For information about how to specify the source object and attribute, see [Edit Source Object & Attribute Dialog](#).

For information about how to specify the target object and attribute, see [Edit Target Object & Attribute Dialog](#).

Note UIL controls such as separators and borders do not have source or target objects, because these UIL controls do not have values that can be updated or concluded.

Target Objects for Push Buttons

Push buttons can have target objects, but do not have source objects.

The target object of a push button is always a dialog — this can be either the dialog that contains the push button, or some other dialog.

When a user clicks a push button, the actions associated with that push button are run on the dialog that is specified as that push button's target object. For example, if the actions on the push button include an action for concluding values, then all the UIL controls on the target dialog conclude their values when a user clicks the push button.

Unlike other UIL controls, push buttons do not send values to their target objects.

You do not specify source attributes for push buttons, because push buttons do not have values that can be derived from other objects.

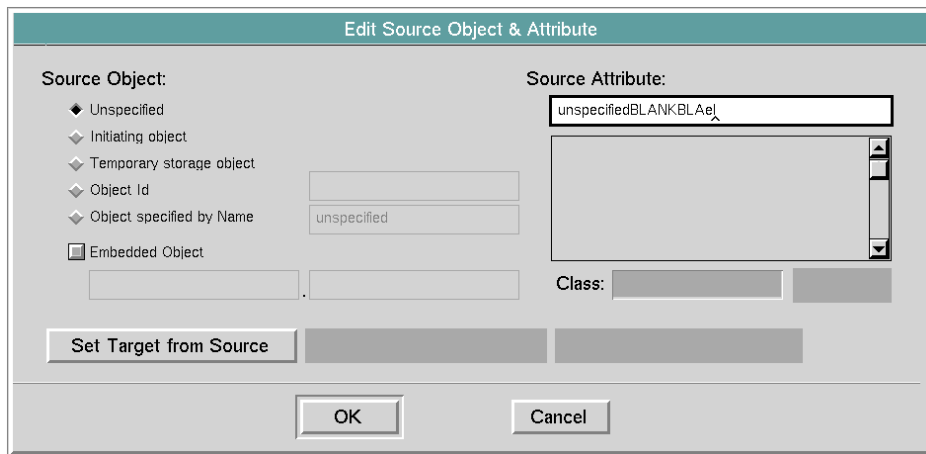
For information about how to specify the target object for a push button, see [System-Defined Actions for Dialog Processing](#).

Note If you use a push button to perform specialized processing through a user-defined callback procedure, rather than to run actions on a dialog, the push button does not require or use a target object.

Edit Source Object & Attribute Dialog

The Edit Source Object & Attribute dialog enables you to specify the source object and attribute for the UIL control that you are editing. When the update method is run on an object, the value used to update the UIL control comes from the attribute of the source object that you specify in the Edit Source Object & Attribute dialog.

To open the Edit Source Object & Attribute dialog, click the Edit Source Object push button on one of the following dialogs: Edit Toggle Button, Edit Radio Button, Edit Check Button, Edit Edit Box, Edit Message and Edit Text. The Edit Source Object & Attribute dialog looks like this:



The following table describes the components of the Edit Source Object & Attribute dialog:

Components of Edit Source Object & Attribute Dialog

Component	Description
Source Object	The radio buttons below the label Source Object specify the object used as the source object.
Unspecified	Indicates that the UIL control that you are editing has no source object.

Components of Edit Source Object & Attribute Dialog

Component	Description
Initiating Object	<p>Selecting this button indicates that the value of the UIL control that you are editing is to be updated with a value from the object initiating dialog activity.</p> <p>The class of the initiating object appears in the Class field.</p>
Temporary storage object	<p>Selecting this button indicates that the value of the UIL control that you are editing is to be updated with a value from the temporary storage object associated with the dialog or parent dialog. For information about how to use temporary storage objects, see Creating Temporary Storage Objects.</p> <p>The class of the temporary storage object appears in the Class field.</p>
Object Id	<p>Specify the source object by its numeric object ID. You can obtain the ID of a UIL object from its attribute table or from its graphical editor.</p>
Object specified by Name	<p>If you click this button, you must enter the name of the source object in the field to the right of the Object specified by Name button.</p> <p>The class of the source object that you specify appears in the Class field.</p>
Embedded Object	<p>Selecting this button indicates that the source object is an embedded object.</p> <p>For information about how to use embedded objects as source and target objects, see Updating from and Concluding to Embedded Objects.</p>

Components of Edit Source Object & Attribute Dialog

Component	Description
Source Attribute	<p>The field immediately below the label Source Attribute indicates the currently specified source attribute for the UIL control.</p> <p>You can enter the name of a class-specific attribute directly into this field. The attribute that you enter must be an attribute of the class specified in the Class field.</p> <p>Selecting an attribute from the scroll area automatically enters it in the edit box below the label Source Attribute.</p>
Class	<p>Displays the class of the source object.</p> <p>To display the class-specific attributes of this class, click the Update button. The attributes are listed in the scroll area immediately above the Class field.</p>
Update	<p>Clicking on this button causes the class-specific attributes of the class in the Class field to be displayed in the scroll area immediately above the Class field and Update button.</p> <p>You can specify a source attribute by selecting one of the attributes displayed in the scroll area.</p>
Set Target from Source	<p>Sets the target object and target attribute to be the same as the source object and source attribute that you select in this dialog.</p>

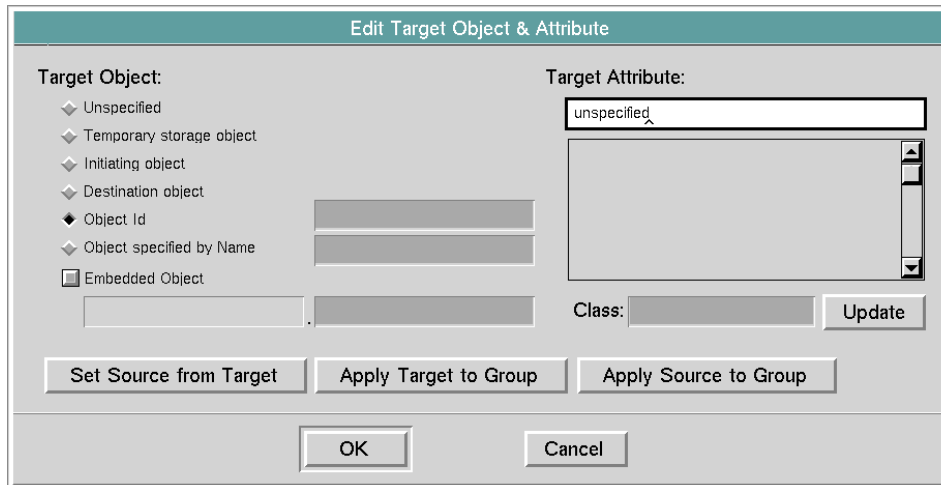
Components of Edit Source Object & Attribute Dialog

Component	Description
Apply Target to Group	<p>This button is enabled only when you are editing a radio button or check button.</p> <p>If you select Set Target from Source, you can click Apply Target to Group to assign the source object and source attribute specified in this dialog to be, in addition, the target object and target attribute for all the other buttons in the radio box or check box.</p>
Apply Source to Group	<p>This button is enabled only when you are editing a radio button or check button.</p> <p>Selecting Apply Source to Group assigns the specified source object and source attribute that you specify in this dialog to all the other buttons in the radio box or check box.</p>

Edit Target Object & Attribute Dialog

The Edit Target Object & Attribute dialog enables you to specify a target object and attribute for the UIL control that you are editing. When the conclude method for this UIL control is run, the current value of the UIL control is concluded into the attribute of the target object that you specify in the Edit Target Object & Attribute dialog.

To open the Edit Target Object & Attribute dialog, click the Edit Target Object push button on one of the following dialogs: Edit Toggle Button, Edit Radio Button, Edit Check Button, Edit Edit Box, Edit Message or Edit Text. The Edit Target Object & Attribute dialog looks like this:



The following tables describes the components of the Edit Target Object & Attribute dialog:

Components of Edit Target Object & Attribute Dialog

Component	Description
Target Object	(required) The radio buttons below the label Target Object indicate the object to be used as the target object when the conclude method is run on the UIL control that you are editing.
Unspecified	Indicates that the UIL control that you are editing has no target object.
Temporary storage object	Indicates that the value of the UIL control that you are editing is to be concluded into the temporary storage object associated with the dialog or parent dialog when the conclude method is run. For information about how to use temporary storage objects, see Creating Temporary Storage Objects .

Components of Edit Target Object & Attribute Dialog

Component	Description
Initiating Object	Indicates that the value of the UIL control that you are editing is to be concluded into the object initiating dialog activity when the conclude method is run. The initiating object is found by traversing up the chain of cascaded dialogs. The initiating object can be any G2 item.
Destination Object	<p>Indicates that the value of the UIL control that you are editing is to be concluded into a destination object.</p> <p>When you select this option, the <code>uil-event-target-object</code> attribute is set to <code>destination-object</code>. You must then use the <code>uil-set-destination-for-dialog</code> procedure to specify a destination object for the dialog at run time. You can also use the <code>uil-find-destination-for-dialog</code> procedure to return the destination object. For information about these procedures, see the <i>G2 GUIDE/UIL Procedures Reference Manual</i>.</p>
Object Id	Specify the source object by its numeric object ID. You can obtain the ID of a UIL object from its attribute table or from its graphical editor.
Object specified by Name	Indicates that the value of the UIL control that you are editing is to be concluded into the object with the given name. If this radio button is selected, a valid name for an object must be specified in the edit box to the right of this radio button. The named object can be any G2 item.
Embedded Object	<p>Selecting this button indicates that the target object is an embedded object.</p> <p>For information about how to use embedded objects as source and target objects, see Updating from and Concluding to Embedded Objects.</p>

Components of Edit Target Object & Attribute Dialog

Component	Description
Target Attribute	<p>The field immediately below the label Target Attribute indicates the currently specified target attribute for the UIL control.</p> <p>You can enter the name of a class-specific attribute directly into this field. The attribute that you enter must be an attribute of the class specified in the Class field.</p> <p>Selecting an attribute from the scroll area automatically enters it in the edit box below the label Target Attribute.</p>
Class	<p>Displays the class of the target object.</p> <p>To display the class-specific attributes of this class, click the Update button. The attributes are listed in the scroll area immediately above the Class field.</p>
Update	<p>Clicking on this button causes the class-specific attributes of the class in the Class field to be displayed in the scroll area immediately above the Class field and Update button.</p> <p>You can specify a target attribute by selecting one of the attributes displayed in the scroll area.</p>
Set Source from Target	<p>Sets the source object and source attribute to be the same as the target object and target attribute that you select in this dialog.</p>

Components of Edit Target Object & Attribute Dialog

Component	Description
Apply Target to Group	<p>This button is enabled only when you are editing a radio button or check button.</p> <p>Selecting Apply Target to Group assigns the specified target object and target attribute that you specify to all the other buttons in the radio box or check box.</p>
Apply Source to Group	<p>This button is enabled only when you are editing a radio button or check button.</p> <p>If you select Set Source from Target, you can click Apply Source to Group to assign the target object and target attribute specified in this dialog to be, in addition, the source object and source attribute for all the other buttons in the radio box or check box.</p>

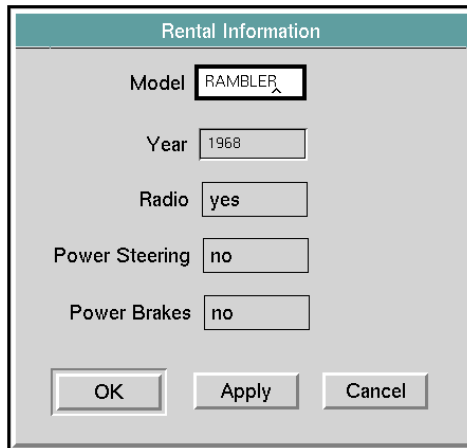
Updating from and Concluding to Embedded Objects

An **embedded object** is an object that is referenced or contained by an attribute of another object.

You can specify an embedded object as the source object or target object of a UIL control. If you do this, you must specify an attribute of the embedded object as the source attribute or target attribute of the UIL control.

The following example shows how UIL controls in a dialog can use an embedded object as their source object.

Suppose that an automobile-rental application uses an object named Rambler, which is an instance of a class named Car. A dialog named Rental Information displays the information stored in the object named Rambler. In this dialog, the different pieces of information stored in Rambler are represented by edit boxes:



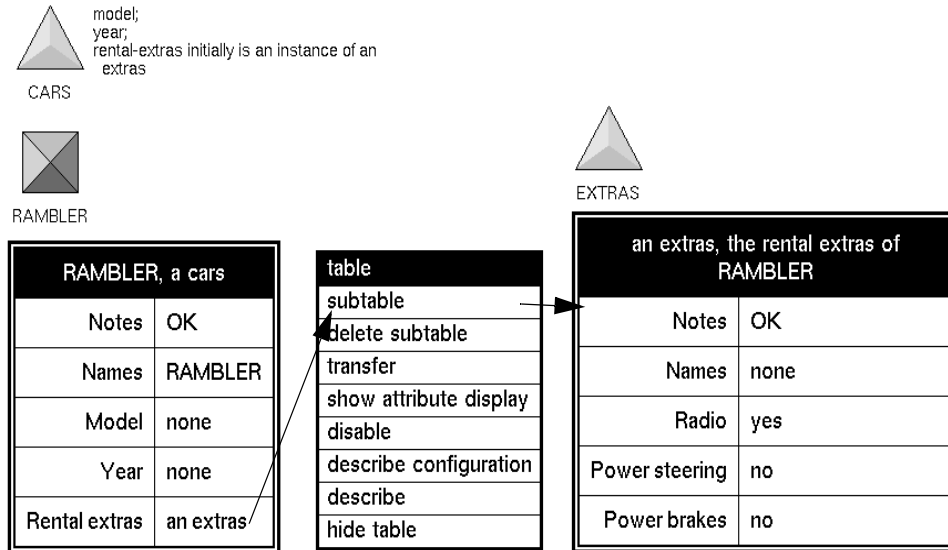
The image shows a dialog box titled "Rental Information". It contains five text input fields (edit boxes) arranged vertically. The first field is labeled "Model" and contains the text "RAMBLER". The second field is labeled "Year" and contains "1968". The third field is labeled "Radio" and contains "yes". The fourth field is labeled "Power Steering" and contains "no". The fifth field is labeled "Power Brakes" and contains "no". At the bottom of the dialog, there are three buttons: "OK", "Apply", and "Cancel".

The edit boxes labeled Model and Year display the values of attributes of Rambler.

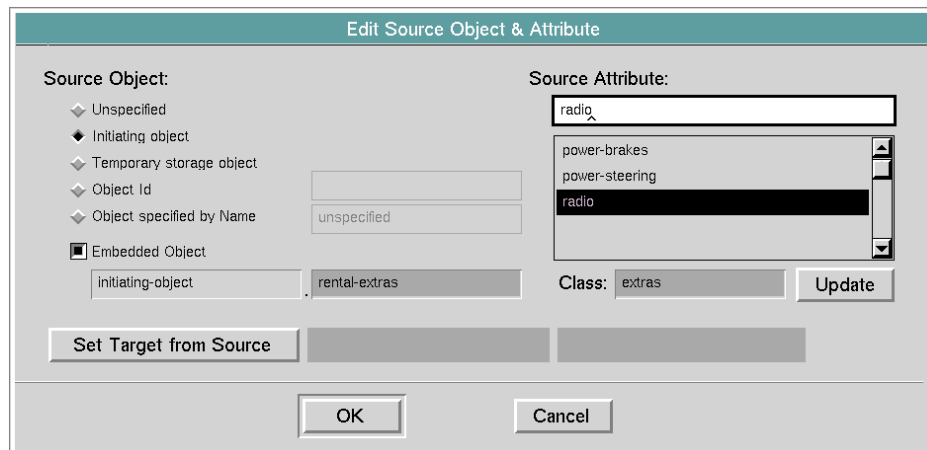
However, the edit boxes labeled Radio, Power Steering, and Power Brakes display attribute values of an embedded object, rather than attribute values of rambler itself. This embedded object is embedded in an attribute of rambler named rental-extras.

The embedded object is an instance of a class named extras. The class extras has attributes that represent the radio, power steering, and power brakes rental options. These attributes are the source attributes for the edit boxes labeled Radio, Power Steering, and Power Brakes.

In the attribute table for rambler, the rental-extras attribute contains this instance of extras as an embedded object. To view or edit the attributes of this embedded object, open the subtable for the rental-extras attribute:



The following figure illustrates how to specify an embedded source object for the Radio edit box in the Rental Information dialog:



The following example illustrates in detail how to use an embedded object as the source object or target object of UIL controls on a dialog.

Note In the following example, you must use the GUIDE palette to add the edit boxes representing attributes of the embedded object (Radio, Power Steering, and Power Brakes) to the Rental Information dialog. For this reason, you may want to create the entire dialog using the GUIDE palette, rather than using the GUIDE Dialog Generator.

The GUIDE Dialog Generator does not generate individual UIL controls to represent attributes of an embedded object. Instead, the GUIDE Dialog Generator generates a push button for the embedded object. To use this push button, you must first generate a dialog for the class of the embedded object. You can then use the push button to launch the dialog for the embedded object.

To specify an attribute of an embedded object as the source attribute:

- 1** Open the object editor for the UIL control and click the Source Object button to open the Edit Source Object & Attribute dialog.
- 2** Specify the source object and the attribute of the source object that contains the embedded object. To do this:
 - a** In the Edit Source Object & Attribute dialog, specify the source object of the UIL control. The source object must be the object that contains the embedded object in one of its attributes.

For example, if the source object is the initiating object, click the Initiating object button.
 - b** Click the Embedded Object button. The name of the source object that you specified appears in the first field to the right of the Embedded Object button.
 - c** In the second field to the right of the Embedded Object button, enter the attribute of the source object that contains the embedded object.

In the example above, the embedded object is contained in the attribute of cars named **rental-extras**. Thus, you enter **rental-extras** in the second field to the right of the Embedded Object button.
- 3** Specify the attribute of the embedded object that you want to use as the source attribute of the UIL control. To do this:
 - a** Enter the class of the embedded object in the field to the right of the label Class.

In this example, the embedded object is an instance of the class **extras**. Thus, enter **extras** in the field to the right of the label Class.
 - b** Click the Update button. This displays the attributes of the **extras** class in the scroll area above the Update button.

- c In the scroll area, select the attribute of the embedded object that you want to use as the source attribute of the UIL control.

In this example, select **Radio** as the attribute of the embedded object that you want to use as the source attribute.

- 4 Click OK in the Edit Source Object & Attribute dialog.
- 5 Click OK in the object editor for the UIL control.

When you finish specifying attributes of an embedded object as a source or target attributes for a UIL control, you can examine the attribute table of the UIL control to verify that you have specified these attributes correctly. The following figure illustrates the attribute table of the Radio edit box when it is edited to specify the object in the `rental-extras` attribute of `rambler` as the source and target object:

an uil-edit-box-medium	
Notes	UIL-EDIT-BOX-MEDIUM-XXX-21: OK
Names	none
Id	""
Configuration	uil-text-only-edit-box-configuration
Uil configuration method	uil-configure-grobj-method
Uil maximum characters to display	unlimited
Uil size of method	uil-size-of-grobj-method
Uil size	medium
User data	none
Uil clone method	uil-clone-grobj-method
Uil conclude value immediately	false
Uil event target object	initiating-object.rental-extras
Uil event target attribute	radio
Uil event source object	initiating-object.rental-extras
Uil event source attribute	radio

The attribute value specified for `uil-event-target-object` or `uil-event-source-object` can include only one level of embedding. GUIDE does *not* support the third level of embedding represented, for example, by `radio-type` in the following attribute specification:

```
initiating-object.rental-extras.radio-type
```

To enable users to view and edit the attributes of two or more levels of embedded objects, you can either create a system of cascaded dialogs, or create methods that update and conclude values from the appropriate objects.

Creating Temporary Storage Objects

Describes how to create and use temporary storage objects, which you can use when you process data while it is being updated into or concluded from a dialog.

Introduction **365**

How Temporary Storage Objects Work **366**



Introduction

For some purposes, an application may need to store data in one set of terms while enabling users to view and edit the data in a different set of terms.

For example, a G2 object may store certain system parameters as numbers. Each parameter value has a meaning that users need to understand. Instead of requiring users to view and edit the parameters in numeric format, you can create a dialog that uses a descriptive text string to represent each parameter value.

Similarly, you might want to use a dialog to display the average value of attributes of several different objects. Your application must recalculate the average each time a user requests an update of the display in the dialog.

How Temporary Storage Objects Work

Temporary storage objects make it possible for you to use dialogs as described in the preceding paragraphs. A temporary storage object serves as a buffer during update and conclude actions on the dialog, in the following ways:

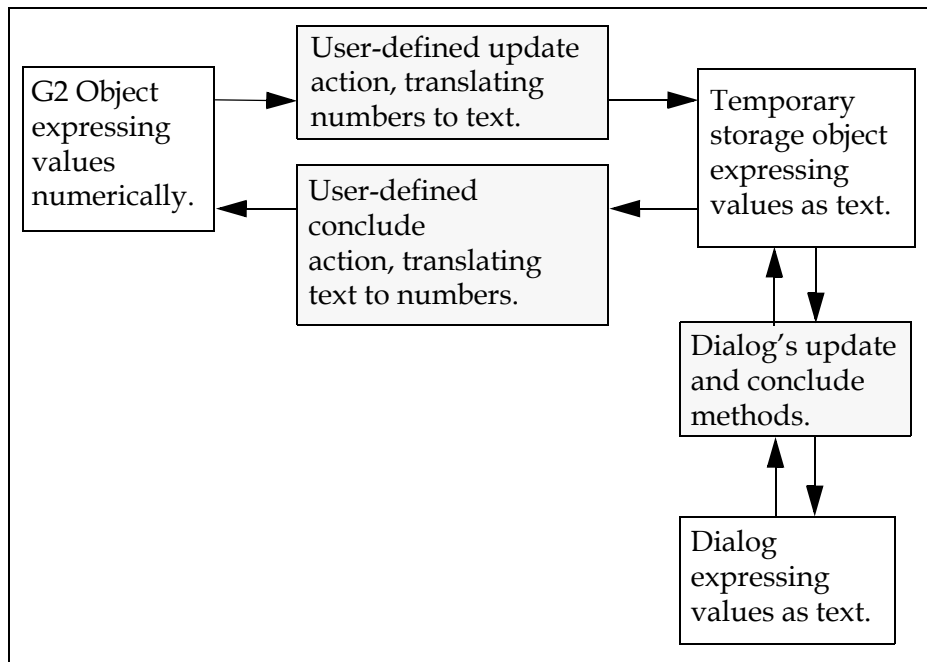
- When an update action is run on the dialog, a user-defined update action first translates the attribute values of a G2 object or objects into the terms used by the dialog, and updates the temporary storage object with these translated values.

The dialog's update method then updates the dialog with the translated values in the temporary storage object. This update method is referenced in the dialog's `uil-update-method` attribute.

- When a conclude action is run on the dialog, the dialog's values are first concluded into the temporary storage object by the dialog's conclude method. This conclude method is referenced in the dialog's `uil-conclude-method` attribute.

A user-defined conclude action then translates the values back into their original terms and concludes them to the G2 object or objects.

The following figure illustrates one possible use of a temporary storage object as a buffer during update and conclude actions on a dialog:



Note Temporary storage objects can be updated from and conclude values to any number of different objects.

How Temporary Storage Objects Are Created

Temporary storage objects are instances of user-defined object definitions. You associate the object definition with any dialog that needs to use temporary storage objects of that particular class. When the dialog is reserved, a temporary storage object of the associated class is automatically created.

GUIDE automatically links the temporary storage object to the dialog with the dynamic relation `the-uil-temporary-storage-object-of`. This relation is maintained until the dialog is released by the `uil-release-dialog` action.

Steps for Defining a Temporary Storage Object for a Dialog

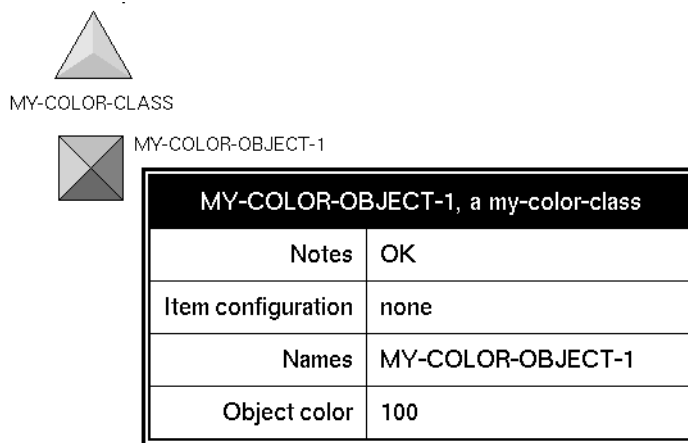
To use a temporary storage object with a dialog:

- 1** Create a class definition for the temporary storage object. The class definition should include a class specific attribute for each UIL object on the dialog whose value will be updated from or concluded to the temporary storage object.
- 2** Specify the name of this class in the Dialog Options dialog, which you can open by clicking on the More Options button in the Edit Dialog dialog.
- 3** For the source and target attribute of each UIL object on the dialog, specify a class-specific attribute of the temporary storage object. This enables the values of the UIL objects to be updated from and concluded to the temporary storage object.
- 4** Create an update action that updates the temporary storage object with attribute values of the G2 object or objects. The update action must translate these attribute values into the terms that the dialog uses.
- 5** Include your user-defined update procedure among the actions run by the push button, action button, user menu choice, or procedure that starts the dialog. Place the user-defined update action before the update method.
- 6** Create a conclude action that writes the attribute values of the temporary storage object back into the G2 object or objects. The conclude action must translate the attribute values of the temporary storage object back into their original terms.
- 7** Include your user-defined conclude procedure among the actions run by the push button on the dialog that concludes the values in the dialog. Place the user-defined conclude action after the conclude method.

Note You can access a working example of a temporary storage object through the GUIDE Examples workspace. To open this workspace, click UIL Examples in the GUIDE Help dialog.

The following section illustrates the steps that you must follow to use a temporary storage object.

For example, suppose that an application uses objects of a class named `my-color-class` to store the numeric color codes 100, 101, and 102, representing red, white, and blue respectively. The following figure illustrates an object of this class that stores the color code 100 (red) in an attribute named `object-color`:



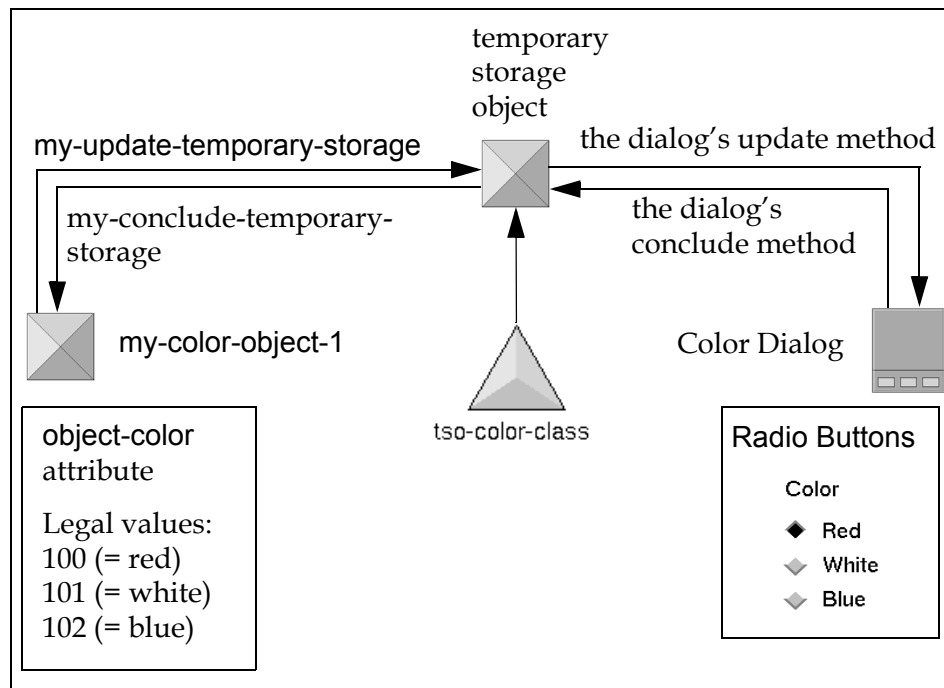
Users can view and edit the `object-color` attribute of the objects of the class `my-color-class`, using the following dialog:



The radio buttons in this dialog have the values Red, White, and Blue, which correspond to the possible values of the `object-color` attribute. To make it possible for users to edit the numerical values of the `object-color` attribute using the radio

buttons, your application must first translate the numeric values into the textual on values of the radio buttons.

The following figure illustrates how your application can do this:



Creating this Example

The following sections describe how to implement the application shown in the figure above.

Create a Class Definition for the Temporary Storage Object

To create the class definition `tso-color-class` for the temporary storage object used in this example, select:

KB Workspace > New Definition > object-definition

Open the table of the new class definition. In the table, specify a name, a direct superior class, and a class specific attribute named `my-color`, as follows:

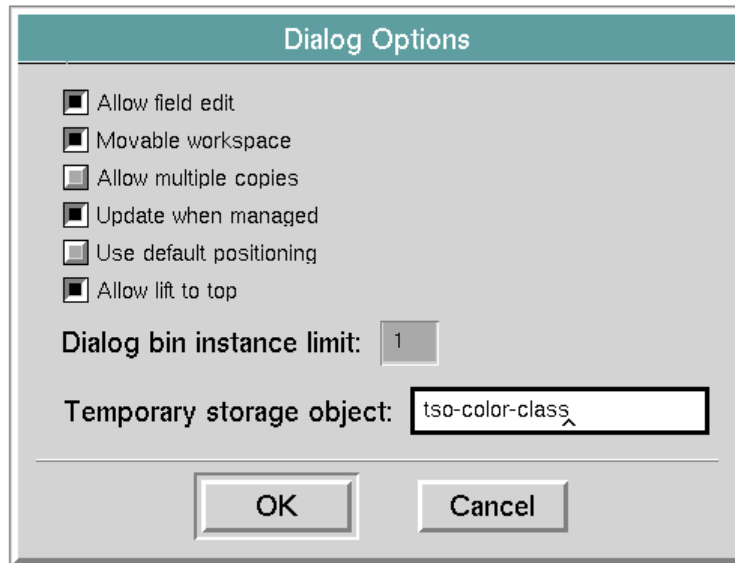
TSO-COLOR-CLASS, an object-definition	
Notes	OK
Authors	jfb (5 Jul 1995 2:51 p.m.)
Class name	tso-color-class
Direct superior classes	object
Class specific attributes	my-color initially is unspecified
Instance configuration	none
Change	none

Specify this New Class Definition as the Class of the Dialog's Temporary Storage Object

To specify this new class definition as the class from which temporary storage objects will be created for this dialog:

- 1 Choose `edit dialog` from the menu of the Color Dialog dialog to open the Edit Dialog dialog.
- 2 Click More Options in the Edit Dialog dialog to open the Dialog Options dialog.

- 3 In the Dialog Options dialog, specify the name of the user-defined class for temporary storage objects. For example:



When the dialog Color Dialog is reserved, an instance of the class `tso-color-class` is created automatically. This instance is used as the temporary storage object.

- 4 Click the OK buttons in the Dialog Options and Edit Dialog dialogs.

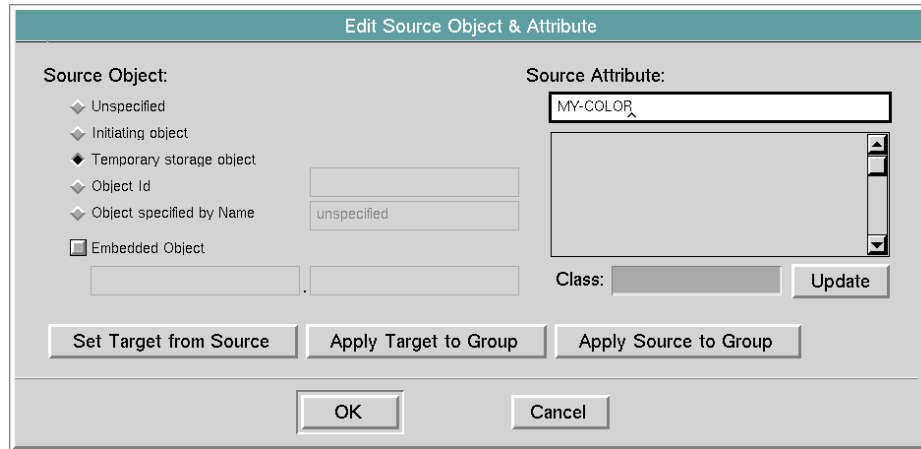
Set the Source and Target Attributes of the UIL Objects on the Dialog

In this example, all the radio buttons on the Color Dialog dialog use the `my-color` attribute of the temporary storage object as their source object and target object.

To specify the `my-color` attribute as the source and target attribute of the buttons:

- 1 Click the radio button labeled Red and select edit radio button from its menu to open the Edit Radio Button dialog.
- 2 In the Edit Radio Button dialog, click the Source Object button to open the Edit Source Object & Attribute dialog.

- 3 In the Edit Source Object & Attribute dialog, select the Temporary Storage object option under the heading Source Object, and enter my-color in the edit box under the heading Source Attribute:



These settings specify that the temporary storage object associated with Color Dialog is the source object for the Red button, and that the my-color attribute of the temporary storage object is the source attribute of the Red button.

- 4 Click the Set Target from Source button in the Edit Source Object & Attribute dialog. This makes my-color the target attribute of the Red button.
- 5 Click the Apply Target to Group button. This makes my-color the target attribute of all the buttons in the same group as the Red button.
- 6 Click the Apply Source to Group button. This makes my-color the source attribute of all the buttons in the same group as the Red button.
- 7 Click the OK buttons in the Edit Source Object & Attribute and Edit Radio Button dialogs.

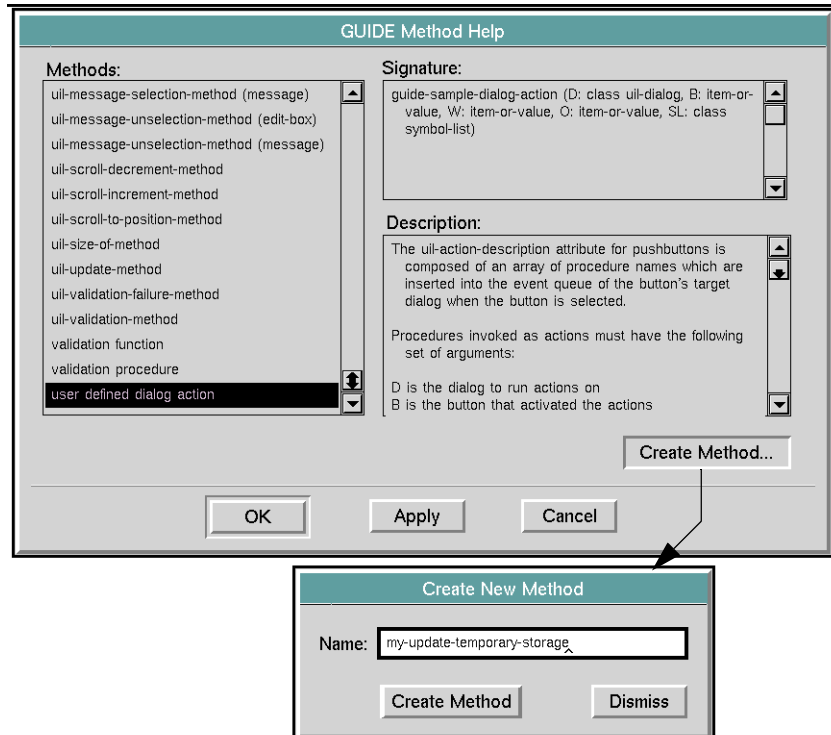
Create a User-Defined Update Action for the Temporary Storage Object

In this example, a user-defined update action named my-update-temporary-storage translates the numeric values 100, 101, and 102 into the corresponding text values used in the dialog, red, white, and blue. It then updates the temporary storage object with the translated values.

To create an update action:


- 1 Click the question mark (?) button in the GUIDE palette to open the GUIDE Help dialog.
- 2 In the GUIDE Help dialog, click the button labeled UIL Methods, Actions, and Callbacks to open the GUIDE Method Help dialog.

- 3 In the Methods column of the GUIDE Method Help dialog, select user defined dialog action in the Methods column. The required argument signature for actions appears in the Signature scroll area. A description of actions appears in the Description scroll area.
- 4 Click the Create Method button. This opens the Create New Method dialog.
- 5 In the Create New Method dialog, enter the name of the update action that you want to create. For example:



When the icon for the new action appears, select **edit** from its menu, and enter the procedure code. The following figure illustrates the code of the user-defined action `my-update-temporary-storage`:

MY-UPDATE-TEMPORARY-STORAGE



```

my-update-temporary-storage (D: class uil-
dialog, B: item-or-value, W: item-or-value, O:
item-or-value, SL: class symbol-list)
OBJ: item-or-value;

begin
  if (the object that is the uil-temporary-
storage-object-of D exists) then
    begin
      OBJ = the object that is the uil-
temporary-storage-object-of D;
      if (OBJ is an item) then
        begin
          if (the object-color of O = 100)
            then conclude that the my-color
of OBJ = the symbol red
          else if (the object-color of O =
101) then conclude that the my-
color of OBJ = the symbol white
          else if (the object-color of O =
102) then conclude that the my-
color of OBJ = the symbol blue;
        end;
      end;
    end
  end
end

```

This procedure:

- Finds the temporary storage object associated with the dialog.
- Translates the numeric value (100, 101, or 102) of the `object-color` attribute of a G2 object into a corresponding text string ("red", "white", and "blue", respectively).
- Updates the `my-color` attribute of the temporary storage object with this text string.

Include the User-Defined Update Procedure among the Actions Run on the Dialog When the Dialog Is Started

You must include the `my-update-temporary-storage` procedure among the actions that are run on the dialog when the dialog is started. You can start a dialog from an action button, a procedure, a push button in another dialog, or a user-menu choice.

The following steps illustrate how to include `my-update-temporary-storage` among the actions that are run on the dialog when the dialog is started from a user-menu choice:

- 1 Create a user menu choice named `start color dialog`.

To do this, select:

`KB Workspace > New Definition > user-menu-choice`

- 2 Open the table of the new user menu choice and edit the table to specify the name, label, applicable class, and action, as follows:

START-COLOR-DIALOG, an user-menu-choice	
Notes	OK
Authors	jfb (9 Aug 1995 9:58 a.m.)
Item configuration	none
Names	START-COLOR-DIALOG
Label	color-information...
Applicable class	my-color-class
Condition	none
Action	start uil-start-dialog-processing("Color Information", the item, this window, setup-color-dialog-actions, true, true, refocus-color-dialog-actions, false)
Action priority	2

The `start` statement in the `action` attribute of the user menu choice invokes the procedure `uil-start-dialog-processing` to start the dialog, or to update it if it is currently displayed. The procedure references the dialog by its ID, `Color Information`. The `uil-start-dialog-processing` procedure references two actions lists, `setup-color-dialog-actions` and `refocus-color-dialog-actions`.

The call to `uil-start-dialog-processing` runs the actions in `setup-color-dialog-actions` when it starts the dialog. This action description array includes the following actions:

- `my-update-temporary-storage` (the user-defined update procedure)
- `uil-call-update-method` (the system-defined update method)
- `uil-simulate-play-mode`
- `uil-show-managed-dialog`

The call to `uil-start-dialog-processing` also runs the actions in `refocus-color-dialog-actions` when it updates a dialog that is currently displayed. This action description array includes the following actions:


- `my-update-temporary-storage` (the user-defined update procedure)
- `uil-call-update-method` (the system-defined update method)

For information about how to create action description arrays, see [Controlling Dialogs with Actions](#). For information about how to start dialogs, see [Launching Dialogs](#).

Create a User-Defined Conclude Action for the Temporary Storage Object

In this example, a user-defined conclude procedure translates the text values "red", "white", and "blue" into their original numeric terms 100, 101, and 102.

The following figure illustrates the user-defined method `my-conclude-temporary-storage`:

```
MY-CONCLUDE-TEMPORARY-STORAGE
 my-conclude-temporary-storage (D: class uil-
  dialog, B: item-or-value, W: item-or-value, O:
  item-or-value, SL: class symbol-list)
  OBJ: item-or-value;

begin
  if (the object that is the-uil-temporary-
    storage-object-of D exists) then
    begin
      OBJ = the object that is the-uil-
        temporary-storage-object-of D;
      if (OBJ is an item) then
        begin
          if (the my-color of OBJ = the
            symbol red) then
            conclude that the object-color
              of O = 100
          else if (the my-color of OBJ = the
            symbol white) then
            conclude that the object-color
              of O = 101
          else if (the my-color of OBJ = the
            symbol blue) then
            conclude that the object-color
              of O = 102;
        end;
      end;
    end;
end
```

This procedure:

- Finds the temporary storage object associated with the dialog.
- Translates the text string ("red", "white", or "blue") in the `object-color` attribute of the temporary storage object into the corresponding numeric value (100, 101, or 102, respectively).
- Concludes this numeric value into the `object-color` attribute of the target object of the dialog. You specify the target object in the following step.

Include the User-Defined Conclude Procedure among the Actions Run When the Dialog's Values are Concluded

To do this, edit the push button or buttons in Color Dialog that you want to use to conclude the values in the dialog.

The OK and Apply buttons on a dialog are commonly used to conclude the values in the dialog. By default, OK and Apply buttons include `uil-call-conclude` method among their actions.

You can add `my-conclude-temporary-storage` to the OK and Apply buttons using the Customize Dialog Actions dialog.

Note You must put `my-conclude-temporary-storage` or any other user-defined conclude action *after* the call to `uil-conclude-method`.

For information about how to use the Customize Dialog Actions dialog, see [Customize Dialog Actions Dialog](#).

Add an Action to Delete the Temporary Storage Object

It is good practice to delete the temporary storage object when you dismiss the dialog. To do this, add the system-defined action `uil-delete-temporary-storage-object` to the button that dismisses the dialog, before the `uil-release-dialog` action.

Methods, Actions, and Callbacks

Describes how to create and use UIL methods, actions, and callbacks.

Introduction **379**

UIL Methods **380**

How UIL Methods Work **382**

UIL Actions **386**

UIL Callbacks **386**

Creating Methods, Actions, and Callbacks **387**



Introduction

GUIDE/UIL provides an extensive set of system-defined procedures that make it possible to create and use a GUIDE user interface. There are three kinds of procedures:

- UIL methods, which perform operations required by developers, such as cloning and deleting objects, or operations required by users, such as opening and closing dialogs, and updating and concluding their values.
- Actions, which perform run time operations on dialogs, such as opening or closing them, and updating or concluding their values.
- Callbacks, which are invoked whenever a user clicks on a button.

GUIDE/UIL provides an extensive set of system-defined methods, actions, and callbacks. When you create a user interface, GUIDE automatically ensures that the interface references the appropriate system-defined procedure for each common

operation. You can create and use a GUIDE user interface without modifying the set of methods, actions, and callbacks that the interface references by default.

However, if your user interface needs to perform specialized operations, you can create customized methods, actions, and callbacks to perform these operations, and then edit the interface to reference these procedures in place of the default system-defined procedures.

The following sections describe how you can create and use customized UIL methods, actions, or callbacks.

UIL Methods

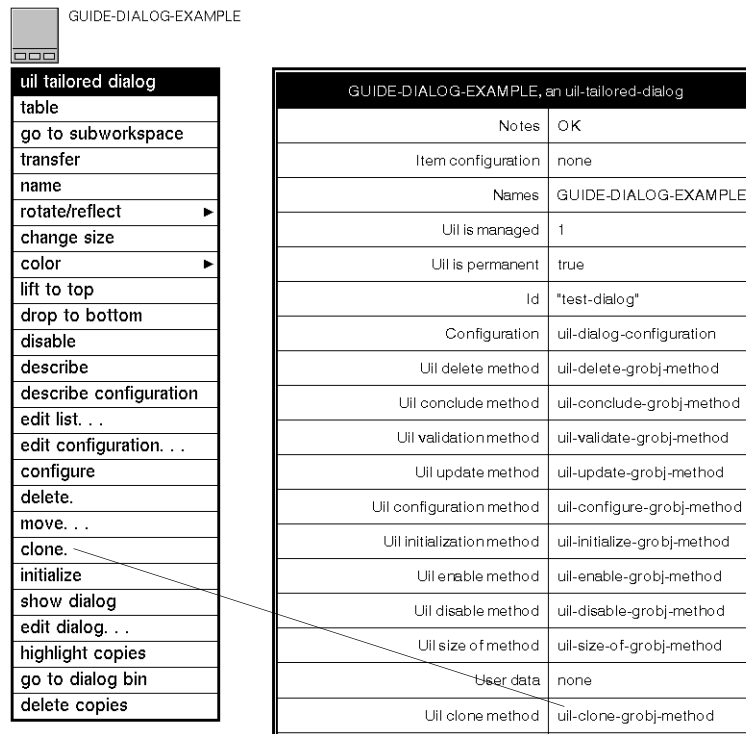
Every dialog and UIL control has attributes that correspond to the common operations that can be performed on the object. For example, dialogs have attributes named `uil-delete-method`, `uil-conclude-method`, `uil-update-method`, `uil-validation-method`, `uil-clone-method`, and so on. Edit boxes have attributes named `uil-initialization-method`, `uil-enable-method`, `uil-disable-method`, and so on.

Each of these attributes references the method responsible for performing the corresponding operation. When an application developer or user initiates one of these operations, the method responsible for performing that operation is run.

For example, when you select the `clone` operation from the menu of a UIL object, the procedure specified in the `uil-clone-method` attribute of that object is run. The method that is specified by default in this attribute, `uil-clone-groobj-method`, dispatches a procedure, `uil-clone-groobj`, that carries out the `clone` operation.

To use a customized method, edit the attribute for the corresponding operation to reference the customized method. For example, to use a customized clone method on an object, reference the name of that method from the `uil-clone-method` attribute of that object.

The following figure illustrates the clone. menu choice of a dialog and the uil-clone-method attribute in the table of that dialog:



UIL methods are designed to perform the same operation on UIL objects of different classes. For example, uil-clone-grobj-method is the default method for cloning all classes of UIL objects that can be cloned.

UIL Methods for Application Development

Some UIL methods perform operations that you need in order to construct a user interface, such as cloning, deleting, enabling, and disabling UIL objects.

UIL Methods for Runtime Operations

Some UIL methods are run in response to actions by the user, such as clicking a push button or tabbing to an edit box. The user operations performed by methods include:

- Concluding a value in a UIL object to a destination in the application, or updating a UIL object with a value retrieved from somewhere else in the application.
- Starting and stopping the editor for edit boxes.

- Validating edits that users make to contents of edit boxes, using validation criteria that you specify.
- Initializing UIL objects to a default, initial state.

Similarly, when a user clicks an OK push button and that button includes an action for concluding the values of all UIL objects on a dialog, the `conclude` method specified in the `uil-conclude-method` attribute of each UIL object is run. The result is that the values in these UIL objects are concluded to their target objects.

How UIL Methods Work

When an application developer or user requests an operation, a UIL method is invoked as the handler for that operation. UIL handler methods perform operations in the following way:

- 1 A user or application developer requests an operation on a UIL object. This executes the UIL handler method for that operation.

For example, if an application developer chooses `delete.` from the menu of an object, the handler for that operation, `uil-delete-grobj-method`, is run.

- 2 The handler examines the attribute of the UIL object that corresponds to the requested operation.

For example, the handler for delete operations, `uil-delete-grobj-method`, examines the current value of the `uil-delete-method` attribute of the scroll area.

- 3 If the attribute for the requested operation references the handler itself, the handler calls a system-defined UIL procedure to perform the requested operation on the particular object.

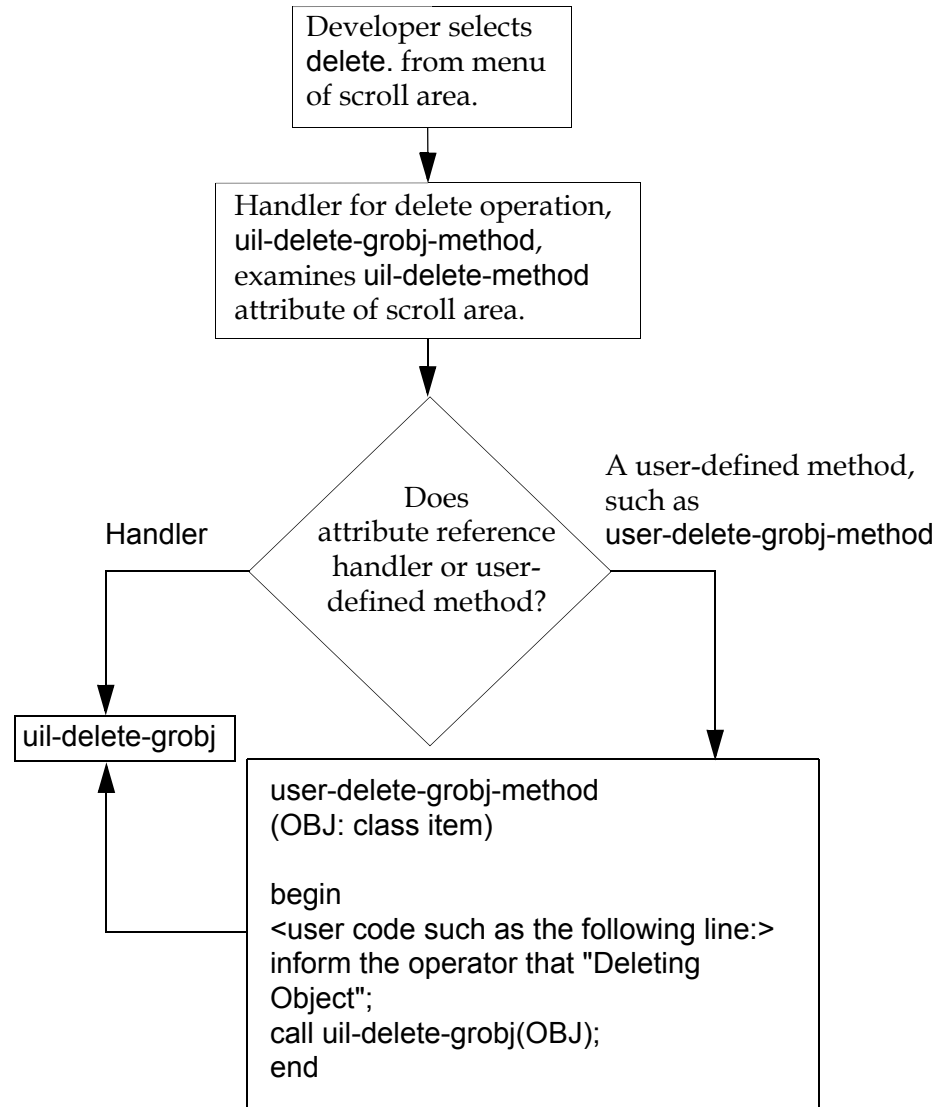
For example, the handler `uil-delete-grobj-method` calls the UIL procedure `uil-delete-grobj`.

- 4 However, if the attribute for the requested operation references a user-defined method, the handler runs the user-defined method. The user-defined method can do the following:

- Executes user code to perform the desired customized operations on the object.
- (optional) Calls a system-defined UIL procedure to perform the requested operation on the object.

For example, a user-defined method for deleting an object can first post a message to the message board to inform the operator that the object is being deleted. The user-defined method then calls the system-defined UIL procedure `uil-delete-grobj` to delete the object.

The following figure illustrates how a scroll area is deleted when the `uil-delete-method` attribute references the UIL handler method `uil-delete-grobj-method`, and when it references a user-defined method, `user-delete-grobj-method`.



Caution Never call the handler method for an operation, such as `uil-delete-grobj-method`, from a user-defined method that you reference from the attribute for that operation. Calling the handler from the user-defined method results in an infinite loop.

Object Attributes that Reference UIL Methods

The following table lists the attributes of UIL objects that reference methods:

Method Attributes of UIL Objects

Attribute	Classes with this Attribute	Operations Performed by Method Referenced from this Attribute
uil-initialization-method	All UIL objects	Initializes the UIL object to the default state for objects of its class.
uil-update-method	All UIL objects	Updates the value of the UIL object from the source attribute of the source object currently specified for this UIL object.
uil-conclude-method	All UIL objects except text objects	Concludes the value of the UIL object to the target attribute of the target object currently specified for this UIL object.
uil-delete-method	All UIL objects	Deletes this UIL object.
uil-clone-method	All UIL objects	Performs a clone operation on the UIL object.
uil-handler-method	Buttons	Invokes callback for button when button is clicked.
uil-validation-method	Dialogs and edit boxes only	Validates the contents of the UIL object using the currently specified format.
uil-configuration-method	All UIL objects	Updates appearance of UIL object to reflect its currently selected configuration.
uil-size-of-method	All UIL objects	Returns the height and width of the UIL object.
uil-manage-method	All UIL objects	Manages (displays) the UIL object
uil-enable-method	All UIL objects	Enables the UIL object.

Method Attributes of UIL Objects

Attribute	Classes with this Attribute	Operations Performed by Method Referenced from this Attribute
uil-disable-method	All UIL objects	Disables the UIL object.
uil-message-selection-method	Edit boxes and message objects	Starts an edit session on an edit box. Selects a message object.
uil-message-unselection-method	Edit boxes and message objects	Ends the edit session on an edit box. Deselects a message object.
uil-scroll-increment-method	Scroll areas	Moves message objects in the scroll area up one page when the user clicks on the scroll bar below the scroll thumb.
uil-scroll-decrement-method	Scroll areas	Moves message objects in the scroll area down one page when the user clicks on the scroll bar above the scroll thumb.
uil-scroll-increment-line-method	Scroll areas	Moves message objects in the scroll area up one line when a user clicks on the scroll down arrow.
uil-scroll-decrement-line-method	Scroll areas	Moves message objects in the scroll area down one line when a user clicks on the scroll up arrow.
uil-scroll-to-position-method	Scroll areas	Scrolls the scroll area whenever a user drags the scroll thumb.

UIL Actions

Actions are procedures that perform operations on a dialog, such as opening or closing the dialog, or updating or concluding the values in it.

UIL provides system-defined actions that perform all the commonly required operations on dialogs. You can create customized actions to perform any specialized operations that your application requires.

Action buttons, user menu choices, and procedures can run actions on dialogs by invoking the procedure `uil-control-dialog-callback`. This procedure references a specified **action description array**, an object that stores a list of actions. `uil-control-dialog-callback` runs the actions listed in the action description array on a specified dialog.

`uil-control-dialog-callback` is also the default callback of push buttons. However, a push button runs the actions referenced by its `uil-action-description` attribute, rather than the actions in an action description array.

The OK, Apply, and Close push buttons that appear on a master dialog by default reference appropriate sets of actions. Your application can use the OK, Apply, and Close push buttons with their default actions for almost all purposes. You can edit any push button to change the list of actions that it runs on a dialog.

For more information about how to run actions on dialogs, see [Launching Dialogs](#). For information about how to associate actions with push buttons, see [Push Buttons](#).

Caution An action fails if it attempts to run `uil-control-dialog-callback` on the target dialog of the push button that runs the action itself.

UIL Callbacks

Callbacks are invoked by the handler of a button whenever a user clicks on the button. The callback invoked by a button is referenced by the `callback` attribute of that button. Callbacks are invoked by push buttons, radio buttons, check buttons, and toggle buttons.

Callbacks on Push Buttons and Other Kinds of Buttons

Push buttons use a different callback from other kinds of buttons.

Push buttons use the callback `uil-control-dialog-callback`. When a user clicks on a push button, `uil-control-dialog-callback` runs a specified set of actions on the target

dialog of that push button. For information about how to specify the actions that `uil-control-dialog-callback` runs, see [Push Buttons](#).

If you need to customize the operations that are performed when a user clicks on a push button, you can do this by creating customized actions and including these actions among the set of actions run by `uil-control-dialog-callback`. Replacing `uil-control-dialog-callback` with a user-defined callback is not recommended.

By default, toggle buttons, radio buttons, and check buttons execute a callback, `uil-do-nothing`, that performs no actions. This callback is suitable for most uses of radio buttons, check buttons, and toggle buttons.

However, if you want a radio button, check button, or toggle button to perform specialized processing, you can create a customized callback to use in place of the default callback `uil-do-nothing`. You must specify the name of the customized callback in the `callback` attribute of the radio button, check button, or toggle button.

Callbacks in GUIDE 3.0 and GUIDE 4.0

In GUIDE 3.0, almost all processing of dialogs and their contents is performed by callbacks. Beginning in GUIDE 4.0, almost all of this processing is performed by methods and actions.

GUIDE 4.0 and later versions support user-defined callbacks created with GUIDE 3.0. However, you can upgrade your 3.0 GUIDE applications to use methods and actions in place of callbacks. For information about how to do this, see Chapter 26, “Upgrading Guide Applications” in the *G2 GUIDE User’s Guide G2 Utilities Version 5.0* manual.

Creating Methods, Actions, and Callbacks

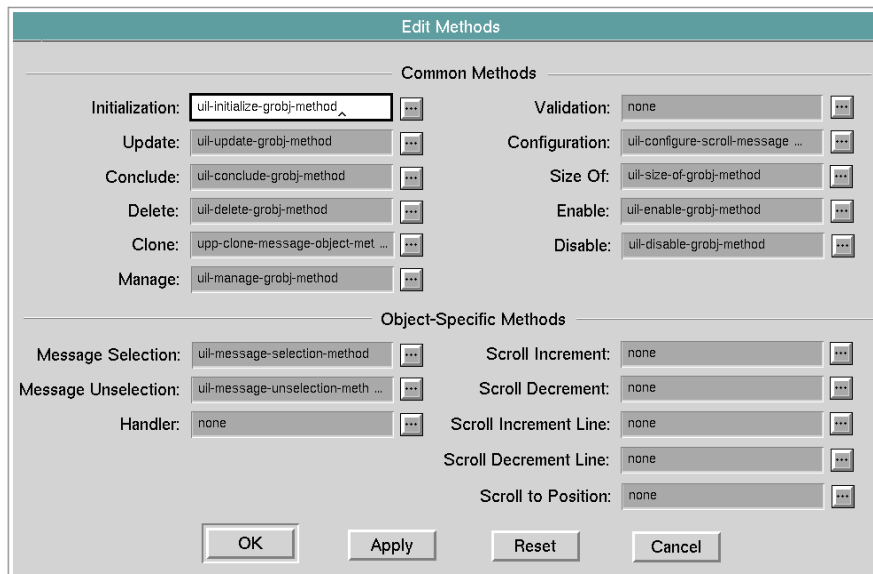
You can create your own methods to use in place of the default methods that GUIDE provides for the method attributes of UIL objects. The methods that you create can perform any operations that are required operations.

Creating UIL Methods Using the Edit Method Dialog

You can create UIL methods, using the Edit Methods dialog. This dialog enables you to select the attribute for which you want to create a method, and provides the correct argument signature for methods referenced from that attribute.

To create a customized method for a UIL object, follow these steps:

- 1 Open the object editor for the UIL object that you want to create.
For example, if you want to create a customized method for an edit box, select edit edit box from the edit box's menu.
- 2 In the object editor, click the Methods button to open the Edit Methods dialog:



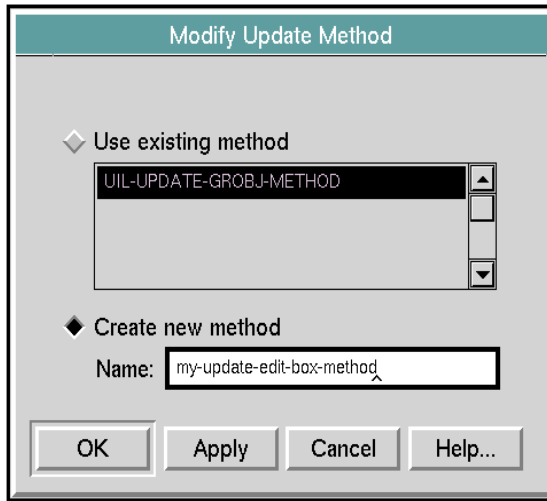
If the class of UIL control that you are editing does not use a particular kind of method, the field and button for that method are disabled in the Edit Methods dialog.

The methods listed under the label Common Methods are common to dialogs and all or most classes of UIL controls that reference methods from their attributes.

Methods used only by certain classes of UIL controls are listed under the label Object Specific Methods.

- 3 In the Edit Methods dialog, click the button to the right of the attribute for which you want to create a method. This opens a dialog that enables you to create a method for that attribute.


For example, if you want to create an update method, click the button to the right of the edit box labeled Update. This opens the Modify Update Method dialog:



- 4 Enter a name for the method that you are creating and click OK. This creates a method with the argument signature that is required for the attribute that you selected. When the method is run, appropriate values are passed to these arguments automatically.

For example, an update method for an edit box looks like this:

```

MY-UPDATE-EDIT-BOX-METHOD
 my-update-edit-box-method (G: class item, W:
class g2-window, D-or-W: class item, B:
item-or-value, action-queue: item-or-value)
begin
{-- The update method for an object takes the
value specified by the object's uil-event-
source-object and uil-event-source-attribute
and reflects it in the object's state. (Note
that for objects residing on workspaces, the
update method works only if the source
object is specified by name.)

G is the UIL object on which to run the update
method
W is the window on which D-or-W is managed
D-or-W is the dialog or workspace containing
the object
B is the button or object that initiated the
update action (optional)
action-queue is the list of pending actions for
the dialog initiated by B (optional)

No values are returned by this method. --}
end
    
```

Caution Do not edit the arguments of the new method. Each method referenced from the attribute of an object has a set of arguments to which GUIDE passes values automatically when the method is invoked. The method cannot function properly if you alter these arguments.

- 5 Edit the body of the method to specify the operations that you want it to perform.

Creating Callbacks, Methods, Procedures, Functions, and Actions Using the GUIDE Method Help Dialog

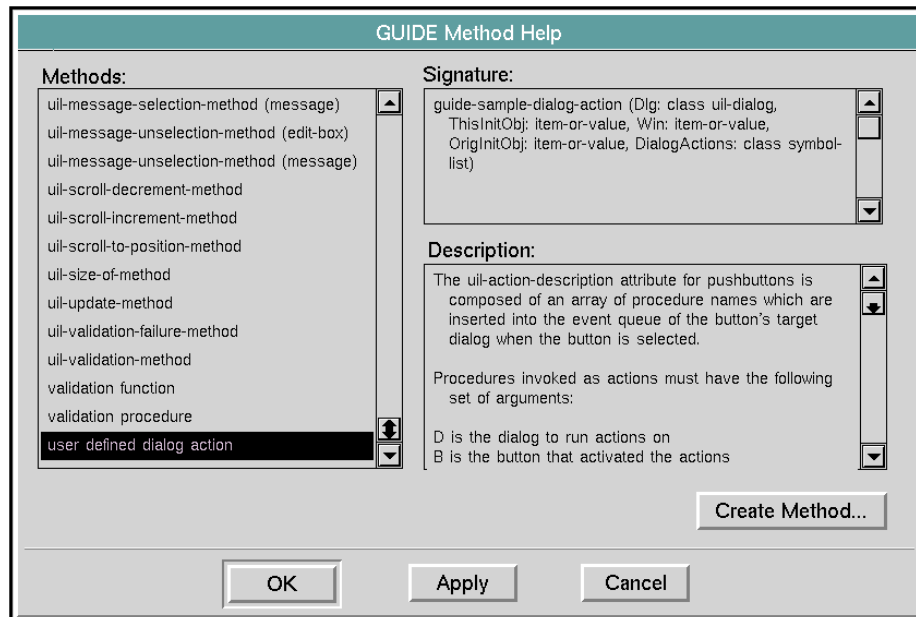
You can use the GUIDE Method Help dialog to create the following kinds of procedures:

- Callbacks
- Methods
- Validation functions
- Validation procedures
- User defined dialog actions

To create a callback, method, or action using the GUIDE Method Help dialog:

- 1 Click the question mark (?) button in the GUIDE palette to open the GUIDE Help palette.
- 2 In the GUIDE Help palette, click the button labeled UIL Methods, Actions, and Callbacks to open the GUIDE Method Help dialog.
- 3 In the Methods column of the GUIDE Method Help dialog, select the kind of procedure that you want to create.

For example, to create an action, select **user defined dialog action** in the Methods column. The required argument signature for actions appears in the Signature scroll area. A description of actions appears in the Description scroll area.



- 4 Click the Create Method button to open the Create New Method dialog.
- 5 In the Create New Method dialog, enter the name of the procedure that you want to create.

For example, you can name an action **my-sample-action**:



- 6 Click the Create Method button.

GUIDE creates a procedure of the kind that you specified (method, action, or callback). The new procedure has the name that you specified in the Create New Method dialog.

GUIDE places the new procedure on an unnamed workspace. You can transfer the procedure from the unnamed workspace to another workspace, and then delete the unnamed workspace.

The following figure illustrates the user-defined action `my-sample-action` and the table showing the action's default definition:



MY-SAMPLE-ACTION

MY-SAMPLE-ACTION, a procedure	
Notes	OK
Authors	none
Item configuration	none
Tracing and breakpoints	default
Class of procedure invocation	none
Default procedure priority	6
Uninterrupted procedure execution limit	use default
<p><code>my-sample-action</code> (D: class <code>uil-dialog</code>, B: <code>item-or-value</code>, W: <code>item-or-value</code>, O: <code>item-or-value</code>, SL: class <code>symbol-list</code>)</p> <pre>begin {– The <code>uil-action-description</code> attribute for pushbuttons is composed of an array of procedure names which are inserted into the event queue of the button's target dialog when the button is selected. Procedures invoked as actions must have the following set of arguments: D is the dialog to run actions on B is the button or object that activated the actions W is the window on which D is displayed O is the object that initiating the original activity for this dialog, or the top most parent dialog (if there is one) SL is a <code>symbol-list</code> containing the names of the remaining actions to run on dialog –} end</pre>	

Help Dialog

Describes the GUIDE help facility.

Introduction **393**

Displaying Argument Signatures of UIL Methods, Callbacks, and Actions **395**

Displaying Help for UIL Methods **397**

Generating Master Dialogs **399**

Using UIL Examples **400**

Using the GUIDE Online Tutorial **402**

Using the GUIDE Debugging Utility **402**



Introduction

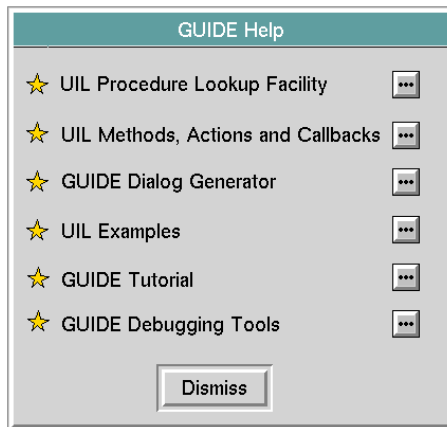
This chapter describes the GUIDE Help dialog, through which you can access the following features of GUIDE:

- UIL Procedure Lookup Facility, which you can use to display the argument signatures of particular methods, actions and callbacks.
- GUIDE Method Help dialog, which you can use to display general descriptions of methods, actions and callbacks, and to create customized methods, actions, and callbacks.
- GUIDE Dialog Generator, which you can use to generate dialogs for viewing and editing the attributes of objects created from user-defined classes.

- The GUIDE Examples workspace. Through this workspace, you can access different categories of working online examples of GUIDE features.
- The GUIDE tutorial. This online tutorial leads you through the steps of creating and using a GUIDE user interface.
- The GUIDE Debugging Utility. This feature enables you to display messages that help you track and debug the execution of your GUIDE application.

Opening the GUIDE Help Dialog

To open the GUIDE Help dialog, click the question mark (?) icon to the left of the Mode button in the GUIDE palette. The GUIDE Help dialog looks like this:

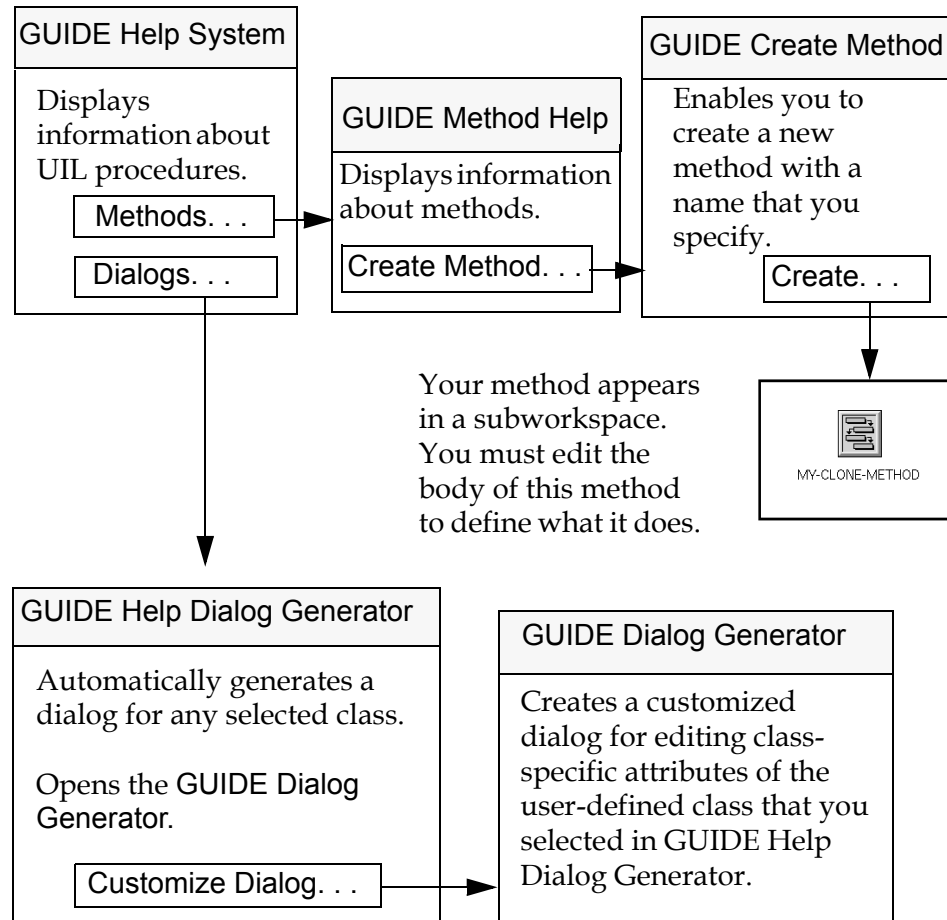


To access a feature through the GUIDE Help dialog, click one of the button with three dots (...) that appear in a column on the right side of the dialog.

The GUIDE Help dialog remains open and usable until you click the Dismiss button.

The following figure illustrates the dialogs that you can access through the GUIDE Help dialog:

Help System Dialogs



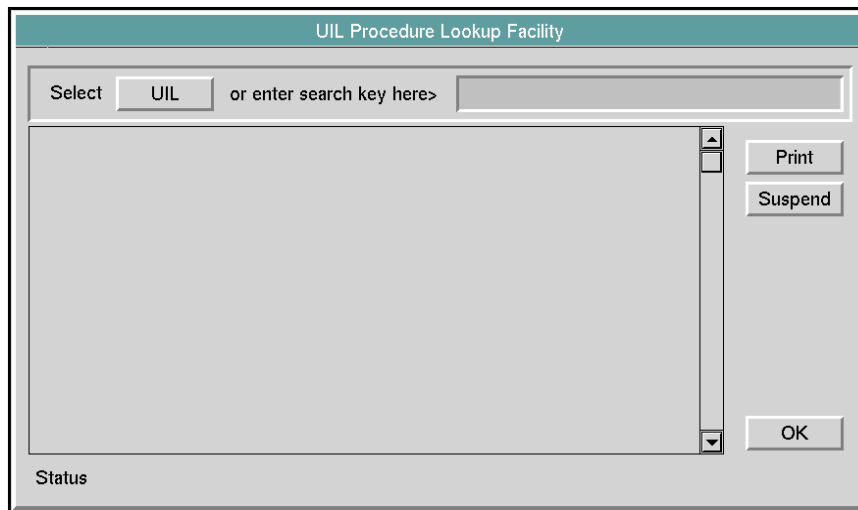
Displaying Argument Signatures of UIL Methods, Callbacks, and Actions

You can display the argument signature of any UIL method, action, or callback using the UIL Procedure Lookup Facility.

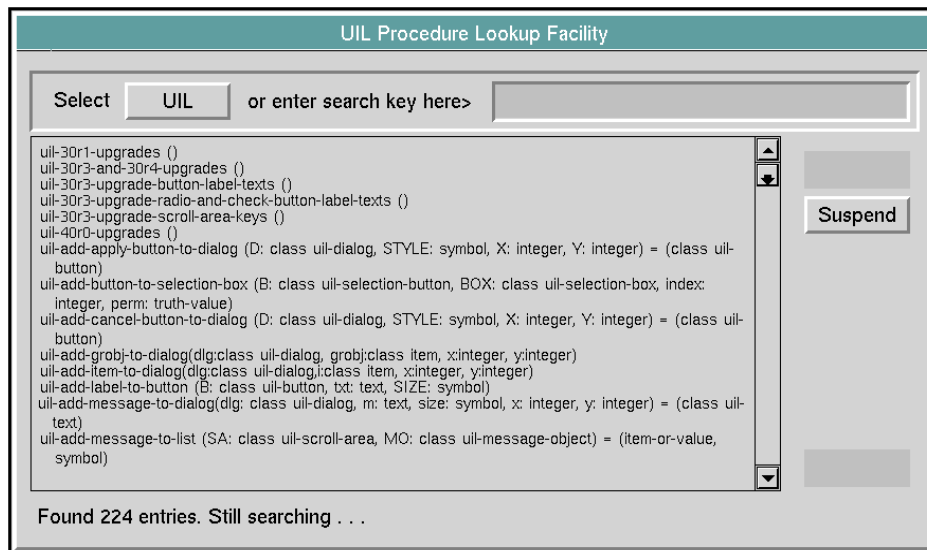
You can open the UIL Procedure Lookup Facility in either of two ways:

- Click the button labeled **Uil Procedure Lookup Facility** on the **GUIDE Help** dialog.
- Select **Help > UIL Procedure Lookup Facility** from the **GUIDE** menu bar.

The Uil Procedure Lookup Facility looks like this:



To display a list of all UIL procedures, click the UIL button to the right of the label Select. You see the following display of UIL procedures:

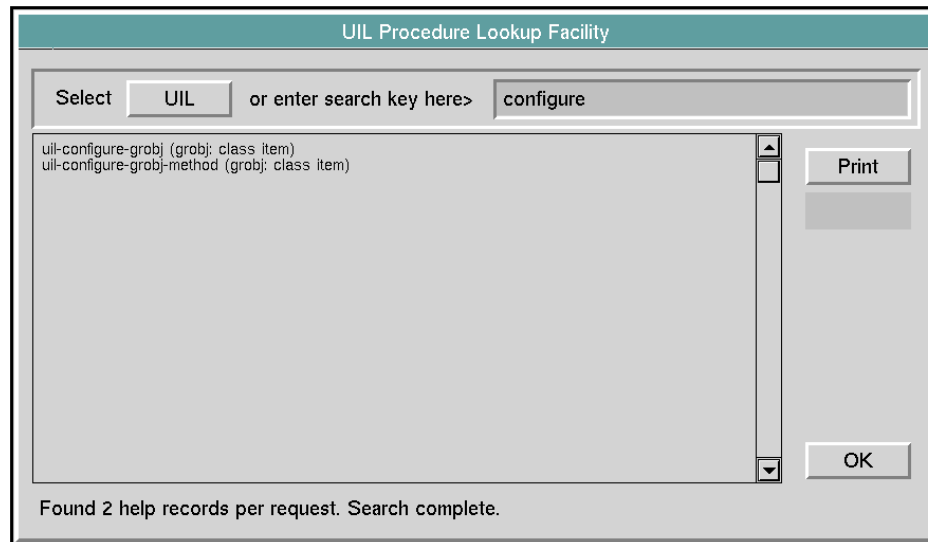


You can stop the search for UIL procedures before it is complete by clicking on the Suspend button.

To print the contents of the scroll area, click the Print button.

If you want information only about certain procedures, you can enter a search key in the edit box to the right of the prompt: or enter search key here >. When you press Return after entering the search key, GUIDE searches for all procedures whose names or argument lists include the search key.

For example, the following UIL Procedure Lookup Facility dialog displays the names of procedures that contain the word configure:



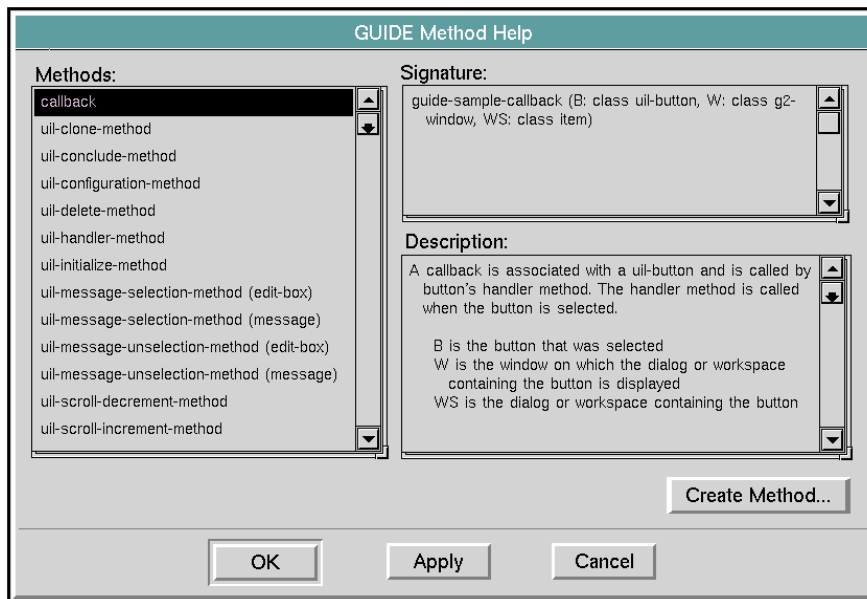
Displaying Help for UIL Methods

You can display a general descriptions of methods, actions, and callbacks, using the GUIDE Method Help dialog. Through the GUIDE Method Help dialog, you can also open the GUIDE Create Method dialog, which you can use to create methods, actions, and callbacks.

You can open the GUIDE Method Help dialog in either of two ways:

- Click the button labeled Uil Methods, Actions, and Callbacks on the GUIDE Help dialog.
- Select Help > UIL Methods, Actions, and Callbacks from the GUIDE Menu Bar.

The GUIDE Method Help dialog looks like this:



In the scroll area under the label **Methods**, you see a list of the currently defined methods. Clicking on a method in the **Methods** scroll area displays its calling signature in the **Signature** scroll area and a short description of its purpose in the **Description** scroll area.

For example, if you click **callback**, the signature and general description of callbacks are displayed, as shown in the figure above.

To create a method, dialog action, or callback, click the category of the procedure in the **Methods** column, and click the **Create Method** button. This opens the **GUIDE Create Method** dialog. For information about how to use this dialog, see [Creating Callbacks, Methods, Procedures, Functions, and Actions Using the GUIDE Method Help Dialog](#).

Finding the UIL Help System File

The **GUIDE** help system file lists the procedures in the public API to **UIL**. These procedures can be invoked by any other **G2** procedure.

The **More Options** palette includes an icon that represents the **GUIDE** help system file:



When you click this icon, a dialog appears that gives the file name and the current pathname of the help system file:

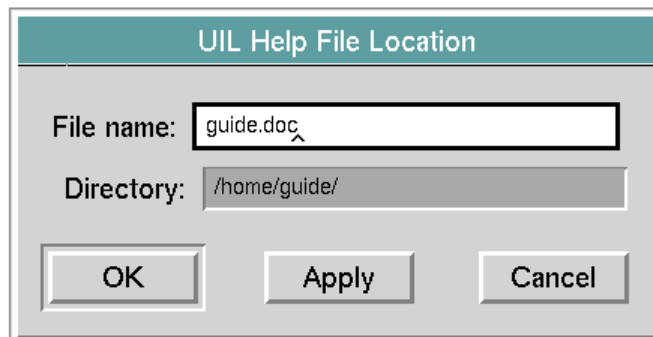
UIL-HELP-FILE-LOCATION, an uil-help-system-location	
Uil help system file name	"guide.doc"
Uil help system directory	"/home/guide/"

UIL-HELP-FILE-LOCATION, an uil-help-system-location	
Uil help system file name	"guide.doc"
Uil help system directory	"/home/guide/"

The help system file lists the procedures provided by the G2 GUIDE User Interface Library (GUIDE/UIL).

To set the location of the help file:

- ➔ Choose Help/Setting UIL Help File Location from the GUIDE menu bar to open the UIL Help File Location dialog, in which you can set the help file location:



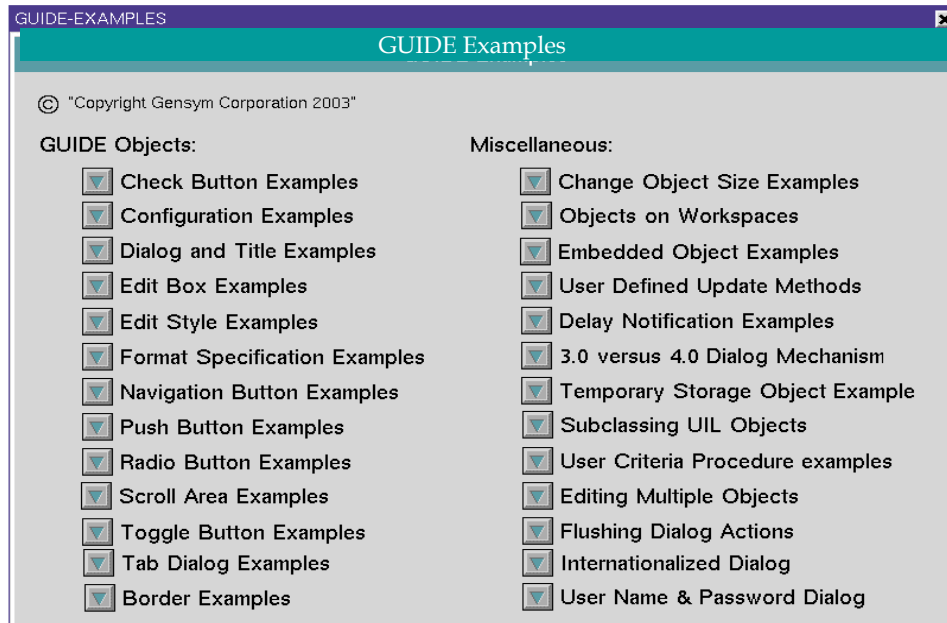
Generating Master Dialogs

The GUIDE Dialog Generator enables you to generate dialogs for viewing and editing the attributes of objects created from user-defined classes.

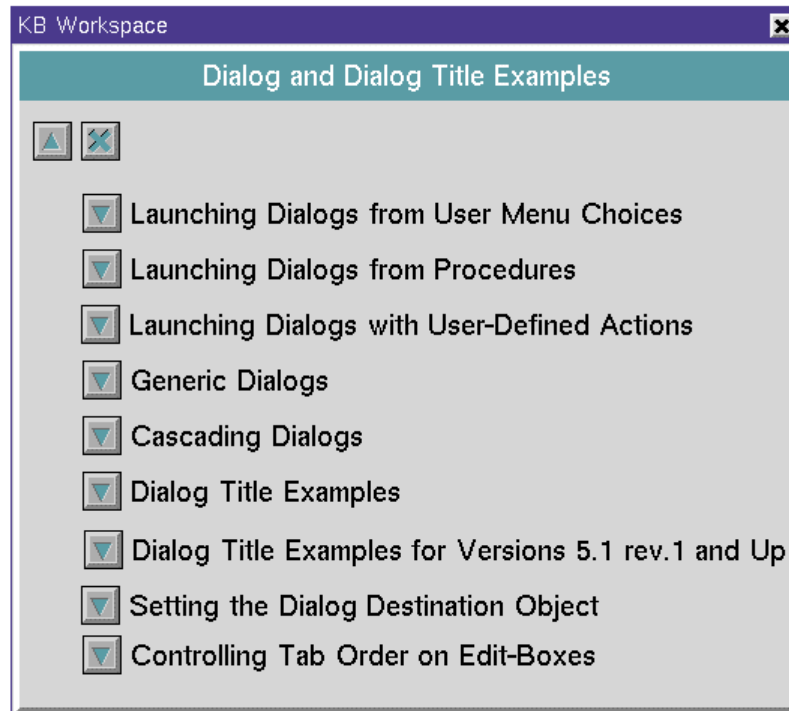
To open the GUIDE Dialog Generator dialog, click the button labeled GUIDE Dialog Generator on the GUIDE Help dialog. For information about how to use the GUIDE Dialog Generator, see [Generating Master Dialogs](#).

Using UIL Examples

To open the GUIDE Examples workspace, click UIL Examples in the GUIDE Help dialog. The GUIDE Examples workspace looks like this:

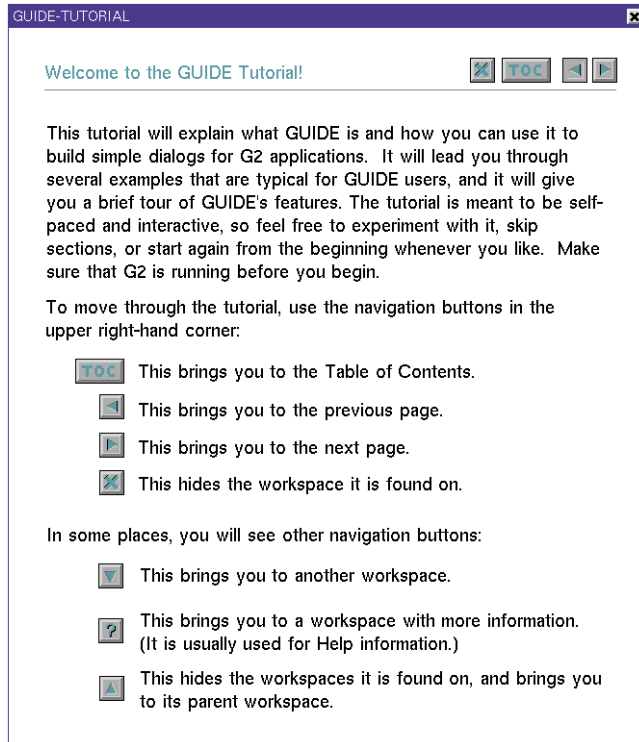


You can click any of the buttons in this workspace to open workspaces of examples on different topics. For example, if you click the Dialog and Title Examples button, you open a workspace from which you can select different working examples of dialogs and dialog titles:



Using the GUIDE Online Tutorial

To open the GUIDE Tutorial, click GUIDE Tutorial in the GUIDE Help dialog. The main workspace of the GUIDE Tutorial workspace looks like this:

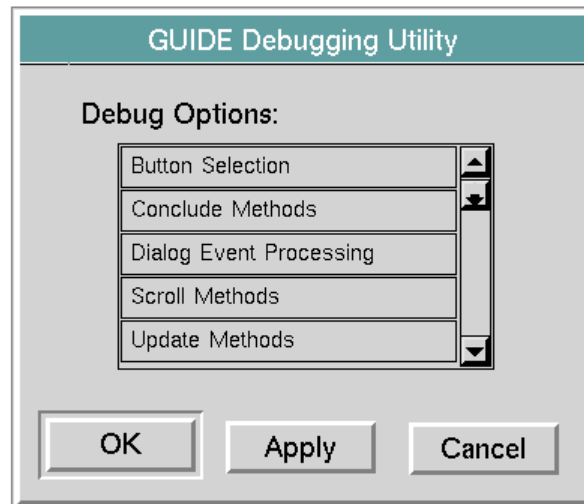


The GUIDE Tutorial is a series of workspaces containing explanations and exercises that teach you how to use of all the main features of GUIDE. You can navigate among these workspaces using the navigation buttons that appear in the upper right corner of the workspaces.

Using the GUIDE Debugging Utility

The GUIDE Debugging Utility enables you to display messages that enable you to track and debug the execution of your GUIDE application. These messages enable you to trace the execution of system-defined methods and actions.

To start the GUIDE Debugging Utility, click GUIDE Debugging Tools in the GUIDE Help dialog. The GUIDE Debugging Utility looks like this:



Click the option or options in the scroll area labeled Debug Options for the features of your GUIDE user interface that you want to debug, then click OK.

You can also select the debugging options provided in the GUIDE Debugging Utility dialog by selecting the following choices from the GUIDE menu bar:

- Tools > GUIDE Debugging> Button Selection
- Tools > GUIDE Debugging> Conclude Methods
- Tools > GUIDE Debugging> Dialog Event Processing
- Tools > GUIDE Debugging> Scroll Methods
- Tools > GUIDE Debugging> Update Methods
- Tools > GUIDE Debugging> Validation Methods

When you run a G2 application that uses your GUIDE interface, debugging messages are posted on the message board for the features of your application that you chose in the GUIDE Debugging Utility.

Creating Custom UIL Subclasses

Describes how to create customized subclasses of system-defined UIL classes provided with GUIDE.

Introduction **405**

Creating and Using Customized Subclasses **406**

Creating a Customized Object Definition **411**

Creating a Customized Message Definition **414**

Creating Instances of Customized Subclasses and Adding them to Master Dialogs **417**

Creating Subclasses of uil-object and uil-message **417**



Introduction

Your application may require dialogs or UIL controls that look or behave differently from dialogs or UIL controls created from system-defined UIL classes.

You can create dialogs and UIL controls with a customized appearance and behavior by creating subclasses of system-defined UIL classes and editing attributes of the subclass to specify the customized appearance or behavior. Every dialog or UIL control that you create from this subclass inherits the customized appearance and behavior that you specify in the subclass definition.

Creating and Using Customized Subclasses

This section describes the steps that you must follow to create and use customized subclasses of UIL controls. These steps are:

- 1 Choose an appropriate parent class for your customized subclass.
Choose a class from which you can create instances, rather than a general class that cannot have instances. For example, base a subclass of toggle buttons on `uil-medium-text-toggle-button`, rather than on `uil-text-button`.
- 2 Create an object definition or message definition that inherits from the parent class that you choose.
- 3 Customize the behavior or appearance of the object definition or message definition that you created.
- 4 Create instances of your customized subclass and add them to the master dialog or workspace where you want to use them.

Choosing a Parent Class for a Customized Subclass

You can create subclasses of object definitions or of message definitions.

- Subclasses for buttons, dialogs, selection boxes, scroll areas, borders, and separators must be object definitions.
- Subclasses for edit boxes, text objects, dialog titles and message objects must be message definitions.

After you decide whether to create an object definition or a message definition, you then choose the specific UIL class that you want to use as the parent class of the subclass. Choose a parent class that has the class-specific attributes and behavior that you want your subclass to inherit. For example, if you want to create a customized subclass of edit boxes, create a message definition that inherits directly from the class `uil-edit-box`.

The following table lists the system-defined UIL classes that are based on object definitions:

UIL Object Definitions

UIL Object	System-Defined UIL Classes
Fully customized object definition	<p>uil-object</p> <p>The class uil-object has no attributes. You must specify attributes for methods or values that you want to associate with any subclass of uil-object that you create.</p> <p>For an example of how to create a subclass of uil-object, see Creating Subclasses of uil-object and uil-message.</p>
Dialog	<p>uil-dialog</p> <p> uil-query-dialog</p> <p> uil-message-dialog</p> <p> uil-confirm-dialog</p> <p> uil-tailored-dialog</p>
Selection box	<p>uil-selection-box</p> <p> uil-check-box</p> <p> uil-radio-box</p>
Buttons	<p>uil-button</p> <p> uil-icon-button</p> <p> uil-selection-button</p> <p> uil-workspace-button</p> <p> uil-text-button</p>
Toggle button	<p>uil-icon-button</p> <p> uil-icon-toggle-button</p> <p>uil-text-button</p> <p> uil-text-toggle-button</p>
Push button	<p>uil-icon-button</p> <p> uil-icon-pushbutton</p> <p>uil-text-button</p> <p> uil-text-pushbutton</p>

UIL Object Definitions

UIL Object	System-Defined UIL Classes
Selection button	uil-selection-button uil-radio-button uil-check-button
Radio button	uil-radio-button
Check button	uil-check-button
Navigation button	uil-workspace-button uil-navigation-button uil-goto-workspace-button uil-goto-superior-button uil-goto-next-button uil-goto-previous-button uil-hide-button uil-help-button
Border	uil-box-border uil-box-border-right uil-box-border-left
Separator	uil-line-separator uil-line-separator-left uil-line-separator-right

The following table lists the system-defined UIL classes that are based on message definitions:

UIL Message Definitions

UIL Object	System-Defined UIL Classes
Fully customized message definition	<p>uil-message</p> <p>The class uil-message has no attributes. You must specify attributes for methods or values that you want to associate with any subclass of uil-message that you create.</p> <p>For an example of how to create a subclass of uil-message, see Creating Subclasses of uil-object and uil-message.</p>
Graphical message	<p>uil-grmes</p> <p>uil-message-object, uil-edit-box, uil-text</p>
Message object	uil-message-object
Edit box	uil-edit-box
Text object	uil-text
Dialog title	uil-dialog-title

Note If UIL provides subclasses of a UIL control in different sizes (for example, the three subclasses of uil-text: uil-small-text, uil-medium-text, and uil-large-text), use one of the subclasses with a size specification as the parent class of your object definition or message definition. The subclass that you create inherits a fully-defined icon from the subclass with the size specification.

Most classes of UIL controls have subclasses in different sizes. To see the complete class hierarchy of any UIL class, use the **Inspect** utility.

Customizing a Message Object Class

You can create customized classes of message objects to add more information capability to the message objects. For example, you can add an alarm-state attribute to message objects.

To customize message objects, you should customize the default message object class that GUIDE automatically generates for a scroll area when you clone the scroll area from the G2 GUIDE palette.

The default message object class is referenced by the `message-class-name` attribute of the scroll area and has an automatically generated name. The following line from a scroll area attribute table illustrates an automatically generated name:

Message class name	top-mod-message-object-h-150-w-250-medium
--------------------	---

The default message object class inherits directly from `uil-message-object`.

Caution It is good practice to customize the default message object subclass of a scroll area, rather than replace it with a user-defined subclass that has a different name.

When you manually resize a scroll area, GUIDE automatically generates a new subclass of `uil-message-object` for the message objects in the scroll area. Any modifications that you made to the original default message object subclass are retained in the new subclass that GUIDE generates.

However, if you replace the default message object class of a scroll area with a user-defined subclass, the customized features of this subclass are lost when you manually resize the scroll area.

The following sections describe how to create customized subclasses based on object definitions and message definitions.

Customizing the Behavior and Appearance of Subclasses of `uil-grobject` or `uil-grmessage`

To customize the behavior of a user-defined subclass of `uil-grobject` (an object definition) or `uil-grmessage` (a message definition), follow these steps:

- 1 Create methods that perform customized operations. These methods define the customized behaviors of your object definition or message definition.
- 2 Reference these methods from the appropriate attributes of the object definition or message definition that you created. Your user-defined methods replace the default methods that your object definition or message definition inherits from its parent class.

For example, suppose that you want to create edit boxes whose contents can be viewed and edited by some users, but only viewed by other users. To create these edit boxes, you can:

- 1 Create a message definition that inherits from `uil-edit-box`, the system-defined class for edit boxes. Name your subclass `my-edit-box-class`, for example.
- 2 Create a edit box selection method that grants editing privileges only to the users that you want to be able to edit the boxes created from this subclass. Name your edit box selection method `my-edit-box-selection-method`, for example.

Your customized edit box selection method can grant editing privileges only to users with particular user IDs, or only to users with certain privileges.

- 3 In the message definition `my-edit-box-class`, replace the default edit box selection method, `uil-edit-box-selection-method`, with you customized edit box selection method, `my-edit-box-selection-method`. The edit box selection method is referenced by the `uil-message-selection-method` attribute in the message definition.
- 4 Create instances of `my-edit-box-class` to use in the user interface that you are creating.

For information about how to create methods, see [Methods, Actions, and Callbacks](#).

To customize the appearance of objects created from your user-defined object definition, edit the value of `icon-description` attribute of the definition.

The following sections describe in detail how to create and customize object definitions and message definitions.

Creating a Customized Object Definition

Classes and subclasses of buttons, dialogs, dialog titles, selection boxes, scroll areas, borders, and separators are object definitions.

To create a customized object definition:

- 1 Create a new object definition. To do this, select:
KB workspace > New Definition > object definition

- 2 Open the attribute table of the new object definition:



an object-definition	
Notes	INCOMPLETE, and note that (1) no class name is specified; (2) no direct superior classes are specified
Authors	none
Class name	none
Direct superior classes	none
Class specific attributes	none
Instance configuration	none
Change	none
Menu option	a final menu choice
Class inheritance path	none
Inherited attributes	none
Attribute initializations	none
Attribute displays	inherited
Stubs	inherited
Icon description	inherited


As the notes slot indicates, the new object definition is incomplete. It requires values for class-name and for direct-superior-classes.

- 3 Enter a name for the object definition in the class-name slot.
- 4 Enter the name of the parent class for your subclass in the direct-superior-classes slot.

When you specify a parent class, the `class-inheritance-path` slot and the `inherited-attributes` slot are filled with values reflecting this class:

- The `class-inheritance-path` slot lists the UIL classes from which the new subclass inherits attributes.
- The `inherited-attributes` slot lists these inherited attributes.

The following figure illustrates the attribute table of the user-defined class `my-grobj-class`, which is a subclass of the system-defined class `uil-grobj`:



MY-GROBJ-CLASS

MY-GROBJ-CLASS, an object-definition	
Notes	OK
Authors	jfb (8 Jun 1995 7:41 a.m.)
Class name	my-grobj-class
Direct superior classes	uil-grobj
Class specific attributes	none
Instance configuration	none
Change	none
Menu option	a final menu choice
Class inheritance path	my-grobj-class, uil-grobj, uil-object, object, item
Inherited attributes	uil-is-managed initially is 0; uil-is-permanent initially is false; id initially is "", with an index; configuration; uil-delete-method initially is uil-delete-grobj-method; uil-conclude-method initially is uil-conclude-grobj-method; uil-validation-method initially is uil-validate-grobj-method; uil-update-method initially is uil-update-grobj-method;

- 5 To customize behaviors for objects created from this new subclass, specify attribute values in the `attribute-initializations` slot to override the attribute values in the `inherited-attributes` slot.

The following figure illustrates how to override the inherited attribute values for `uil-is-permanent` and `uil-disable-method` by specifying non-default values in the `attribute-initializations` slot:

Attribute initializations	uil-is-permanent initially is true; uil-disable-method initially is my-disable-method
---------------------------	--

The first override value in the `attribute-initializations` slot sets the `uil-is-permanent` value to `true`, rather than `false` (the default). The second override value specifies that a user-defined disable method named `my-disable-method` is used as the disable method for all instances of this class, rather than the default system-defined method `uil-disable-method`.

- 6 Edit the `icon-description` value to customize the size and appearance of the icons that represent objects created from this new subclass.

To edit the size and appearance of the icons, select **edit icon** from the menu of the object definition. This opens the Icon Editor. You can also open the Icon Editor by clicking in the `icon-description` slot in the object definition and choosing **edit icon** from the menu. For information about how to use the Icon Editor, see the *G2 Reference Manual*.

Caution When you create subclasses of system-defined UIL classes, do not change the names of color regions in the icon-descriptions. These names are specifically referenced when the button selection method is activated. You can define additional color regions. However, the UIL configuration method can not access or change them.

To access user-defined regions during the selection and unselection process, you must create a user-defined configuration method. The user-defined configuration method should be modeled after the system-defined UIL configuration method.

Creating a Customized Message Definition

Classes and subclasses of edit boxes, text objects, and message objects are message definitions. The steps for creating a message definition are similar to the steps for creating an object definition.

To create a customized message definition, follow these steps:

- 1 Create a message definition. To do this, select:
KB workspace > New Definition > message definition
- 2 Open the attribute table of the new message definition. The table looks like this:




a message-definition	
Notes	INCOMPLETE, and note that (1) no class name is specified; (2) no direct superior classes are specified
Authors	none
Class name	none
Direct superior classes	none
Class specific attributes	none
Instance configuration	none
Change	none
Menu option	a final menu choice
Class inheritance path	none
Inherited attributes	none
Attribute initializations	none
Default message properties	none

As the **notes** slot indicates, the new message definition is incomplete. It requires values for **class-name** and for **direct-superior-classes**.

- 3 Enter a name for the message definition in the **class-name** slot.

- 4 Enter the name of the parent class for this message definition in the direct-superior-classes slot.

The following figure illustrates the attribute table of the user-defined message definition `my-text-object-class`, which is a subclass of the system-defined class `uil-text`:



MY-TEXT-OBJECT-CLASS

MY-TEXT-OBJECT-CLASS, a message-definition	
Notes	OK
Authors	jfb (8 Jun 1995 8:34 a.m.)
Class name	my-text-object-class
Direct superior classes	uil-text
Class specific attributes	none
Instance configuration	none
Change	none
Menu option	a final menu choice
Class inheritance path	my-text-object-class, uil-text, uil-grmes, uil-message, message, item
Inherited attributes	uil-label-id initially is "", with an index; uil-format-specification initially is unspecified; uil-stand-alone-text initially is true; uil-border-relation initially is unspecified; uil-border-id initially is "", with an index; state initially is enabled; message-contents initially is ""; uil-update-value initially is true; uil-is-managed initially is 0; uil-is-permanent initially is false; id initially is "", with an index;

- 5 To customize behaviors for objects created from this new subclass, specify attribute values in the attribute-initializations slot to override the attribute values in the inherited-attributes slot.

The following figure illustrates how to override the inherited attribute values for `message-contents` and `uil-format-specification` by specifying non-default values in the attribute-initializations slot:

Attribute initializations	message-contents initially is "System is starting. Please wait."; uil-format-specification initially is startup-message-format
---------------------------	---

The first override value in the `attribute-initializations` slot sets the initial message contents of the text object to the string "System is starting. Please wait." The second override value specifies that a format named `startup-message-format` is used initially to format the contents of this message object.

Creating Instances of Customized Subclasses and Adding them to Master Dialogs

To create instances of a customized subclasses of a system-defined UIL class, select `create instance` from the menu of the subclass definition.

You can also use the following procedures to create instances of customized subclasses programmatically:

- `uil-create-custom-button`, which creates buttons.
- `uil-create-custom-text`, which creates edit boxes, text objects, or message objects.

For information about these procedures, see the *G2 GUIDE/UIL Procedures Reference Manual*.

To add an instance of a customized subclass to a master dialog, choose `transfer` from the menu of the instance, and then click the subworkspace of the master dialog to which you want to add the instance.

Creating Subclasses of `uil-object` and `uil-message`

To create a highly customized subclass, you create an object definition that inherits from `uil-object`, or a message definition that inherits from `uil-message`. Unlike other system-defined classes, `uil-object` and `uil-message` have no attributes that subclasses can inherit. For this reason, you can create subclasses of `uil-object` and `uil-message` when you want the greatest possible freedom in customizing your subclass.

Note Objects created from subclasses that inherit directly from `uil-object` and `uil-message` can be moved by users at run-time. In contrast, objects created from any subclass that inherits from `uil-grobj` or `uil-grmes` cannot be moved by users at run-time.

Deciding What Attributes to Add to a Subclass of `uil-object` or `uil-message`

For most purposes, you will need to add attributes to a subclass of `uil-object` or `uil-message` to reference methods that perform basic operations on objects of that subclass, such as configuring, managing, cloning, and deleting the objects.

The following table lists the attributes that you will need to add to subclasses of `uil-object`:

Required Attributes of Subclasses of `uil-object`

Attribute	Description
<code>uil-configuration-method</code>	References a method that applies the currently selected configuration to this object.
<code>uil-manage-method</code>	References a method that displays the object.
<code>uil-clone-method</code>	References a method that makes a copy of the object.
<code>uil-delete-method</code>	<p>References a method that deletes the object.</p> <p>Adding a <code>uil-delete-method</code> attribute and referencing a delete method from this attribute enables you to use the GUIDE/UIL delete. menu choice to delete objects of this subclass. The delete. menu choice deletes the object and any components such as labels that it may include.</p> <p>If you do not add a <code>uil-delete-method</code> attribute to this object definition, you can delete objects of this subclass, using the G2 delete command. However, the G2 delete command does not delete components of the object such as labels. You must delete these components individually.</p>
<code>uil-update-method</code>	<p>References a method that updates objects of this subclass with the value of a source attribute in a source object.</p> <p>If you add a <code>uil-update-method</code> attribute and reference an update method from this attribute, you must also add <code>uil-event-source-object</code> and <code>uil-event-source-attribute</code> attributes to the object definition.</p>

Required Attributes of Subclasses of `uil-object`

Attribute	Description
<code>uil-event-source-object</code>	Specifies source objects for objects of this class.
<code>uil-event-source-attribute</code>	Specifies source attributes for objects of this class.
<code>uil-conclude-method</code>	References a method that concludes the values of objects of this subclass to a target attribute in a target object. If you add a <code>uil-conclude-method</code> attribute and reference a conclude method from this attribute, you must also add <code>uil-event-target-object</code> and <code>uil-event-target-attribute</code> attributes to the object definition.
<code>uil-event-target-object</code>	Specifies target objects for objects of this class.
<code>uil-event-target-attribute</code>	Specifies target attributes for objects of this class.
<code>message-contents</code>	(<code>uil-message</code> subclasses only) The full, unformatted contents of objects of this subclass. The formatted and clipped version of the text is stored in the <code>text</code> attribute of the object, which you cannot view or edit directly.

Specifying the Colors of UIL Objects

Describes how to create reusable objects called configurations, which specify the colors of the different regions of the graphical components in your user interface.

Introduction **421**

Creating Configurations **422**

Using The GUIDE Configuration Editor **422**



Introduction

You can create customized configurations that specify the colors of different regions of UIL objects, including both dialogs and UIL controls. You can apply each configuration to UIL objects of a particular UIL class.

The color or colors of each UIL object are specified by a **configuration object**. A configuration object specifies the colors of the different regions in each object

To customize the appearance of individual UIL objects, you can edit the configuration of that UIL object. You can also create a customized configuration and apply it to any number of UIL objects of a given class.

Each configuration object is an instance of a configuration class. Each configuration class is designed for use with a single class of UIL objects. Configuration classes for push buttons cannot be used with message objects, configuration classes for message objects cannot be used with scroll areas, and so on.

Each configuration class has attributes that specify the default colors (text, background, border), and the colors that distinguish the enabled and disabled states, and/or the selected and unselected states.

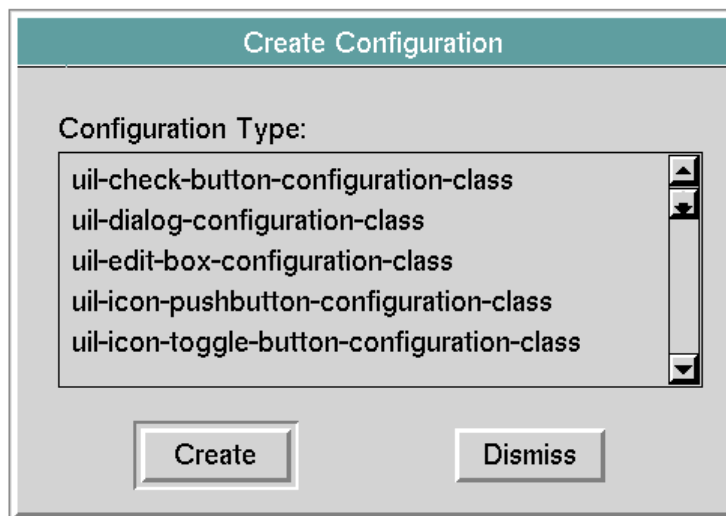
Creating Configurations

To create a configuration, you can:

- Select Item > GUIDE Objects > Configuration from the GUIDE menu bar.
- Click the Configurations button in the More Options palette, which you can open from the GUIDE palette.

These actions open the Create Configuration dialog, which you can use to create a configuration:

Create Configuration Dialog

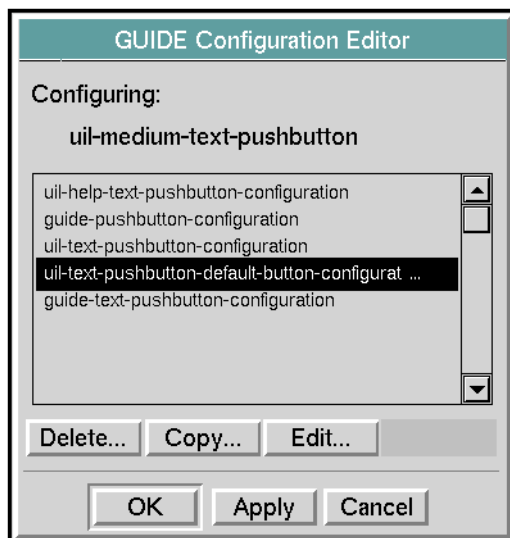


Using The GUIDE Configuration Editor

The GUIDE Configuration Editor enables you to select an existing configuration to apply to an object, to create a new configuration object, to edit the attributes of an existing configuration object, or to delete a configuration object.

Note Altering the values in a configuration object affects all UIL objects that reference that configuration.

To use the GUIDE Configuration Editor on a particular object, click the object and select Edit Configuration from the object's menu. The GUIDE Configuration Editor dialog appears:



The scroll area in the dialog displays the names of all currently defined configurations for the class of the UIL object that you selected. If the **configuration** attribute of the object whose configuration you are editing specifies a configuration, that configuration is selected in the scroll area of the GUIDE Configuration Editor.

The push button labeled OK, when selected, applies the attributes of the selected configuration to the initiating object and dismisses the dialog.

Clicking the Apply button applies the attributes of the selected configuration to the initiating object.

Clicking the Cancel button dismisses the dialog without applying any of the changes made since the last OK or Apply action.

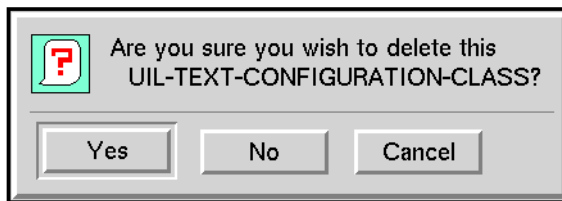
The Delete, Copy, and Edit buttons open dialogs in which you can perform specialized editing operations. Closing a configuration dialog automatically closes any child dialogs of that configuration dialog. Thus if you close the GUIDE Configuration Editor dialog, any child dialogs of GUIDE Configuration Editor that happen to be open are also closed.

The following sections describe the special operations for deleting, copying, and editing configurations.

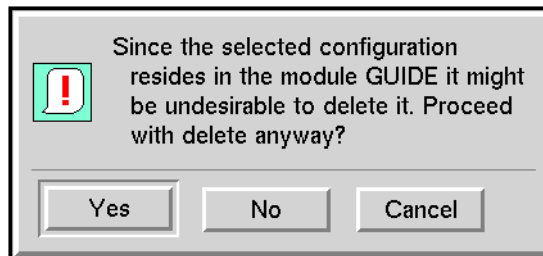
Note If you select edit configuration from the menu of an object while the configuration editor is running on another object in the same window, the configuration editor is refocused on the new object. Within a knowledge base, only one user at a time can edit a configuration.

Deleting Configurations

To delete a configuration, select that configuration and click the push button labeled Delete in the GUIDE Configuration Editor. A dialog appears prompting you to confirm that you want to delete the configuration:



If the configuration that you are deleting is stored in the GUIDE or UIL modules, you also see the following confirmation dialog before you can delete the configuration:



Copying Configurations

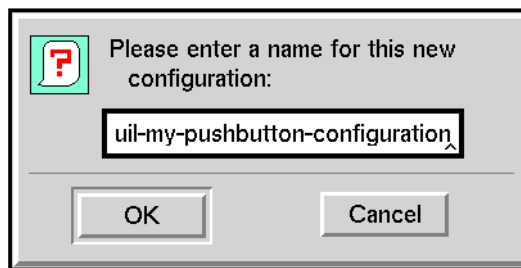
For some purposes, you may want to change the appearance of one UIL object without changing the appearance of other UIL objects that use the same configuration. For example, you may want one push button to have a red border, while all other push buttons in your application have black borders.

To do this, you can:

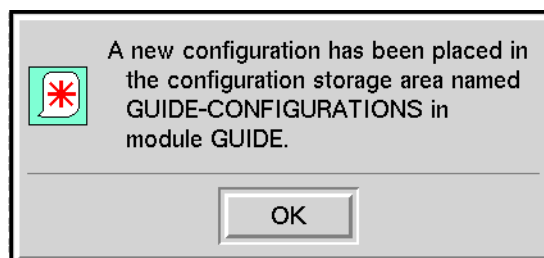
- Create a copy of the configuration used by other UIL objects.
- Modify this new configuration to specify the appearance that you want.
- Apply the new configuration to the UIL object whose appearance you want to change.

To create a copy of a configuration:

- 1 Open the GUIDE Configuration Editor on the UIL object whose appearance you want to modify. The configuration currently used by the UIL object is selected in the scroll area in editor.
- 2 Click the push button labeled Copy to create a copy of the configuration that you see selected in the scroll area.
- 3 Enter a name for the new configuration in the dialog that appears when you push the Copy button:



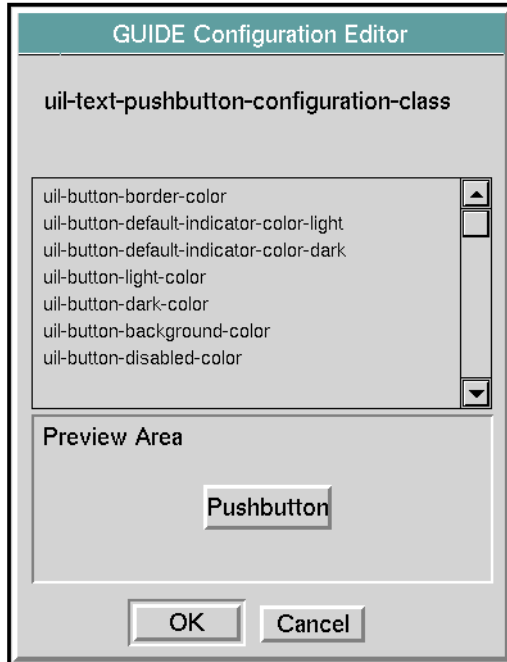
- 4 Click the OK button. A dialog appears notifying you where the configuration was placed:



- 5 The name of your new configuration appears in the scroll area in the GUIDE Configuration Editor. To edit the new configuration to make the change you want – for example, to change the border color to red – click the Edit button. Then follow the steps described in the following section.
- 6 When you finish editing the new configuration, make sure that the new configuration is selected in the GUIDE Configuration Editor, and click OK. The new configuration is applied to the UIL object.

Editing Configurations

To edit an existing configuration, select the configuration in the scroll area in the GUIDE Configuration Editor and then push the button labeled Edit. The following dialog is displayed for editing the configuration:



The scroll area in the upper half of the dialog lists configurations for the different color regions in the object that you are editing. Each configuration also contains color regions for the disabled state of the object. Click the configuration for the color region that you want to edit.

Below the scroll area is a Preview Area. An instance of the initiating object is displayed here. As you make changes, the instance in the Preview Area is updated to reflect the changes.

When you select a color region from the scroll area, a dialog such as the following appears:



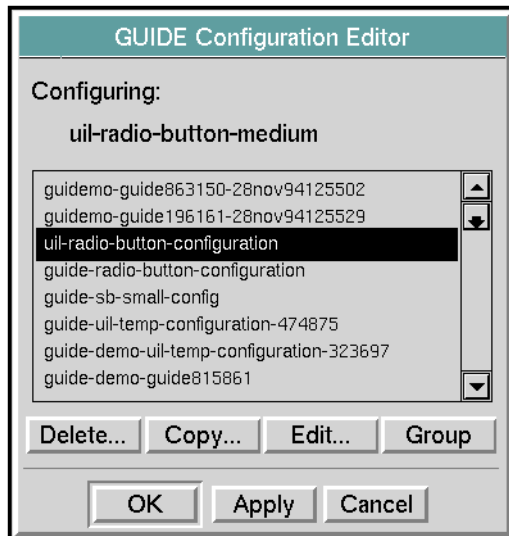
The title of this dialog indicates the color region of the configuration that you are editing. The dialog contains a scroll area with a message object for every supported color. Each message object is configured so that its background color is the actual color that it represents, and the text displayed is the color's name.

Selecting a color updates the object in the Preview Area on the Edit Configuration dialog to reflect your choice of color.

You can open the color selection dialog programmatically, using the procedure `uil-display-color-selector-dialog`. For information about this procedure, see the *G2 GUIDE/UIIL Procedures Reference Manual*.

Applying Configuration Edits to All Buttons in a Group

If you are editing the configuration of a radio button or a check button, the GUIDE Configuration Editor dialog includes a push button labeled Group. Clicking the Group button applies the configuration that you select to all the buttons in the same group as the button whose configuration you are editing. The GUIDE Configuration Editor dialog with a Group button looks like this:



Upgrading GUIDE Applications

Describes how to modify dialogs and other components of a user interface created with earlier versions of GUIDE, to take advantages of the features introduced in newer versions.

Introduction **429**

Upgrading 5.0 KBs **430**

4.0 and 5.0 Conversion Tools **430**

Editing the Label Text of Generic Dialogs **430**

Extending Context-Sensitive Help **436**



Introduction

This chapter lists the steps that you must follow to convert applications based on an earlier version of GUIDE.

You can upgrade a GUIDE application only from the previous release of GUIDE. For example, you cannot directly upgrade a GUIDE 4.0 application to GUIDE 7.0. You must first upgrade from 4.0 to 5.0, then to 6.x, then to 7.x, then to 8.x. If you need to upgrade from a version earlier than 5.0, see the documentation that was distributed with the version of GUIDE to which you need to upgrade or contact Gensym Customer Support for assistance.

Upgrading 5.0 KBs

You can upgrade applications developed with GUIDE 5.0 simply by loading the KB into the G2 6.x, 7.0, or 8.0 component. There are no special steps that you must take.

4.0 and 5.0 Conversion Tools

The GUIDE menu bar has a selection for Tools > GUIDE 50r0 Migration Tools. These tools are included with GUIDE to maintain backward compatibility with previous versions of GUIDE, and should not be used to convert aspects of 3.0 or 4.0 applications to the current version of GUIDE.

See the *G2 GUIDE User's Guide G2 Utilities Version 5.0* manual to do any of the following:

- Convert uil-scroll-areas to 41r0 scroll-areas.
- Convert uil-buttons to 50r0 buttons.
- Convert dialog subworkspaces to uil-dialog-subworkspaces.
- Use Icon Text in Button Labels.

Editing the Label Text of Generic Dialogs

In GUIDE/UIL, generic dialogs derive the text of their labels from GFR (G2 Foundation Resources) local text resources. GUIDE/UIL supports four kinds of generic dialog: confirmation, query, message, and notification. For information about generic dialogs, see [Message, Query, Confirmation, and Notification Dialogs](#).

Through GFR, you can easily:

- Change the English-language text of labels in generic dialogs.
- Convert labels to languages other than English.

For information about GFR, see the *G2 Foundation Resources User's Guide*.

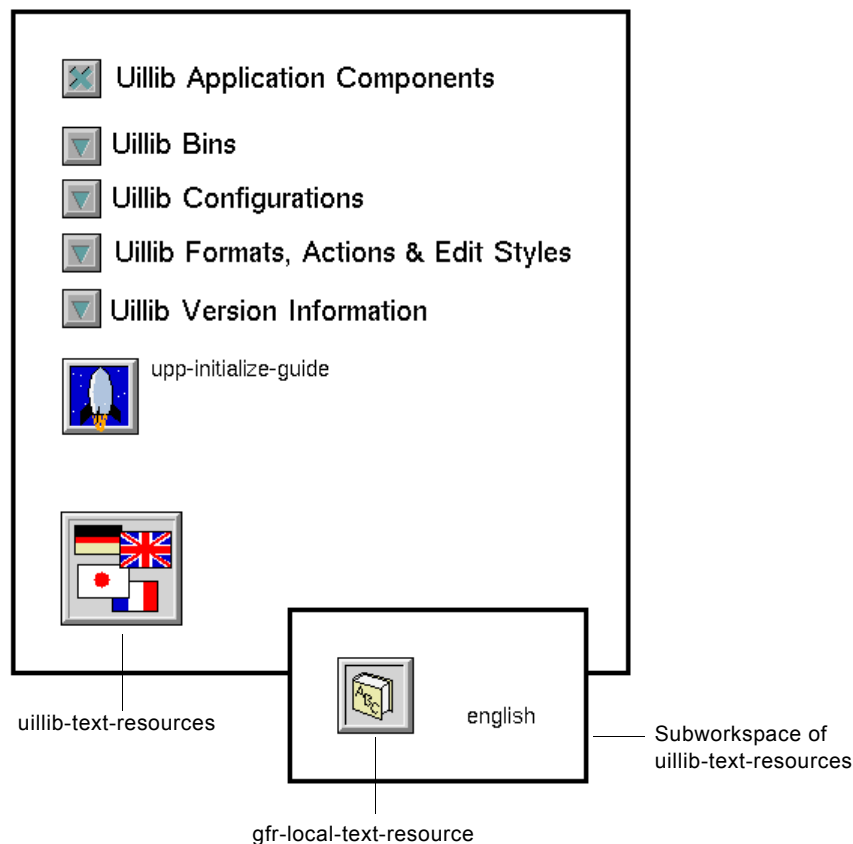
You can edit the English text of generic dialog labels, or specify non-English text for the labels, in either of two ways:

- Edit the `gfr-local-text-resource` in the default GFR text resource group `uilib-text-resources`.
- Create a `gfr-text-resource-group` and a `gfr-local-text-resource` for English, and edit the `gfr-local-text-resource`. Then edit the master generic dialogs so that they reference the English `gfr-text-resource-group`.

To edit label text in uilib-text-resources:

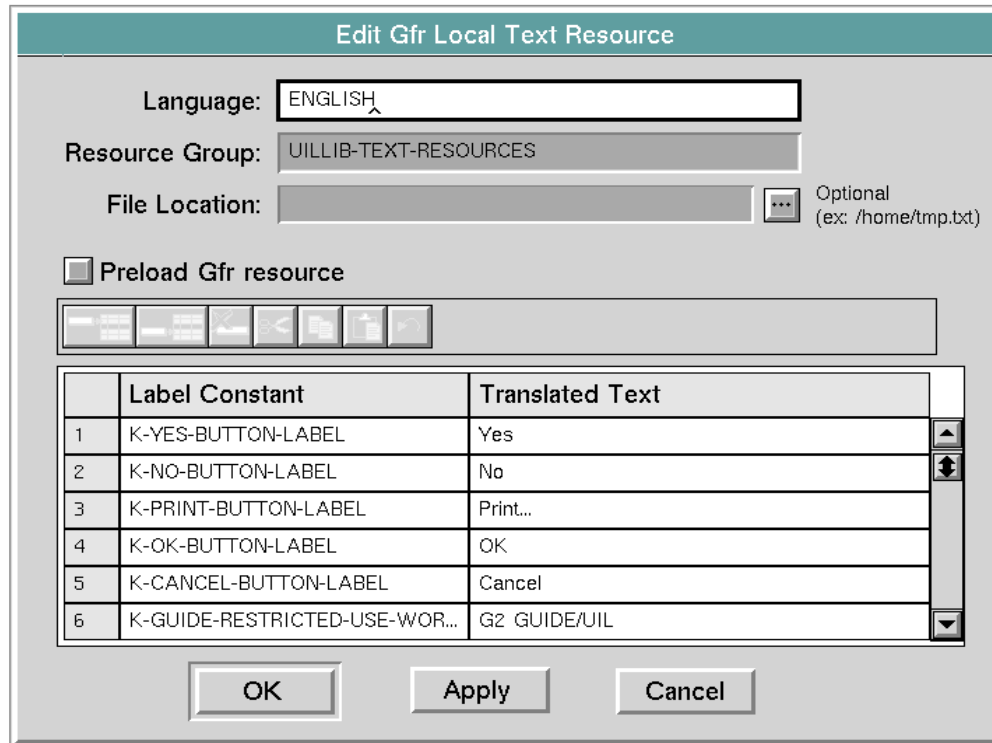
- 1 Use the Inspect utility to go to uilib-text-resources.
- 2 Pull down the menu of uilib-text-resources and select go to subworkspace.

This opens the subworkspace that contains the English language gfr-local-text-resource:



- 3 To create a non-English GFR local text resource, pull down the menu of the English GFR local text resource and select **clone**. A clone appears in the subworkspace next to the English GFR local text resource. (If you are editing English label text, skip this step.)

- 4 Pull down the menu of the `gfr-local-text-resource` and select `edit gfr local text resource` to open the Edit Gfr Local Text Resource dialog:



- 5 To modify a text label:
 - a Select the key in the column labeled Label Constants for the label whose text you want to modify. The constants that you see apply to the text in standard buttons such as OK, Yes, No, Cancel, and Print.

Note You can add constants to a GFR local text resource through this dialog. However, in order to make the text value associated with the constant appear in a button, you must edit the `uil-label-constant` attribute of the text in that button to specify the new constant.

In `GUIDE/UII`, buttons that inherit from `uil-text-button` also have a `uil-label-constant` attribute that references a constant in a GFR local text resource. This is for translating the text of buttons whose text is embedded in the icon description of the button.

- b Edit the text of the label in the edit box under the heading Translated Text.
- c Click Apply to apply your edit, or click OK to apply your edit and close the dialog.

To create a new GFR text resource group:

- 1 Go to `gfr-top-level`, the top level GFR workspace. This workspace is a palette from which you can clone a GFR text resource group:



- 2 Clone the Text Resource Group and the Local Text Resource and drag them to a workspace of your user module.
- 3 To create a non-English GFR local text resource, pull down the menu of the English GFR local text resource and select **clone**. A clone appears in the subworkspace next to the English GFR local text resource. (If you are editing English label text, skip this step.)

- 4 Edit the text in the label constants of this GFR local text resource. To do this:
 - a Pull down the menu of `gfr-local-text-resource` and select `edit gfr local text resource`. This opens the Edit Gfr Local Text Resource dialog.
 - b In the Edit Gfr Local Text Resource dialog, specify constants and text values, and click OK or Apply.
- 5 Edit the master dialogs so that they reference this `gfr-text-resource-group` instead of the default `uilib-text-resources`. To do this:
 - a Select Main Menu > Inspect.
 - b In the Inspect window, enter `go to uil-generic-dialog-class-master`, where *dialog-class* specifies the particular generic dialog master that you want to edit (`query`, `confirmation`, `message`, or `notification`). This opens the workspace containing the master generic dialogs. For example:
`go to uil-generic-query-dialog-master`

- c Pull down the menu of the master generic dialog that you want to edit and select edit dialog. The Edit Dialog dialog appears:

- d In the Edit Dialog dialog, enter the name of the GFR text resource group in the edit box to the right of the label GFR text resource group.

Note The Translate dialog option must be selected.

- e Click Apply to apply your edit, or click OK to apply your edit and close the dialog.

Extending Context-Sensitive Help

You can extend the set of context-sensitive help topics available through G2 Online Documentation (GOLD) to provide help for user-defined UIL classes. You can add help for master dialogs and for UIL controls or other items that appear on master dialogs.

How to Extend Context-Sensitive Help for Dialogs and Items on Dialogs

When a user requests context-sensitive help for a copy dialog or for some item on a copy dialog, GOLD:

- 1 Maps the copy dialog or item to the corresponding master dialog or item. The master dialog and its items reside in an application module.
- 2 Finds the help tags for the dialog or item. The application module that contains the master dialog must provide these help tags through a search procedure.
- 3 Uses the help tags to locate the corresponding section in the online documentation, and then displays this section.

To map a UIL item (subclass of `uil-object` or `uil-message`) on a copy dialog to an item on a master dialog, GOLD uses the ID of the item. When a user requests help for an item on a copy dialog:

- If the item has an ID that matches the ID of an item on the master dialog, GOLD maps the item on the copy dialog to the item on the master dialog.
- If the item has an ID but there is no item on the master dialog with the same ID, GOLD maps the item to the master dialog.
- If the item does not have an ID, GOLD maps the item to the master dialog.

Steps for Extending Help

Note If you have dialog subworkspaces that you created in a version of GUIDE prior to 5.0, you must convert them to `uil-dialog-workspaces`. If you are not providing extended context-sensitive help for UIL dialogs or items, you do not need to convert dialog subworkspaces.

To add context-sensitive online help topics:


- 1 Write a search procedure that returns a help tag for the items for which you want to provide help. For information about how to do this, see the *G2 OnLine*

Documentation Developer's Guide.

- 2 Assign an ID to every UIL control or item on a master dialog for which you want to provide help. The ID must be unique among the items on the master dialog.
- 3 To provide help for a UIL object that is dynamically added to a dialog when the dialog is launched:
 - a Create a subclass of the class of this UIL object.
 - b Write a `gold-map-item-to-context` method for the new subclass that maps the subclass to a class in the application module containing the master dialog. Without this mapping, UIL cannot find a matching control and maps the UIL object to the master dialog.
- 4 To provide help for a non-UIL object on a master dialog, write a `gold-map-item-to-context` method for the class of this object.

The following figure illustrates a `gold-map-item-to-context` method for a class named `my-grobj`:

MY-GROBJ::GOLD-MAP-ITEM-TO-CONTEXT



```

gold-map-item-to-context(l: class my-grobj,
Win: class g2-window) = (item-or-value)
MasterDlg: item-or-value;
Dlg: class item;
begin
  if the workspace of l exists and the item
    superior to the workspace of l exists
    and the item Dlg superior to the
    workspace of l is a uil-tailored-dialog
  then begin
    MasterDlg = call uil-get-master-
      dlg(the id of Dlg);
    if MasterDlg is a uil-tailored-dialog
    then
      return MasterDlg
    else return false;
  end;
  return false
end

```

When a user requests context-sensitive help for an instance of MY-GROBJ, this procedure maps the instance to the master dialog.

For information about how write `gold-map-item-to-context` methods, see the *G2 OnLine Documentation Developer's Guide*.

A B C D E F G H I J K L M
N O P Q R S T U V W X Y Z

A

action: A procedure that performs a specialized operation on a dialog, such as launching or closing the dialog, or updating or concluding the values of the UIL controls on the dialog. When a user clicks on a push button, a set of actions associated with that push button is run on a specified dialog. This can be the dialog that contains the push button, or another dialog. You can edit a push button to specify the actions that are run when a user clicks on it, and the dialog on which the actions are run. Actions have a set of five required arguments that distinguish them from methods and callbacks. **Note:** GUIDE/UIL actions are different from G2 actions in the G2 language.

action description array: *See* uil action description array.

administrator mode: A system-defined G2 user mode that allows you to construct a GUIDE user interface. In Administrator Mode, you can access the menus of all GUIDE objects. *See also* **build mode**.

API: *See* Application Programmer's Interface.

Application Programmer's Interface: A formally defined programming language interface. For G2 GUIDE/UIL, the API is the User Interface Library (UIL).

attribute: A characteristic or property of an item. There are four types of attributes: simple attributes, parameter attributes, which get their values from parameters, variable attributes, which get their values from g2-variables, and object attributes, which contain objects.

attribute displays: A display that shows the values and optionally the names of one or more attributes of an item.

attribute table: A two-column table that shows the name and value of each attribute of an item.

B

border: A graphical object that encloses a workspace, text object or edit box, to provide visual contrast and clarity.

build mode: A system-defined G2 user mode in which you can construct a GUIDE user interface. Selecting objects in Build Mode displays their menus, so that you can configure and move them. Build Mode is the recommended mode for using GUIDE.

C

call: A G2 action used to invoke other G2 procedures. *See also* **start**.

callback: A G2 procedure that is invoked by the handler of a button whenever a user clicks on a push button, radio button, check button, or toggle button. The callback invoked by a button is referenced by the **callback** attribute of that button.

cascaded dialogs: A sequence of dialogs, in which some dialogs can be launched from within other dialogs. When one dialog is opened from within another dialog, the first open dialog is known as the **parent dialog**, and the dialog that is launched from within the parent dialog is called the **child dialog**.

check box: A selection box that corresponds to a particular group of check buttons.

check button: A button used in a group (referred to as a check box) to enable users to select one or more choices. The choices are not mutually exclusive.

child dialog: A dialog that a user launches from within another dialog by clicking on a push button in that dialog. The dialog from which the child dialog is launched is called the parent dialog of the child dialog.

class: A group of items that have the same attributes and the same parent (or superior) class. Classes are organized into a hierarchy in which each class inherits the attributes of its superior classes, but may have additional attributes of its own. In G2, every item is organized into a class.

class hierarchy: A structure defining the inheritance relationships of classes to each other, including both built-in and user-defined classes.

click: A mouse action. A click involves pressing the mouse button and releasing immediately without moving the mouse. You can click the mouse button to display menus, select menu choices, and so on.

clone: A menu choice for some items that copies the item and re-initializes some attributes of the copy. A clone allows you to quickly create a similar item.

combo box: An edit box combined with a scrollable list of text items. Users can display the list by clicking the button in the upper right corner of the Combo Box. When a user selects an item in the list, the list is closed and the selected item appears in the edit box.

confirmation dialog: A system-defined dialog that prompts users to confirm that they want to take an action, such as deleting an object. The dialog contains Yes, No, and Cancel push buttons for responding to the prompt. You can post message dialogs using `uil-post-generic-dialog`.

configuration object: An object that specifies the color of a region or regions of UIL objects. The GUIDE Configuration Editor enables you to select an existing configuration to apply to an object, to create a new configuration object, to edit the attributes of an existing configuration object, or to delete a configuration object.

copy dialog: A dialog that GUIDE creates by cloning a master dialog. Copy dialogs are the dialogs that users see and use when they run a G2 application with a GUIDE user interface. Copy dialogs are permanent objects.

D

default button: A push button on a dialog that is activated when the user presses the Return key. Each dialog can have only one default button.

delay notification: A clock face in the upper left corner of the window, with an optional string of text. You can use the delay notification to tell users that a delay in processing is happening. You can post and remove the delay notification using the procedures `uil-post-delay-notification` and `uil-remove-delay-notification-if-any`.

demo knowledge base (guidemo.kb): A knowledge base, shipped with GUIDE, that contains:

- Working examples of GUIDE dialogs, and other important features of a GUIDE user interface.

You can look at these examples as illustrations of how a GUIDE user interface can be designed. You can also copy and modify examples to use in your own applications.

- An online tutorial that shows you how to use GUIDE to create a user interface

dialog: A subworkspace of a dialog object that displays UIL controls through which users can view and edit class-specific attributes of G2 objects. GUIDE enables you to create templates for customized dialogs. GUIDE also supports the following kinds of system-defined dialogs: message dialogs, confirmation dialogs, query dialogs, and notification dialogs.

dialog bin: A workspace in which GUIDE stores copy dialogs. Dialog bins are designed to enable G2 applications to retrieve copy dialogs quickly and efficiently. Dialog bins are hidden by default.

E

edit box: A field in which an application can display textual information to users, and users can input textual information to the application. User input to edit boxes can be formatted and validated.

edit style: A object that specifies how the editor behaves when it is opened on edit boxes, in all user modes. An edit style can specify editing features such as the language in which the editor displays menus and prompts when it is opened on an edit box, and whether an edit box can display more than one line of text.

F

format object: A reusable object that specifies how text is formatted in edit boxes, message objects, and text objects. Formats can also establish criteria for validating data that users enter into edit boxes.

G

G2 GUIDE: A development tool that enables you to create a graphical user interface for G2 applications. Objects created using G2 GUIDE are a permanent part of your knowledge base.

g2uiprnt.kb: The GUIDE module that contains the print workspace dialog. You can customize printer selection, papersize, orientation, margins, and other print options.

GUIDE palette: A workspace that includes basic tools for developing a GUIDE user interface. These tools include icons that you can select and drop to clone UIL controls, a button that you can click to access the GUIDE Help dialog, the GUIDE Control Panel, the GUIDE Information Dialog, and the More Options palette.

GUIDE Configuration Editor: A dialog that enables you to select an existing configuration to apply to an object, to create a new configuration object, to edit the attributes of an existing configuration object, or to delete a configuration object

GUIDE Control Panel: A workspace that you can access through the GUIDE palette to select a user mode for running GUIDE, and to specify the default window style and size of the UIL controls that you create.

GUIDE Dialog Generator: A dialog that to enables you to generate a master dialog for viewing and editing attributes of a particular user-defined class.

GUIDE Garbage Pail: A icon that you can add to a workspace to enable users to delete dialogs and UIL controls. Users delete the objects by dragging them to the GUIDE Garbage Pail and dropping them on it.

GUIDE Help dialog: A dialog that you can access through the GUIDE palette to:

- Display information about each system-defined UIL procedure. The information includes a synopsis of the procedure's arguments and return values.
- Display information about UIL methods and invoke editors that help you create customized methods and actions to use in place of the system defined methods, actions, and callbacks.
- Invoke the GUIDE Dialog Generator.
- Open a workspace that contains working examples of dialogs and UIL controls. This workspace contains the examples in *guidemo.kb*.

- Run the online GUIDE tutorial, which leads you through the basic steps of creating a user interface with GUIDE.
- Debugging tools

GUIDE Method Help dialog: A dialog that enables you to display general descriptions of methods, actions and callbacks, and to create customized methods, actions, and callbacks.

GUIDE Information Dialog: A dialog that you can access through the GUIDE palette to display the version numbers of GUIDE and UIL that you are running.

guide.kb: The GUIDE module that supports all GUIDE editors and the front-end to all UIL objects.

guicolor.kb: The GUIDE module that contains the color selection dialog.

guidata.kb: The GUIDE module that supports editing lists and arrays or items containing lists and/or array.

guimove.kb: The GUIDE module that contains a move dialog, in which you can make adjustments to the X and Y positioning of any G2 object.

guitools: The GUIDE module that includes the modules guicolor, guimove, guigrf, and guidata.

H

hierarchy of classes: An organization of classes into superior and subclasses to allow for inheritance of attributes and other knowledge. Each class inherits the attributes of its superior classes.

I

inheritance: An important property of object-oriented development environments. A class inherits the attributes of its superior. Inheritance facilitates rapid development, eliminates redundancy in an application, and builds reusable application components. *See also superior class.*

initiating object: The G2 object that launches a dialog. For example, when a user chooses a menu choice from the menu of the G2 object, that object becomes the initiating object of the dialog.

M

master dialog: A template for the dialogs that users see and use when they run your G2 application. The dialogs that users see and use are called **copy dialogs**. GUIDE creates copy dialogs by cloning master dialog that you create.

message dialog: A system-defined dialog that displays a customized message and contains an OK button, which a user can click to dismiss the dialog. You can post message dialogs using `uil-post-generic-dialog`.

message object: An object that contains a text string and appears in scroll areas. You can apply formats to the text in message objects.

method: See UIL method.

More Options palette: A palette that you can access through the GUIDE palette to display icons for the Print Workspace dialog, formats, action description arrays, and field edit styles.

N

navigation button: An iconic button that enables users to navigate from workspace to workspace. When selected, a navigation button performs a task related to displaying or hiding workspaces. The following kinds of navigation buttons are supported: `uil-goto-workspace-button`, `uil-goto-superior-workspace-button`, `uil-goto-previous-button`, `uil-goto-next-button`, `uil-hide-button`, and `uil-help-button`.

notification dialog: A system-defined dialog that displays a string of text that you specify. You can use this dialog to inform users about a delay in processing or any other condition. You can post message dialogs, using `uil-post-notification-dialog`.

O

object definition: Defines a class of objects and is represented by a triangular-shaped icon.

operate on area: An option on the KB Workspace menu that enables users to select an area on the workspace, and then work with the items in that area as a group. For example, a user can clone or align all the items in an area. **Note:** It is strongly recommended that **Operate On Area** *not* be used with GUIDE Version 4.0.

P

parent dialog: A dialog from within which users can launch another dialog, by clicking on a push button. The dialog that is launched is called the child dialog.

play mode: A term that includes any user mode other than build mode or administrator mode. You can create any number of named user modes. Use Play mode when testing or deploying your graphical interface. Selecting objects in Play mode makes them behave as they do in a running application.

Print button: A button that users can click to display a dialog of printing options for the workspace that contains the button.

Print Workspace button: A button that users can click to print the workspace or dialog subworkspace where the print workspace button resides. The print workspace button prints the workspace according to the default settings found in the system tables.

Print Workspace dialog: A dialog in which users can specify options for printing the workspace that contains the Print button.

push button: A button that starts a callback procedure when a user clicks on it. There are two kinds of push buttons: text push buttons and icon push buttons. Push buttons are called stateless button because that carry no values, but instead are used to initiate actions. Text push buttons and icon push buttons differ only in appearance. Each kind of push button comes in three sizes (small, medium, and large).

Q

query dialog: A system-defined dialog that contains a user-defined message, an edit box in which the user can respond to the message, and OK, Apply and Cancel buttons. You can post query dialogs using `uil-post-generic-dialog`.

R

radio box: A selection box that corresponds to a particular group of radio buttons.

radio button: A button that appears in a group (referred to as a radio box) and allows you to make a mutually exclusive selection. It behaves like buttons in a car radio or on a blender. In a radio box, one and only one radio button can be selected. If no radio button corresponds to the value of the radio box, no radio button is selected. Selecting one radio button causes the previously selected radio button in the same radio box to be deselected

relation: A named association between items. You use relation definitions to define relations between classes of items and the conclude action to conclude relations between specific instances. The creation and deletion of relations between items can cause forward chaining to rules. Also, G2 can reason about existing relations between items. Unlike connections, relations are not depicted graphically nor saved as part of a knowledge base.

releasing a dialog: The process of closing a dialog and returning it to the dialog bin, thus freeing the dialog for use by another user. A user releases a dialog by running the action `uil-release-dialog` on the dialog – generally, by clicking on a push button that runs this action.

reserving a dialog: The process of creating and displaying a copy dialog, or displaying an existing copy dialog. To reserve a dialog, you must call the procedure `uil-control-dialog-callback`, either directly, or through calls to the procedures `uil-start-dialog`, `uil-start-or-refocus-dialog`, or `uil-start-dialog-processing`.

Reset menu choice: An option on the Main Menu of a running knowledge base. This option stops the knowledge base from running and re-initializes all of its values.

Restart menu choice: An option on the Main Menu of a running or paused knowledge base. This option stops the knowledge base, re-initializes all of its values, and then starts it again.

Resume menu choice: An option on the Main Menu of a paused knowledge base. This option continues running a knowledge base that had been paused.

returning a dialog: The process of returning control of a dialog to the procedure that launched it, without returning the dialog to the dialog bin. When a dialog is returned to its launching procedure, it is no longer available to users, but the launching procedure can continue to process the information in the dialog. The dialog is not available for reuse by your application until it is returned to the dialog bin by the `uil-release-dialog` action.

S

scroll area: An object that displays a scrollable list of message objects. Each message object contains a text value that can be updated from or concluded to other objects in your G2 application. A scroll area can represent a set of values that is larger than can be conveniently displayed in its entirety at one time.

scroll bar: A bar on one side of a scroll area that a user can click to scroll the contents of the scroll area up or down.

selection box: An object that corresponds to a particular group of buttons. There are two kinds of selection boxes: check boxes and radio boxes. They allow you to operate on all buttons within a particular box.

separator: A horizontal or vertical line that you can add to a workspace or dialog subworkspace to add visual definition.

slider: A graphical object that enables you to display and select numeric values by moving a pointer, known as a **thumb**, along a horizontal or vertical track.

source attribute: The class-specific attribute of a G2 object whose value is reflected in the UIL control.

source object: The G2 object whose class-specific attribute is used as the source attribute of a UIL control.

spin control box: A specialized edit box that enables users to select one value from a range of values by scrolling a list of values up or down within the box.

start: A G2 action that you can use to start other G2 procedures. When executed by a procedure, the started procedure and the starting procedure run concurrently. A start action cannot return values.

Start menu choice: An option on the Main Menu that runs the knowledge base.

subclass: A class subordinate to another in the hierarchy of classes. For example, throttling-valve is a subclass of valve. A class may have any number of subclasses.

subworkspace: A workspace that is subordinate to an item. G2 treats the information on a subworkspace the same as it treats the contents of top-level workspaces. *See also workspace.*

superior class: A class that is at a higher level than another in the hierarchy of classes. Classes inherit attributes from their superiors. You can override and add to inherited attributes. Each class has only one immediately superior class from which it inherits attributes, but all of the other ancestors of the class (the immediate superior of its immediate superior, and so on) are also considered superiors of the class.

sys-mod.kb: The G2 module that supports the system procedure library.

T

target attribute: The class-specific attribute of a G2 object to which the UIL control concludes its value.

target object: The G2 object whose class-specific attribute is the target attribute of a UIL control.

temporary storage object: An object that serves as a buffer during update and conclude actions on the dialog. Temporary storage objects make it possible to process information when it is updated into a dialog or concluded from the dialog to other objects in your G2 application.

text objects: An object that displays read-only text. You can apply formats to text objects.

title bar: A text object that you can add to a workspace or dialog subworkspace to identify the workspace or subworkspace to users.

toggle button: A button that, when selected, switches between two toggle states (ON and OFF). There are two kinds of toggle buttons: text toggle buttons and icon toggle buttons. Text toggle buttons and icon toggle buttons differ only in appearance. Each kind of toggle button comes in three sizes (small, medium, and large)

U

uil.kb: The GUIDE module that provides an API to all UIL objects.

uil-build mode: *See build mode.*

uil-configuration-class: The root class for all configuration objects.

UIL action description array: A object containing a symbolic array of actions that can be run on a dialog. You can invoke an action description array on a dialog by invoking the procedure `uil-control-dialog-callback`.

UIL control: A graphical component of user interface that you construct using GUIDE. UIL provides different classes of UIL controls for components such as edit boxes, scroll areas, buttons, and separators. Each class is appropriate to a particular type of data. For example, edit boxes are suitable for displaying and editing text values, and scroll areas are suitable for displaying lists and arrays of values

uildefs.kb: The GUIDE module that provides definitions for UIL objects.

uil-grmes: The root class for all UIL messages.

uil-grobj: The root class for all UIL graphical objects.

UIL method: A UIL procedure that performs operations required by developers, such as cloning and deleting objects, or operations required by users, such as opening and closing dialogs, and updating and concluding their values. UIL methods are referenced from attributes of the objects on which the methods are run.

UIL Procedure Lookup Facility: A dialog that enables you to display the argument signatures of all UIL procedures. You access the UIL Procedure Lookup Facility through the GUIDE Help dialog.

uilroot.kb: The GUIDE module that supports definitions and API support for navigation buttons only.

user-defined attributes: Attributes provided by G2 users while configuring the `attributes-specific-to-class` attribute of a class definition.

User Interface Library (UIL): A knowledge library providing an application programmer's interface (API) to procedures that perform basic operations on dialogs and UIL controls, including dynamically creating, configuring, and deleting these objects.

user menu choice: A menu choice that you define for a particular class. A user menu choice is visible in an item's menu only when the knowledge base is running and the menu choice's condition is `true`.

user mode: A mode of operation that restricts access to G2 and specifies how the user interface behaves. GUIDE supports three kinds of user modes: Build Mode, Administrator Mode, and Play Mode (named user modes). You select a user mode for GUIDE operation using the GUIDE Control Panel.

@ A B C D E F G H I J K L M
 # N O P Q R S T U V W X Y Z

A

About button
 action buttons used to launch dialogs
 actions

- controlling dialogs with
- creating
- system-defined
- uil-call-conclude-method
- uil-call-conclude-method-for-children
- uil-call-conclude-method-for-parent
- uil-call-configuration-method
- uil-call-update-method
- uil-call-update-method-for-children
- uil-call-update-method-for-parent
- uil-call-validate-method
- uil-delete-dialog
- uil-delete-temporary-storage-object
- uil-hide-dialog
- uil-release-dialog
- uil-release-temporary-storage-object
- uil-return-dialog
- uil-show-dialog
- uil-simulate-play-mode
- uil-unsimulate-play-mode

 adding

- objects to tab dialogs
- tab pages to a master dialog

 appending items to message objects
 arrays

- editing initial contents of
- UIL controls for representing

B

bins

- See* dialog bins

 borders

- accessing menus of
- adding to dialogs, workspaces, edit boxes, or text objects
- editing margins of
- editing on workspaces

- editing stand-alone
- releasing from workspaces
- resizing
- selecting styles of

buttons

- changing placement of tab buttons
- GUIDE
 - conversion of

C

callbacks

- defined
- GUIDE 3.0 functionality for supported in
- GUIDE
 - in GUIDE 3.0 and GUIDE 4.0
 - of check buttons
 - of radio buttons
 - of toggle buttons

 cascaded dialogs

- creating systems of
- initiating object of
- relations among

 case of text
 check boxes

- adding check buttons to
- adding to master dialogs or workspaces
- concluding values of
- editing
- lead buttons of
- menu choices of
- updating values of

 check buttons

- adding to check boxes
- callbacks of
- deleting
- editing
- moving
- resizing

 child dialogs

- concluding values of
- launching
- updating values of

- cloning
 - not supported through operate on area
 - menu choice
 - tab pages
- concluding
 - check boxes
 - edit boxes
 - scroll areas
 - toggle buttons
 - values of UIL controls on a dialog
- Configuration Editor
- configuration methods
- configurations
 - applying to all buttons in a selection box
- copying
- creating
- deleting
- editing
 - of message objects
 - of scroll areas
 - specifying for a particular UIL control
- confirmation dialog
- context-sensitive online help, extending
- converting
 - existing GUIDE buttons
- copy dialogs
 - action for deleting
 - defined
 - making permanent
- Create Gfr Local Text Resource dialog
- Create Gfr Text Resource Group dialog
- Create New Action dialog
- creating
 - new GFR text resource group
- customer support services
- Customize Dialog Actions dialog
 - components of
 - opening

D

- Date & Time Options dialog
 - components of
 - opening
- date formats
- debugging GUIDE applications
- default buttons
- Delay Notification
- deleting

- not supported through operate on area
 - menu choice
 - tab pages
- deleting UIL controls
 - using Garbage Pail
 - using user menu choices
- demo knowledge base
 - accessing
 - merging into an application
 - using
- destination objects
- dialog bins
 - defined
 - setting maximum number of copies in
- dialogs
 - action for releasing
 - action for returning
 - action for showing
 - actions run on
 - adding borders to
 - adding scroll areas to
 - adding title bars to
 - advantages of editing attribute values
 - through
 - allowing multiple copies on same G2
 - window
 - automatically generated
 - bins for storing
 - building masters of
 - concluding values of
 - controlling with actions
 - converting text of
 - creating and deleting permanent copies of
 - creating cascaded dialogs
 - deleting copy dialogs
 - editing behaviors of
 - editing generated
 - editing with Edit Dialog dialog
 - example with English and Spanish
 - versions
 - examples of
 - generating masters for using uil-generate-
 - customized-dialog
 - generic
 - editing text of using GFR
 - kinds of in GUIDE
 - local text resources for
 - hiding
 - initiating objects of
 - internationalization of
 - launching from action buttons

- launching from other dialogs
- launching from push buttons
- launching from rules
- launching from user menu choices
- launching generated dialogs from push buttons
- menu choices of
- on multiple G2 windows
- procedures that launch
- processing contents of before returning to bin
- relations among cascaded
- releasing
- reserving
- returning
- UIL procedures for launching
- updating values of
- viewing attribute values with
- disabling the GUIDE menu bar at startup

E

- Edit Array dialog
- Edit Border dialog
 - components of
- Edit Border Margins dialog
- edit boxes
 - action for validating contents of
 - adding borders to
 - before and after method processing for
 - concluding
 - disabling keyboard navigation to
 - edit styles for
 - editing initial contents of
 - editing using Edit Edit Box dialog
 - editing using the edit. menu choice
 - formatting text of
 - keyboard navigation to
 - menu choices of
 - resizing
 - selecting colors for background and text
 - selecting edit styles for
 - selecting formats for
 - selection method for
 - setting initial contents of
 - single-line and scrollable
 - specifying validation criteria for in formats
 - unselection method for
 - updating
 - validating

- validation methods for
- Edit Check Box dialog
 - components of
 - opening
- Edit Check Button dialog
 - components of
 - opening
- Edit Dialog Actions dialog
- Edit Dialog dialog
 - components of
 - General tab page of
 - Title tab page of
 - Translation tab page of
- Edit Edit Box dialog
 - components of
- Edit Field Edit Style dialog
 - components of
- Edit Format Specification dialog
 - components of
 - opening
- Edit Legal Values dialog
 - opening
- Edit List dialog
- Edit Message dialog
 - components of
- Edit Navigation Button dialog
- Edit Pushbutton Dialog
 - components of
- Edit Radio Box dialog
 - components of
 - opening
- Edit Radio Button dialog
 - components of
 - opening
- Edit Scroll Area dialog
 - components of
- Edit Slider dialog
 - components of
- Edit Source Object & Attribute dialog
 - components of
 - opening
- Edit Spin Control dialog
 - components of
- edit styles
 - applying to edit boxes
 - creating
 - editing using Edit Field Edit Style dialog
 - single line and multi-line
- Edit Target Object & Attribute dialog
 - opening
- Edit Text dialog

- components of
- Edit Toggle Button dialog
 - components of
- edit. menu choice
- Editor Behaviors dialog
 - components of
- embedded objects
 - example of use
 - specify as source or target objects
- Enable User Menu Choices button
- event queues
- examples of GUIDE online
- extending context-sensitive online help

F

- field edit style
 - See edit styles
- Float Formatting Specification dialog
 - components of
- format objects
 - See formats
- Format Specification icon
- formats
 - applying to text objects, message object, or edit boxes
 - creating
 - data validation features of
 - editing
 - for text objects
 - managing case of text with
 - managing quotation marks with
 - specifying date and time formats with
 - specifying legal values with
 - text formatting features of
 - validating edit boxes with

G

- G2 Foundation Resources
 - See GFR
- G2 GUIDE palette
 - creating master dialogs from
 - features of
 - tools provided by
 - UIL controls on
- G2 GUIDE/UIL initialization panel
- g2cuidev module
 - g2cuidev.kb*
- g2uiprnt module

- g2uiprnt.kb*
- Garbage Pail
- General tab page of Edit Dialog dialog
- generated dialogs
 - creating with default UIL controls
 - creating with non-default UIL controls
 - default UIL controls on
 - editing
 - steps for creating
- generic dialogs
 - converting label text of
- GFR
 - creating a new text resource group
 - for GUIDE local text resources
 - internationalizing dialogs with
 - using to edit GUIDE generic dialog labels
- gfr module
 - gfr.kb*
- gfr-local-text-resource objects
 - attribute table of
 - generating
- gfr-text-resource-group objects
 - attribute table of
 - generating
- gms module
 - gms.kb*
- gold module
 - gold.kb*
- goto next navigation buttons
- goto previous navigation buttons
- goto superior navigation buttons
- goto workspace navigation buttons
- guicolor module
 - guicolor.kb*
- guidata module
 - guidata.kb*
- GUIDE
 - buttons
 - converting existing
 - creating
 - new GFR text resource group
 - disabling menu bar at startup
 - license requirements
 - local text resources
 - from GFR
 - online examples
 - online tutorial
 - reinitializing
 - resetting editor
 - reusing submenus
 - setting G2 Minimum Scheduling Interval
 - parameter for best performance of

- starting
- suggestions for using
- verifying initialization status of
- GUIDE Configuration Editor
- GUIDE Control Panel
 - opening
 - opening when in Administrator mode
 - selecting defaults for window size and object style
 - selecting user mode with
- GUIDE Dialog Generator
 - starting
 - steps for using
- GUIDE Help dialog
 - opening
 - using to access help and tools
- GUIDE Information dialog
- GUIDE Method Help dialog
 - creating procedures with
 - displaying descriptions of UIL methods with
 - opening
- guide module
- GUIDE modules
 - g2cuidev
 - g2uiprnt
 - gfr
 - gms
 - gold
 - guicolor
 - guidata
 - GUIDE
 - guide
 - guidelib
 - guidemo
 - guidesa
 - guigfr
 - guimove
 - guislide
 - guitools
 - removing from a KB
 - sys-mod
 - uil
 - uilcombo
 - uildefs
 - uildg
 - uilib
 - uilroot
 - uilsa
 - uilslide
- guide.kb*

- GUIDE/UIL
 - GUI front end of
- GUIDE/UIL procedure library
- guidelib module
- guidemo module
 - guidemo.kb*
 - See demo knowledge base
- guidesa module
 - guidesa.kb*
 - guidlib.kb*
 - guidslide.kb*
- guigfr module
 - guigfr.kb*
- guimove module
 - guimove.kb*
- guislide module
- guitools module
 - guitools.kb*

H

- Help button
- Help dialog
 - See GUIDE Help dialog
- help for UIL methods
 - See GUIDE Method Help dialog
- help navigation buttons
- hide navigation buttons

I

- icon toggle buttons
- initialization panel
- initiating object
 - defined
 - of cascaded dialogs
- Inspect
 - using to get to GUIDE resources
- internationalization of dialogs
 - example with English and Spanish versions
 - how text is translated

K

- keyboard navigation
 - disabling for an edit box
 - specifying behavior of
 - to edit boxes
- knowledge bases
 - g2cuidev.kb*

g2uiprnt.kb
gfr.kb
gms.kb
gold.kb
guicolor.kb
guidata.kb
guide.kb
guidemo.kb
guidesa.kb
guidlib.kb
guidslide.kb
guigfr.kb
guimove.kb
guitools.kb
sys-mod.kb
uil.kb
uilcombo.kb
uildefs.kb
uilib.kb
uilroot.kb
uilsa.kb
uilslide.kb
uiltdlg.kb

L

launching dialogs
 as children of other dialogs
 from action buttons
 from push buttons
 from rules
 from user menu choices
 procedures for
 UIL procedures for
lead buttons
 of check boxes
 of radio boxes
license requirements
lists
 editing initial contents of
 UIL controls for representing

M

master dialogs
 adding check buttons to
 adding radio buttons to
 adding UIL controls to
 closing subworkspaces of
 creating and deleting permanent copies of
 creating from G2 GUIDE palette
 defined
 deleting using Garbage Pail
 generating rules for updating

 generating with default UIL controls
 generating with non-default UIL controls
 icon on G2 GUIDE palette for
 opening subworkspaces of
 showing subworkspaces of
 steps for building
 steps for generating
 user modes required for creating
menu choices
 of check boxes
 of dialogs
 of edit boxes
 of push buttons
 of radio boxes
 of spin control boxes
 of toggle buttons
merging GUIDE
 into a KB that uses an earlier version of
 GUIDE
message definitions
 classes of
 creating customized
message dialog
message objects
 adding to scroll areas
 appending items to
 configurations of
 editing
 editing initial contents of
 formatting text of
 managing size of
 moving
 resizing
 selecting formats for
Minimum Scheduling Interval system
 parameter
More Options palette
 Action Description Array icon on
 Field Edit Style icon on
 Format Specification icon on
 opening
 Print Workspace Dialog icon on
Move menu choice
moving
 check buttons
 message objects
 radio buttons
 scroll areas
 separators
 stack of tab pages
 UIL objects

- UIL objects with borders
- UIL objects with labels

N

- navigation buttons
 - adding to workspaces
 - classes of
 - creating without GUIDE
 - editing using Edit Navigation Button dialog
 - modules supporting
 - use of on dialogs not recommended
- notification dialog

O

- object definitions
 - classes of
 - creating customized
- online examples for GUIDE
- online help, extending
- online tutorial for GUIDE
- Operate on Area menu choice not supported
 - for cloning or deleting

P

- parent dialogs
 - concluding values of
 - updating values of
- permanent dialog copies
- permanent UIL controls
- play mode
 - action for simulating
 - action for unsimulating
 - defined
- Print button
- print workspace button
- Print Workspace Dialog icon
- printing GUIDE workspaces
- push buttons
 - associating actions with
 - associating actions with using Customize
 - Dialog Actions dialog
 - controlling dialogs with
 - creating new actions for
 - editing
 - event queues of
 - launching dialogs from

- menu choices of
- performing operations on dialogs with
- setting target objects for
- target objects of

Q

- query dialog
- quotation marks

R

- radio boxes
 - adding radio buttons to
 - adding to master dialogs or workspaces
 - editing
 - lead buttons of
 - menu choices of
 - updating and concluding
 - updating value of
- radio buttons
 - adding to radio boxes
 - callbacks of
 - editing
 - moving
 - resizing
- reinitializing GUIDE
- releasing dialogs
- reordering
 - tab pages
- reserving dialogs
- resetting GUIDE editor
- resizing
 - borders
 - check buttons
 - edit boxes
 - message objects
 - radio buttons
 - scroll areas
 - separators
 - stack of tab pages
 - text objects
 - UIL objects
- returning dialogs

S

- Scroll Area Options dialog
 - components of
- scroll areas

- adding message objects to
- adding to dialogs or workspaces
- configurations of
- editing using Edit Scroll Area dialog
- managing message size in
- moving
- resizing
- scrolling up or down
- setting maximum number of messages for
- setting options for
- user-defined methods for
- scroll down arrow
- scroll thumb
- scroll up arrow
- Select Edit Style dialog
- Select Format dialog
 - components of
 - opening
- separators
 - adding to workspaces or subworkspaces
 - menus of
 - moving
 - moving and resizing
 - rotating
- Setup Translation Text dialog
- sliders
 - creating
 - editing
 - using
- source attributes
 - defined
 - specifying
- source objects
 - defined
 - how to specify
 - specifying
 - specifying embedded objects as
- spin control boxes
 - menu choices of
- starting GUIDE
- subclasses
 - creating for UIL controls
 - creating instances of
 - deciding what attributes to add to
 - of message definitions
 - of object definitions
 - of uil-message
 - of uil-object
- sys-mod module
 - sys-mod.kb*
- system-defined dialogs

- confirmation dialog
- message dialog
- notification dialog
- query dialog

T

- tab buttons
 - changing placement of
- tab dialogs
 - adding objects to
 - editing
- tab pages
 - adding to a master dialog
 - cloning
 - defined
 - deleting
 - lifting and dropping with non-UIL objects
 - moving stack of
 - reordering
 - resizing stack of
 - transferring
- tab pushbutton
 - defined
- target attributes
 - defined
 - specifying
- target objects
 - defined
 - how to specify
 - of push buttons
 - setting for push buttons
 - specifying
 - specifying embedded objects as
 - specifying programmatically at run time
- temporary storage objects
 - action for deleting
 - action for releasing
 - as source objects
 - as target objects
 - creating
 - defined
 - example of
 - steps for defining
 - when to delete
- Text Formatting Options dialog
 - components of
 - opening
- text objects
 - adding borders to

- editing initial contents of
- editing using Edit Text dialog
- formats for
- formatting text of
- resizing
- selecting formats for
- setting initial contents of
- updating
- text toggle buttons
- time formats
- title bars
 - adding to workspaces or dialog
 - subworkspaces
- Title tab page of Edit Dialog dialog
- toggle buttons
 - callbacks of
 - editing
 - icon
 - menu choices of
 - text
 - updating and concluding
- transferring
 - tab pages
- transferring UIL controls
- Translation tab page of Edit Dialog dialog
- tutorial for GUIDE

U

- UIL controls
 - adding to workspaces
 - creating customized subclasses of
 - default classes for representing different
 - data types
 - defined
 - deleting using Garbage Pail
 - deleting using user menu choices
 - described
 - making permanent
 - moving by dragging
 - moving using the Move menu choice
 - on G2 GUIDE palette
 - setting default sizes and styles of
 - setting default window styles of
 - table of
 - transferring
- UIL Examples workspace
 - opening
- UIL help system file
- UIL methods

- attributes that reference
 - creating
 - displaying general descriptions of
 - displaying help for
 - for application development
 - for runtime operations
 - for validating contents of edit boxes
 - how they work
- uil module
- UIL objects
 - attributes of that reference methods
 - moving
 - resizing
 - specifying the colors of
 - transferring
- UIL Procedure Lookup Facility
 - opening
 - using to display argument signatures of
 - UIL procedures
- UIL procedures
 - displaying argument signatures of
 - examples of
 - for launching dialogs
 - list of in UIL help system file
- uil.kb*
- uil-call-conclude-method
- uil-call-conclude-method-for-children
- uil-call-conclude-method-for-parent
- uil-call-configuration-method
- uil-call-update-method
- uil-call-update-method-for-children
- uil-call-update-method-for-parent
- uil-call-validate-method
- uilcombo module
- uilcombo.kb*
- uil-control-dialog-callback
- uildefs module
- uildefs.kb*
- uil-delete-dialog
- uil-delete-temporary-storage-object
- uildlg module
- uil-generate-customized-dialog
- uil-hide-dialog
- uil-initialization-status parameter
- uilib module
- uilib.kb*
- uilib-text-resources object
 - in GUIDE
 - for editing label text
- uil-release-dialog
- uil-release-temporary-storage-object

- uil-return-dialog
- uilroot module
 - uilroot.kb*
- uilsa module
 - uilsa.kb*
- uil-show-dialog
- uil-simulate-play-mode
- uilslide module
 - uilslide.kb*
- uil-start-dialog
- uil-start-dialog-processing
 - uiltdlg.kb*
- uil-unsimulate-play-mode
- unselection method
- updating
 - check boxes
 - edit boxes
 - radio boxes
 - scroll areas
 - text objects
 - toggle buttons
- upgrading
 - label text of generic dialogs
- User Interface Library
 - See* GUIDE/UIL
- user menu choices
 - enabling and disabling for GUIDE/UIL objects
 - launching dialogs from
- user modes
 - for using GUIDE
 - required for creating master dialogs
 - selecting

V

- validation
 - creating customized procedures for of edit boxes

W

- window styles of UIL controls
- workspaces
 - adding borders to
 - adding navigation buttons to
 - adding separators to
 - adding UIL objects to
 - editing borders of
 - printing
 - releasing borders from