

Integrity

User's Guide

Version 5.0 Rev. 0



Integrity User's Guide

July 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: DOC038-500

Contents

Preface xi

About this Guide xi

Audience xii

A Note About the API xii

Conventions xii

Related Documentation xiv

Customer Support Services xvi

Chapter 1 Overview 1

Introduction 1

 Integrity Core Services 2

 Discovery Import Tools 4

 Reasoning Engines 4

Installing Integrity 4

Features and Benefits 4

The Integrity Core Services 6

 What is a Domain Map? 7

 What is a Message Base? 9

 What are Reasoning Routines? 9

 What are Completion Routines? 10

Handling Events 10

Building an Application 12

The Basic Components of G2 12

 What is a Knowledge Base? 13

 What is an Object? 13

 What is a Workspace? 14

 What are Modules? 14

 What are Classes? 15

Integrity Bundle 15

Chapter 2 Running Integrity 17

Introduction 17

Starting the Server and Connecting the Client 18

Connecting to a Specific Server at Startup 19

 Connecting the Client to the Default Server 19

 Starting the Server on a Specific Port 20

 Connecting the Client to a Specific Server 20

Starting the Server with Your Application Loaded 21

Exiting Integrity 21

Chapter 3 Working with Models 23

Introduction 24

Summary of Common Tasks 24

Using the Project Menu 25

 Using the Project Menu 25

 Using the Manage Dialog 25

 Using the Project Submenus 27

Navigating Applications 27

 Using the Navigator 27

 Searching for Objects 29

Interacting with Workspaces 30

 Displaying a Detail Workspace 30

 Hiding a Workspace 30

 Deleting a Workspace 31

 Editing Workspace Properties 31

 Scaling a Workspace 31

 Shrink Wrapping a Workspace 32

 Showing the Superior Object of a Detail Workspace 32

 Printing a Workspace 32

 Saving a Workspace to a JPEG File 33

 Loading Background Images 33

 Creating and Accessing Top-Level Workspaces 33

Using the Menus 34

 Using the File Menu 35

 Using the Edit Menu 35

 Using the View Menu 36

 Using the Layout Menu 38

 Using the Go Menu 39

 Using the Project Menu 40

 Using the Workspace Menu 42

Using the Tools Menu	42
Using the Help Menu	43
Using the Integrity Toolboxes	44
Using the G2 Toolbox	46
Interacting with Objects	46
Selecting Objects	46
Cutting, Copying, Pasting, and Deleting Objects	47
Controlling the Layout of Objects	47
Displaying the Properties Dialog for an Object	48
Resizing an Object	48
Editing Icon Color Regions	48
Using the Toolbars	49
Standard Toolbar	49
Web Toolbar	50
Layout Toolbar	51
Integrity Toolbar	52
Status Bar	52
Switching User Modes	53
Configuring User Preferences	54
Specifying User Preferences for Different Types of Users	54
Configuring User Preferences	56
Delivering Messages by Email	59
Starting the G2 JMail Bridge Process	60
Creating, Configuring, and Connecting the JMail Interface Object	60
Configuring Integrity to Send Email Messages	63
Configuring Startup Parameter for Sending Email Messages	64
	64
Chapter 4 Customizing the Application	65
Introduction	65
Constructing an Operator Menu	65
Selecting a Menu Template Workspace	67
Enabling List and Array Editing	67
Creating a Text Resource and a Local Text Resource	68
Creating a Menu Bar Template	69
Configuring a Menu Template Item	69
Adding the Menu Selection Icon	70
Cloning a Menu Group	71
Creating a Cascade Menu Item	73
Creating a Show Workspace Menu Item	73
Creating a Choice Menu Item	74

Displaying Browsers from a Menu 75

Connection Menu Items 76

Compiling the Menu 77

Defining Initializations 78

Creating an Initialization for a New Item 80

Editing the Value of an Initialization 80

Setting Preferences 81

Tip of the Day Preferences 82

Load Options Preferences 82

Save Options Preferences 83

Message Browser Preferences 83

Desktop Layout Preferences 84

Finder Options Preferences 86

Navigator Button Preferences 87

Creating New Palettes 87

Add a Palette Group 88

Add a Palette 89

Adding Palette Items 89

Property Files 90

Customizing the User Interface Using Cyberformer 91

Understanding Properties Files 91

Origination and Purpose 91

What are Property files? 91

Scope 92

Location 93

UI Structure 93

Functionality 94

CyberFormer.properties 94

registeredPlugins.properties 95

Keywords 97

Cyberformer Reference 98

Syntax 98

Chapter 5 Getting Started 131

Introduction 131

Creating a New Application 132

Using the Integrity Setup Dialog 133

Importing Management Information Base (MIBs) 133

Process PPD File 135

SNMP Setup 136

Domain Import 137

ODBC Setup and Import 137

Import from Translayer 138

	Import from MS Visio ENT	138
	Importing from other ODBC Sources	138
	Configuration of the ODBC Import	138
	HPOV Setup and Import	138
	Building a Simple Domain Map	139
	Creating Domain Map Subclasses	140
	Creating Domain Objects	141
	Working with Modules	141
	Creating Modules	142
	Merging Modules	142
	Renaming Modules	142
	Saving Modules	142
	Deleting Modules	143
	G2 Mode	143
	Working with G2 Objects	144
	Other Integrity Modules	144
	Adding Integrity Functionality to an Existing Application	145
	Out-of-Box Functionality	145
	Auto-Clearing	145
	Time-Based Events	146
Chapter 6	Handling Events	147
	Introduction	147
	Setting up an External Interface	148
	Processing Unsolicited Events in the External Bridge	150
	Parsing in the External Bridge	151
	Performing Low-Level Filtering in the Bridge	151
	Interpreting the Event in the Internal Bridge	152
	Relating an Event to the Domain Objects	152
	Defining Completion Routines	153
	Automatic Trap Processing	154
	Processing The Trap	154
	Generic Trap Completion Procedure	154
Chapter 7	Creating a Domain Map	155
	Introduction	155
	The Components of a Domain Map	156
	Containment Objects	156
	Managed Objects	157
	Connections and Connection Posts	157

Defining Domain Map Subclasses	158
Viewing Attributes of a Subclass	159
Adding Attributes to a Subclass	160
Displaying Attributes for a Subclass	161
Creating Icons for Domain Object Classes	161
Creating Patterns for Connections	162
Adding Connection Stubs to Class Definitions	163
Importing Class Definitions	163
Manually Building the Domain Map	164
Creating Domain Objects	165
Naming Domain Objects	166
Connecting Domain Objects	167
Adding and Deleting Connection Stubs from Instances	167
Creating a Connection Configuration Object	168
Using a Stub to Create a Connection	168
Using Connection Posts	169
Importing and Exporting a Domain Map	171
Exporting a Sample Domain Map	171
Importing a Sample Domain Map	172
Using Translation Objects	173

Chapter 8 Message Handling 177

Introduction	177
Setting up the Message System	178
Defining Message Servers	179
Defining Browsers	180
Creating Browser Templates	180
Configuring a Browser	182
Creating and Configuring Subscribers	183
Creating and Configuring Filters	184
Defining the Sorting Characteristics of the Browser	185
Configuring the Columns of the Browser	187
Arranging the Items on the Browser Template	193
Writing Custom Procedures to Display and Hide a Browser	193
Defining Status Bars	195
Writing Custom Procedures to Display and Hide a Status Bar	198
Defining Escalation Specifications	200
Specifying the Target of an Escalation Specification	202
Specifying the Category of an Escalation Specification	203
Specifying the Priority of an Escalation Specification	203
Specifying the Procedures Called in Escalation Specifications	203
Timing the Invocation of Escalation Phases	203
Working with Messages	205
What is a Message?	205

	Creating a Message	207
	Maintaining Message Histories	209
	Using the Browser to View and Interact with Messages	210
	Acknowledging Messages	212
	Deleting Messages	213
	Reading and Writing Messages to a File	214
	Message Alarm Propagation	215
	Setting Priority and Acknowledgment Colors	215
	Setting Default Message Priorities	216
	Logging Messages and Events	217
	Creating a Logging Manager	218
	Logging Messages	219
	Logging Events Programmatically	220
	Defining Closing Times for a Log File	220
	Customizing the Logging Manager	221
	Error Handling	221
	Creating a New Error Handling Procedure	222
	Logging System Errors	223
	Creating User Defined Effects	223
Chapter 9	Reasoning About Events	225
	Introduction	225
	Examples of Reasoning Routines	226
	Creating Reasoning Routines	227
	Searching for Related Messages	227
	Querying Message Histories	228
	Filtering Messages and Events	229
	Filtering Duplicate Messages	229
	Filtering Based on Past Events	230
	Filtering Events that Occur in Pairs	230
	Filtering Events Based on Domain Relationships	231
	Filtering Events in the Bridge	232
	Implementing Alarm Thresholding	232
	Correlating Events	234
	Diagnosing Faults	235
	Automating Recovery and Preventing Faults	235
	Index	237

Preface

Describes this document and the conventions that it uses.

About this Guide	xi
Audience	xii
A Note About the API	xii
Conventions	xii
Related Documentation	xiv
Customer Support Services	xvi



About this Guide

This guide describes how to use Integrity Core Services and OPAC features of the Integrity product family of Network, System, Service, and Application Management tools and applications.

This guide includes:

- User information about the Integrity Core Services and OPAC capabilities and components.
- Examples that illustrate the use of the Integrity Core Services and OPAC.
- Task-oriented sections to help you use the information in this book to develop effective Integrity applications.

Audience

This guide is written for application developers and system integrators. Because Integrity uses the G2 programming environment, a previous knowledge of G2 is helpful but not essential. For descriptions of advanced G2 topics refer to the *G2 Reference Manual*.

This guide addresses the application developer as *you*, and refers to the end user as *the user*.

A Note About the API

The Integrity Core Services and OPAC API, as described in this guide, is not expected to change significantly in future releases, but exceptions may occur. A detailed description of any changes will accompany the Integrity product release that includes them.

If Integrity Core Services and OPAC do not seem to provide the capabilities that you need, contact Gensym Customer Support at (781) 265-7301 for further information.

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

Convention Examples	Description
Main Menu > Start	G2 menu choices and button labels
KB Workspace > New Object create subworkspace Start Procedure	
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
<i>text-string</i>	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 Properties	GMS and native menu choices
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

Integrity

- *Integrity User's Guide*
- *Integrity Utilities Guide*
- *SymCure User's Guide*

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/UII Procedures Reference Manual*

G2 Developers' Utilities

- *Business Process Management System User's 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-OPC Client Bridge User's Guide*
- *G2-PI Bridge User's Guide*
- *G2-SNMP Bridge User's Guide*
- *G2-HLA Bridge 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

Overview

Contains an overview of the Integrity family of products, describes the architecture and basic operation of Integrity, and presents an outline of the documentation provided with Integrity.

Introduction	1
Installing Integrity	4
Features and Benefits	4
The Integrity Core Services	6
Handling Events	10
Building an Application	12
The Basic Components of G2	12
Integrity Bundle	15



Introduction

The Integrity (formerly Operations Expert or OpEx) product family comprises Network, System, Service, and Application Management tools and applications. Integrity has three key features:

- [Integrity Core Services](#)
- [Discovery Import Tools](#)
- [Reasoning Engines](#)

The knowledge base (.*kb* file) that loads all of the functionality is:

integrity.kb

To load Integrity on Windows:

- ➔ Choose Start > Programs > Gensym G2 2011 > G2 Integrity > Start G2 Integrity Server.

To load Integrity on UNIX:

- ➔ Change to the *integrity/bin* directory and enter the following at the command prompt:

StartServer.sh

Integrity Core Services

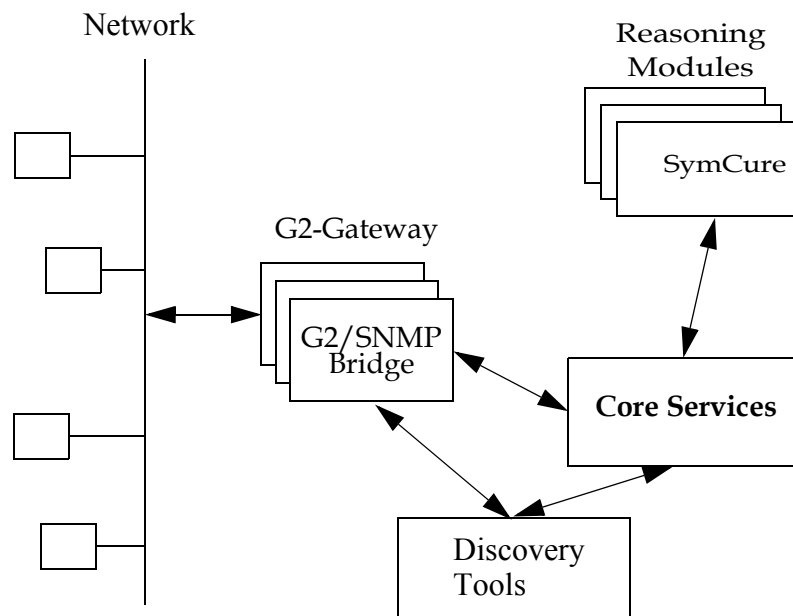
Integrity Core provides foundational components for network and system management applications. Integrity Core contains the Integrity foundation classes, a message management system, tools for building a representation of your managed objects, and a set of utilities. It includes a graphical programming language and G2-Gateway Products for interfacing external systems to Integrity. Also included are industry- and vendor -specific libraries that provide classes, objects, and tools specific to certain domains or vendors.

The specific components of Integrity Core includes:

- Integrity Core Services – The foundation of the Integrity platform. Integrity Core Services provides essential capabilities for automating problem resolution, which are leveraged and extended in the other Integrity packages.
- OPAC (Operator Action) – Integrity’s general-purpose graphical language. OPAC provides graphical representation of operational procedures often outlined in workflow procedure diagrams. Using "clone (by drag and drop from palettes), connect, and configure" visual programming, OPAC facilitates rapid deployment of new systems that require minimal customization.
- G2-SNMP Bridges – Bridges between Integrity and SNMP.
- G2 Java Socket Manager – Bridges between Integrity and Java sockets.
- MPE – Message Parser Engine
- ODiE – OpEx Dispatch Engine
- GMIB – Processes MIB files
- GTRAP – SNMP Trap management

- Ping Manager – Provides ping management to network devices according to user specifications through a graphical interface.
- DSM – Domain Synchronization Manager synchronizes updating of the domain map for additions and deletions of domain objects.

The diagram below shows the components of Integrity:



Integrity Core Services encompasses a set of modules that are designed to help companies monitor and control their operations to increase the availability and service levels of their distributed, mission-critical environments. These modules contain several demonstrations to assist you in becoming familiar with them. These demonstration modules include sample class libraries for some commonly-used equipment and software. The Integrity family of products continues to expand to respond to evolving technologies.

In addition to the structure built into Integrity, the power of the high-level programming environment, G2, makes it possible to solve non-standard problems, configurations, and situations that lie outside the scope of most "out-of-the-box" solutions.

Discovery Import Tools

Discovery import tools included in Integrity allow you to import your domain in several different ways. You have the following import options:

- Regenative's Translayer
- Microsoft's Visio Enterprise Network Tools (ENT)
- HP OpenView Map Importer
- Generic SQL Importer

Each of these import tools represents the data a little differently. Please refer to the documentation for these tools on how each product represents the discovered elements.

Reasoning Engines

Integrity ships with the SymCure reasoning engine. SymCure allows you to develop cause-and-effect models that are represented at the class level. This allows a single model to reason over multiple instances of a class. Please refer to the *SymCure User's Guide* for details.

Installing Integrity

To install Integrity, follow the directions on the distribution CD. Enter the key number located on the Integrity CD when requested by the Install Shield. This key authorizes your Integrity package.

Features and Benefits

The Integrity modules provide the tools for performing alarm and event filtering, correlation, and diagnosis including:

- Early detection of problems from event patterns, reducing the time spent with unrecognized problems.
- Suppression of repetitive alarms (filtering), reducing operator overload, so operators can more quickly notice and react to real problems.
- Grouping of related alarms (correlation), further reducing operator overload.

- Pinpointing the causes for events (diagnosis), which:
 - Reduces downtime by speeding the time to start corrective actions.
 - Reduces the mean time to repair by providing more accurate analysis of the exact problem.
 - Prevents excessive testing and retesting.

OPAC is also used to automate procedures in these areas:

- Testing for diagnostic and filtering purposes
- Enforcing standard automated or manual procedures
- Guiding operators
- Mitigating faults and resolving problems
- Operator interface procedures such as alarm management

In general, automated procedures assure faster testing, eliminate delays in starting and carrying out the steps of corrective action, and enable faster recognition of fully recovered status.

Integrity Core Services also provides:

- General infrastructure for network and message-based applications, (for example, configuration management of objects, topology, and hierarchy).
- Online information and help.

Overall, Integrity improves the availability of all the applications running on your networks. In addition to these run-time benefits, Integrity tools increase productivity. These tools provide an End User, Value Added Reseller (VAR), or System Integrator (SI) with an environment that:

- Reduces application development time.
- Promotes development of reusable objects.
- Allows deployment of the application across diverse hardware platforms.

Applications developed with Integrity are:

- Easy to modify, thereby decreasing the time required to respond to customer requirements.
- Easy to maintain, increasing the profitability of the application.

The Integrity Core Services

At the heart of the entire Integrity family is Integrity Core Services. The core services consist of:

- **Foundation classes** - used to create objects that represent your managed equipment, and also, utility objects. You can specialize these classes to represent specific types of objects.
- **Domain Mapping** - allows you to represent the specific external equipment you are managing in your system, as well as the containment and connectivity relationships between them. These objects are instances of classes you derive from the foundation classes. The map provides a visible and useful user interface with easy navigation through the containment hierarchy, in addition to providing an object repository.
- **Domain Map Importer and Exporter** - lets you import a text file that describes the names and physical relationships among the managed equipment. You can export a domain map to a text file and update the domain map from a text file.
- **Message Base** - The message base is a collection of messages, their histories, and their relationships to the domain objects. A message stores information about an incoming event. It stores information about the sender of the event, the object associated with the event, the type of event, and the actual content of the information sent with the event. Messages are created and stored in message servers. When a message is created, a message history is maintained, allowing you to reason about new events based on the history of related events. The message history query facility provided is central to many alarm correlation applications. The message system propagates message (alarm) priorities and acknowledgment status to the domain map for display of status animated by colors.
- **Browsers** - Integrity provides a facility for creating and configuring browsers for viewing and interacting with messages. Interactions include acknowledgment, deletion, and addition of user comments. Browsers can be configured to subscribe to messages published by any number of message servers and can provide extensive filtering capabilities.
- **Utilities** - Utilities are included for:
 - logging information to a log file.
 - traversing the domain object network.
 - building and maintaining menus.
 - building and maintaining palettes of objects for cloning commonly-needed domain objects.

An Integrity application models a collection of objects in the real world. These objects are referred to as **external objects** to distinguish them from the objects used in the Integrity application which are, in fact, a representation of the external objects. These external objects can be actual physical objects, software processes, databases or any other collection of items linked together to form a system.

The external objects in this system send information about themselves and about other external objects to a manager linked to the system. Information sent from an external object is referred to as an **event**. The events can go directly to the Integrity application or through another managing layer. For example, a data network can use a manager such as Hewlett-Packard's Open View, which passes events into Integrity where filtering, correlation, and other reasoning occurs. The processed events can be displayed with Integrity, or information can be sent back to the manager for display or operator response.

Before you look at how the incoming events move through the system, it is helpful to understand the relationships among the key functional components of Integrity. These components are:

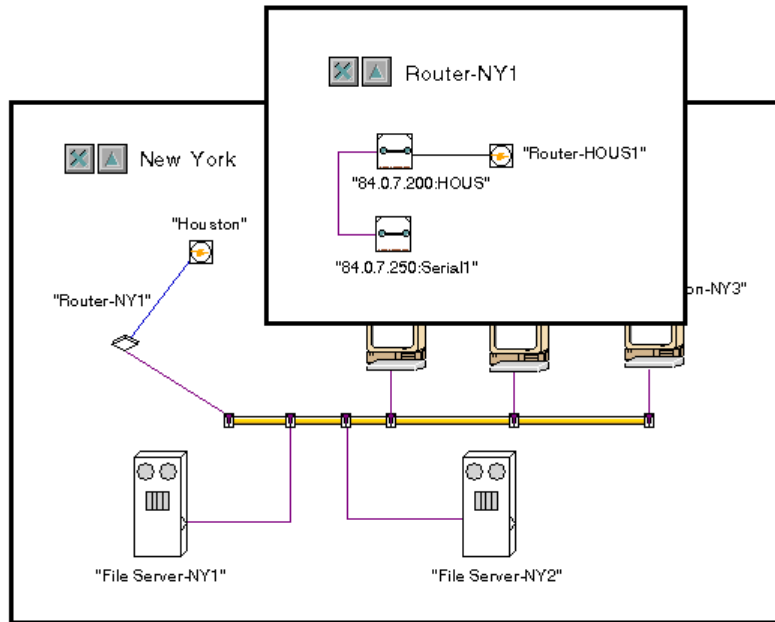
- The domain map
- The message base
- Reasoning routines
- Completion routines

Each of these components is described in the following sections.

What is a Domain Map?

In an Integrity application, the external objects and their relationships are represented in the application by using a set of objects called a **domain map**. For example, a domain map that represents a network might contain external objects such as servers, routers and workstations. The domain map would represent the

objects in the network and would represent the relationships between the objects. The figure below shows part of a domain map:



Each domain object represents an external object. The connections between domain objects mirror the actual connections among the external objects. When an external object contains other objects, a **containment** relationship exists between the objects. In the domain map, you maintain containment relationships by placing contained objects on subworkspaces under the domain objects that contain them. In the example shown above, the smaller window is a subworkspace of the domain object **Router-NY1**. The external object represented by **Router-NY1** contains the serial card shown in the small workspace. The connectivity and containment relationships in the domain map provide important information used for reasoning about the external system.

The individual objects in the domain map are instances of subclasses you create using Integrity Foundation Classes. These define the relationship between objects with similar characteristics. The characteristics can be functional, operational protocols applied to the objects, or a set of attributes shared by the objects.

Integrity provides a User Interface for creating and managing object classes used to create the domain objects. When you start a new application, Integrity creates a complete working environment. This environment contains the Integrity top-level classes and a set of workspaces that organizes the objects you create within your application. How to build a domain map is described in [Creating a Domain Map](#).

What is a Message Base?

A message base is a collection of messages, their histories and their relationships to the domain objects. Any event entering an Integrity application can be used to generate an Integrity message. Integrity messages have the following characteristics:

- Messages relate the event to the domain objects. Relationships are formed between the message and both the target, sender, and category of the event. The **target** of an event is the object described by the event. The **sender** of an event is the object that sends the event. The **category** of an event defines the type of event. The relationship between the event and the target is displayed graphically on the objects contained in the domain map. Objects change color to indicate the highest priority message targeting the object and the acknowledgment status of the object. These graphical message displays propagate up through the containment hierarchy.
- Each unique message can have a history associated with it. A message is considered unique when it has a unique target, sender, and category. Each time a message is created with the same target, sender, and category, the time of the new message is added to the message history.

The domain map provides each message with a context. The message system provides each object with a history of the events that have targeted it.

What are Reasoning Routines?

Reasoning routines contain knowledge about how events should be handled. This knowledge applies the historical and contextual information of the message information base and domain map to the incoming events to make intelligent decisions about the event itself and the status of the system. Since an Integrity application can both monitor and control external objects, this intelligent reasoning can have wide-ranging effects.

Actions that can be taken in response to an event can include one or more of:

- Creating a message from the event that is displayed in the Integrity application or on another system.
- Logging the event.
- Discarding the event.
- Querying the external objects or their manager for further information.
- Taking an action in response to the event.

Reasoning routines declared as methods of domain classes are also directly related to the knowledge implicit in the organization of the class hierarchy. This adds a means of organizing and applying knowledge within the application.

You can create reasoning routines either by using G2 procedures and methods, or by using the OPAC graphical programming environment. For a description of how to create a reasoning routine see [Creating Reasoning Routines](#).

What are Completion Routines?

Completion routines are the methods and procedures that provide the threads that pull all of the other parts of the system together. The completion procedures finish the reception of the external event, relate the event to the domain objects, and determine how the event is initially handled. In the completion routines, you might decide to act directly based on what is already known about the event, or you might call a reasoning routine to do further processing of the event.

You can define completion routines as methods of the domain classes. Use of completion routines is entirely dependent on the design of your particular system. If you do not organize completion routines and reasoning routines by using the domain hierarchies, you must define a methodology for relating incoming events to their completion routines and reasoning routines.

You can create completion routines by using G2 procedures and methods. Completion routines are described in [Defining Completion Routines](#).

Handling Events

Once you have created a domain map, you can interface the external objects to Integrity and bring the events from the external objects into the Integrity application. You can make external interfaces using software called a **bridge**. You can purchase bridges from Gensym or build them using Gensym's G2 Standard Interface (GSI) product. A bridge consists of an external part, usually written in C or C++, and an internal part, which is a part of the Integrity application used to bring the external events into G2.

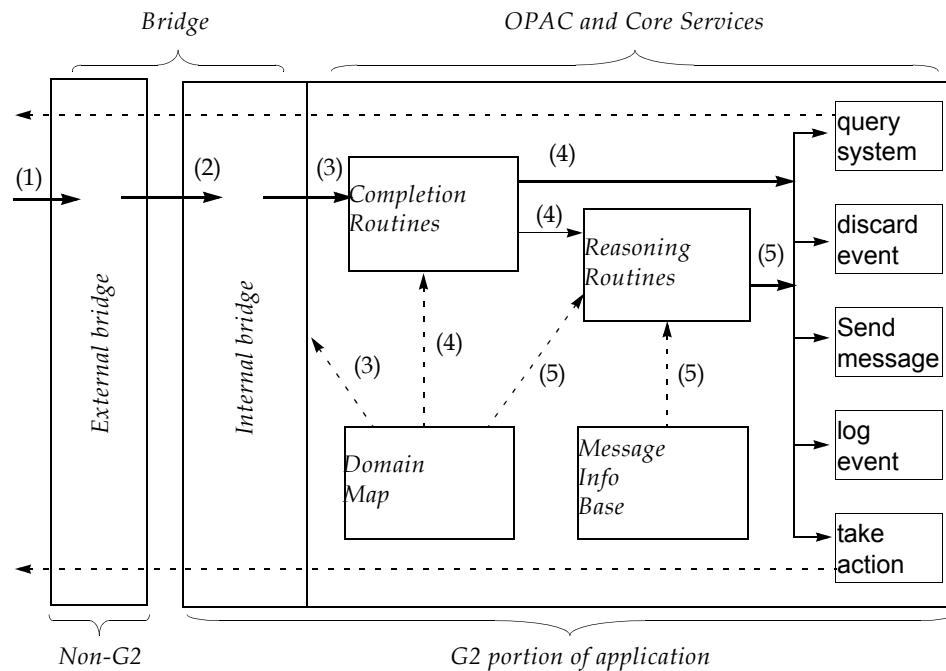
The function of the bridge is to parse the event to determine these key pieces of information:

- **Sender** - the domain object that sent the event
- **Target** - the domain object to which the event applies
- **Category** - the type of event that occurred

These three pieces of information are what define a unique event. You can use them to relate the incoming event to the objects in the domain map that match the sender and the target of the event.

The following diagram shows the components of the Integrity application along with the bridge processes. The arrows show the progress of an event through the

system. The dotted lines show information used from a component in a given step.



The event moves through an Integrity application as follows:

- 1** An event enters the system through the external bridge. The external bridge parses and decodes the information contained in the event.
- 2** The event is passed to the portion of the bridge internal to the Integrity application. Here any further parsing and decoding occurs as needed.
- 3** The internal bridge calls the appropriate completion routine. You can use the class hierarchy of the domain objects to select a completion method to handle the event. If completion methods are not defined in the hierarchy, some alternate coding method must be designed to select the proper completion routine.
- 4** The completion routine sometimes completes the parsing and decoding of the event. In the completion routine you can discard the event, create a message from the event, or call a reasoning routine to further process the event. The completion routine is a G2 procedure or method.
- 5** The reasoning routine uses the connectivity and containment relationships in the domain map and the message histories to reason about the event. The reasoning routine is a G2 procedure or method or an OPAC graphic procedure.

Building an Application

Before you actually begin to build your application, you need to define the following basic system requirements:

- The modules required.
- The class structure of the domain.
- How many levels of priority are required.
- How to receive and parse events.
- The kinds of filtering and correlation that need to be done.
- How the end user interface should appear and behave.

Assuming that the basic design and end result have been determined from a project point of view, the steps below provide an overview of the mechanics to implement the design. Each step contains a reference to the applicable section of the manual.

- 1 Create the top level module and module hierarchy as described in the Introduction.
- 2 Build a domain map as described in [Creating a Domain Map](#). Create a sample of the domain on the 'domain map' workspace.
- 3 Set up the number and color of priorities.
- 4 Set up the interface to bring the events into the system. Although this is highly specific to each application, [Handling Events](#) describes the general principles for bringing events into the application.
- 5 Create supporting completion routines.
- 6 Create messages from external events to test the input of events from the external system.
- 7 Set up any required logging.
- 8 Add reasoning routines and/or OPAC procedures for correlation and evaluation. For a discussion of reasoning routines, see [Reasoning About Events](#).

The Basic Components of G2

Integrity gives you full access to the programming power of the G2 environment. This document does not attempt to provide detailed descriptions of G2's many features. This section provides a basic orientation to some of the important elements of the G2 environment. For complete documentation on G2, refer to

G2 Reference Manual. The tutorial manual, *Getting Started With G2 Tutorial*, is also an excellent way to learn the G2 basics.

What is a Knowledge Base?

G2 applications are stored in knowledge bases. A **knowledge base**, or **KB**, is an ASCII file with a `.kb` extension that contains all of the information your application needs to run. G2 knowledge bases, also known as G2 **applications**, can have a single file or many files. Knowledge comes in many forms in G2:

- **Objects** that represent the physical systems in your application and the connections between them.
- **Definitions** that describe the common features of the objects.
- **Rules, methods, and procedures** that describe the behavior of the objects in the real-time environment.
- **Graphical user interface** components that enable end users to interact with the application.

What is an Object?

G2 is an **object-oriented** development environment. This means that G2 represents certain kinds of knowledge as objects in the application.

An **object** is a piece of information that contains all related knowledge about that object in one location. The object contains all the data that defines the object, and all the operations that the object can perform. In object-oriented terms an object's data are called its **attributes**, and an object's operations are called its **methods**.

An object has particular attributes based on its type. In object-oriented terms, an object's type is called its **class**. For example, the attributes of a video teleconferencing site might be its location, network type, number of connected sites, and connection status.

Using object-oriented techniques, you can create classes of objects that share characteristics with other classes. This technique is called **inheritance**, where the **subclass** of an object inherits all of the characteristics of its **superior class**, including all of its attributes and methods. In the subclass, you describe only the unique features of the class.

Objects represent a powerful way of organizing knowledge and avoiding redundancy in an application, by describing objects with similar characteristics in a single place.

A G2 application describes the behaviors of its objects, reasons about those objects, and provides expertise about those objects in a real-time environment. For example, in a video teleconferencing application, you might describe the behaviors of an office when it is actively connected to another office, and you

might reason about whether the site is over budget while it is connected in real time.

Each object in the knowledge base has an **icon** representation. This means you can use objects to communicate information graphically to the end user, for example, by animating the icon to reflect its status. You can create your own icon representation of an object, or you can use one of the many available icons in the G2 icon library.

In G2, almost all knowledge is represented as an object, including:

- The physical systems
- The connections between systems
- The rules and procedures that describe the behavior of the systems
- The workspaces on which objects exist
- The graphical user interface elements of the application

What is a Workspace?

G2 calls the “blank pages” upon which you create and maintain objects **workspaces**. A KB can contain one or many workspaces. The objects upon workspaces are capable of having their own subsidiary workspaces. Thus, you can create a logical hierarchy of objects and workspaces to group and organize your KB data. Workspaces can contain anything from text messages to entire schematics that model real-time activity.

What are Modules?

G2 applications typically consist of numerous KB files, each of which contains one or more modules. A **module** is a set of related information contained in the KB. For example, an application might have two modules, one for the object definitions, expert system rules, and procedures that describe behaviors, and another for the graphical user interface components.

Modules represent a powerful way of organizing your application, as well as reusing existing knowledge across G2 applications.

Depending on the needs of the application, some modules can be **stand-alone modules**, which means they can run independently of the other KB files in the application, while other modules are **dependent modules**, which means they require additional information contained in one or more other modules to run.

G2 represents the modules of an application in a hierarchy to show the module dependencies. The module at the top of the hierarchy is called the **top-level module**. If the application has lower-level modules, the top-level module is by definition a dependent module. Modules at the bottom of the module hierarchy

are by definition independent modules, because they do not depend on any other module in the hierarchy.

The name of individual KB files typically corresponds to the name of the modules in the application.

What are Classes?

G2 development is based on object-oriented design. Knowledge representation is maintained and extended through classes in the G2 class hierarchy. G2 includes a large set of system-defined classes, many of which you can use as the foundation of customized, user-defined classes.

Classes have attributes, which define the inherited and locally defined properties of the class. G2 maintains class attributes within **attribute tables**.

Classes can have associated methods, which define the operations characteristic of each class. Methods allow generic operations to be implemented in class-specific ways. Code that invokes a method needs only to know the method's name. The details of how to perform the operation exist in the method, not in the code that invokes it.

Integrity Bundle

The Integrity bundle is represented by the following KBs, which are included in your application when you create a new application with the New menu choice or when you merge in the *integrity.kb* module into your own module. The table shows KB modules that are included in the Integrity bundle:

KB File	Description
<i>gndo</i>	All core services
<i>gsnmp</i>	SNMP support
<i>gdx</i>	Domain map import utilities, including HPOV support (map importer)
<i>pingmgr</i>	Ping Manager support
<i>ompe-ui</i>	Message Parsing Engine support

KB File	Description
<i>ode-opac</i>	ODiE support
<i>gsockman</i>	Socket Manager support
<i>ipra</i>	IP Reachability Analysis support
<i>symcure</i>	SymCure support (Causal Directed Graphs)

All of these modules are included when you load *integrity.kb*.

Running Integrity

Describes how to start the server and connect the client.

Introduction 17

Starting the Server and Connecting the Client 18

Connecting to a Specific Server at Startup 19

Starting the Server with Your Application Loaded 21

Exiting Integrity 21



Introduction

Integrity is a client/server application. Integrity provides a batch file that, by default, starts the G2 server as a hidden process on the local machine at a default port (1111).

To run Integrity, you must connect the Telewindows client to the server. By default, Telewindows automatically connects to the server running on the local machine on the default port.

You can run Integrity in a secure G2 environment, which means users must provide a password before Integrity grants access to a KB. User names and passwords are stored in the *g2.ok* file. For details on how to configure Integrity to run in a secure G2 environment, see Chapter 54 "Licensing and Authorization" in the *G2 Reference Manual*.

Starting the Server and Connecting the Client

You can start the server and connect the client by using the Start menu.

To start the server and connect the client:

- 1 Choose Start > Programs > Gensym G2 2011 > G2 Integrity > Start G2 Integrity Server.

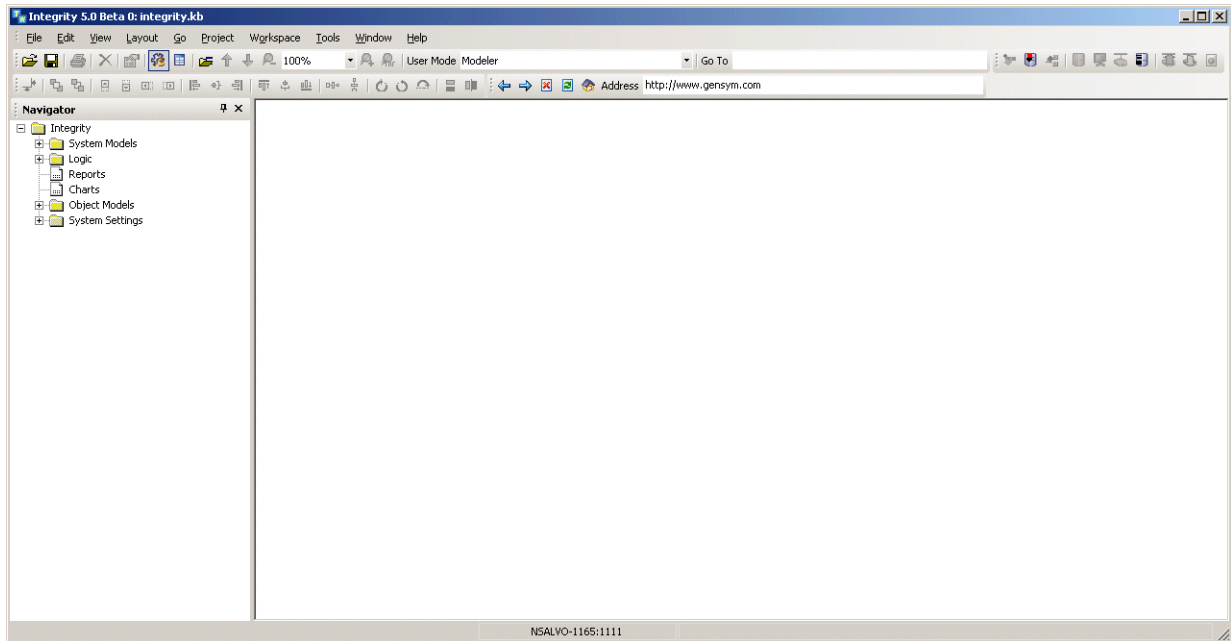
This menu choice starts the G2 server, using the *StartServer.bat* batch file, located in the *g2* directory of your Integrity installation directory. It starts the server on the local machine on TCP/IP port number 1111, and it automatically loads the KB named *integrity.kb*.

When the server has been started, the G2 icon appears in the system tray. When the server is running, the icon looks like this: 

- 2 Once the server is running, connect the client in one of two ways:
 - ➔ To connect Telewindows to the server running on the default host and port, choose Start > Programs > Gensym G2 2011 > Telewindows Next Generation.
 - or**
 - ➔ To connect Telewindows to the server running on the local host on the current port, right-click the G2 icon in the system tray and choose Connect Telewindows.

The Telewindows client is now connected to the G2 server.

When the client is connected and all files have finished loading, you will see this window:



Connecting to a Specific Server at Startup

You can run the Integrity client and server on different computers, or multiple Integrity servers on the same computer.

You can:

- [Connect the client directly to the server.](#)
- [Start the server on a specific port.](#)
- [Connect the client to a specific server.](#)

Connecting the Client to the Default Server

To connect the client to the default server:

- 1 Start the Integrity server from the Start menu.

By default, the server starts on the local host at port 1111. Each time you start a new server on the same machine, the port number increments by one. For example, if you start another server, the port number would be 1112.

- 2 To determine the host and port, hover the mouse over the G2 server icon in the system tray.

For example, *MY-HOST:1111* means the server is running on the machine named *MY-HOST* at port 1111.

- 3 Right-click the G2 server icon in the system tray and choose Connect Telewindows.

The Telewindows client connects to the specific host and port of that server.

Starting the Server on a Specific Port

To start the server on a specific port:

- 1 Right-click the Start G2 Integrity Server menu choice in the Start menu and choose Create Shortcut.
- 2 Rename the shortcut and/or drag it to your desktop, as needed.
- 3 Display the properties dialog for the shortcut and click the Shortcut tab.
- 4 Configure the Target property in the dialog to be the specific port on which to start the server, using the *-tcpport* command-line option.

For example, to start the server on port 1115, the shortcut would look like this:

```
"C:\Program Files\Gensym\g2-2011\g2\StartServer.bat"  
-kb ..\integrity\kbs\integrity.kb -nowindow -tcpport 1115
```

Connecting the Client to a Specific Server

To connect the client to a specific server:

- 1 Create a shortcut to the *twng.exe* file located in the *g2* directory of your Integrity product installation, either directly or by creating a shortcut from the Telewindows Next Generation menu choice in the Start menu.
- 2 Display the properties dialog for the shortcut and click the Shortcut tab.
- 3 Configure the Target by appending the host and port of the Integrity server to which to connect, using this syntax: *host:port*.

For example, to connect to *my-host* at port *1115*, the shortcut would look like this:

```
"C:\Program Files\Gensym\g2-2011\g2\twng.exe"  
my-host:1115
```

Starting the Server with Your Application Loaded

By default, the server starts up with the default Integrity application running, *integrity.kb*. Once you create an application, you might want to create a shortcut to the Integrity server that automatically loads your application at startup.

To start the server with your application loaded:

- 1 Copy the Start G2 Integrity Server shortcut from the Start menu.
You can rename the shortcut and drag it to your desktop, as needed.
- 2 Display the properties dialog for the shortcut and click the Shortcut tab.
- 3 Configure the application to load by editing the Target.

For example, to load the application named *opx_demo.kb* located in the *\integrity\examples* directory, the Target should look like this:

```
"C:\Program Files\Gensym\g2-2011\g2\StartServer.bat"
-kb "c:\Program Files\Gensym\g2-2011\integrity\examples\
opx_demo.kb" -nowindow
```

Exiting Integrity

To exit Integrity, you disconnect the client from the server, then shut down the server. By default, you can only exit the server directly from the client in Developer mode.

To disconnect the client from the server:

- ➔ Choose File > Close.

To exit the server:

- ➔ Right-click the G2 server icon in the system tray and choose Shut Down G2.

or

- 1 Choose Tools > User Mode > Developer.
- 2 Choose File > Exit.
- 3 Click Yes in the confirmation dialog.

Working with Models

Describes how to work with models through the menus and toolbars.

Introduction	24
Summary of Common Tasks	24
Using the Project Menu	25
Navigating Applications	27
Interacting with Workspaces	30
Using the Menus	34
Using the Integrity Toolboxes	44
Using the G2 Toolbox	46
Interacting with Objects	46
Using the Toolbars	49
Switching User Modes	53
Configuring User Preferences	54
	64



Introduction



To work with Integrity models, you perform these tasks:


- [Use the Project menu.](#)
- [Navigate models.](#)
- [Interact with workspaces.](#)
- [Use the menus.](#)
- [Use the Integrity toolbox.](#)
- [Use the G2 Toolbox.](#)
- [Interact with objects in the model.](#)
- [Use toolbars.](#)
- [Switch user modes.](#)
- [View messages.](#)

You can also view a [summary of command tasks.](#)

Summary of Common Tasks

This section summarizes how to perform common tasks in Integrity:

To...	Do this...
Display the popup menu for an object on a workspace	Click the right mouse button on the object.
Display the properties dialog for an object on a workspace	Double-click the object, select the object and press the F4 key, or choose Properties from the object's popup menu. You can also select the item, then choose Edit > Properties or click the equivalent toolbar button: 
Display the detail for an object	Choose Show Detail from the popup menu for the object, choose View > Show Details, enter Ctrl + right click on the object, or click the equivalent toolbar button: 
Display the Integrity toolbox	Choose View > Toolbox - Integrity.

To...	Do this...
Adjust the size of a workspace and its associated window to fit the contents of the workspace	Choose Shrink Wrap on the workspace, choose Layout > Shrink Wrap, or click the equivalent toolbar button: 
Hide a workspace	Click the Minimize button on the window, choose Hide on the workspace, choose View > Hide, or enter Ctrl + right click on the workspace.

Using the Project Menu

You create, configure, and interact with Integrity objects to create a model by using the Project menu.

You can also create and interact with objects through the Navigator, and you can search for objects once they exist. For more information, see:

- [Using the Navigator.](#)
- [Searching for Objects.](#)

Using the Project Menu

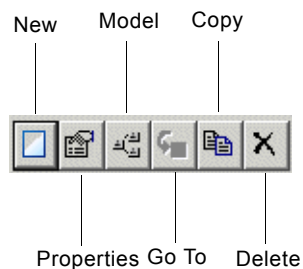
The Project menu allows you to create and manage the various objects you need to build an Integrity application.

For details, see [Using the Project Menu.](#)

Using the Manage Dialog

The Manage dialog allows you to create and configure new Integrity objects, show model details, copy and delete objects, and perform specific operations.

The Manage dialog provides these toolbar buttons:



The buttons in the Manage dialog are enabled or disabled, as appropriate, for the particular type of object.

The Go To button is disabled in Modeler mode because, typically, you interact with objects through properties dialogs and model details. You can go to objects directly through the Navigator or search, if desired.

For information about interacting with objects directly, see [Interacting with Objects in Developer Mode](#).

To use the Manage dialog:

- 1** Choose a submenu from the Project menu and choose Manage.
If the submenu has additional submenus, choose one of the submenus. The Manage dialog appears, which includes all objects in the submenu.
- 2** To create a new object, click the New button in the Manage dialog.
A properties dialog appears for configuring the object. The default name is a unique, system-generated name.
- 3** Configure the properties, depending on the type of object, and click OK.
For information on configuring the properties, see the various chapters in this guide.
The object now appears in the Manage dialog.
- 4** Select an object in the list to enable the toolbar buttons, as appropriate for the type of object.
- 5** To display the properties dialog for an object, click the Properties button.
Note that the only way to configure the properties of a container object once it has been created is through the Manage dialog.
- 6** To display the detail associated with a container object, click the Model button.
- 7** To copy an existing object, select the object you want to copy, then click the Copy button.
A properties dialog appears for configuring the copy. The default name is the existing object name with **-copy** appended.
- 8** To delete an object, select the object you want to delete and click the Delete button.

Using the Project Submenus

Integrity provides access to the various objects in a model through submenus of the Project menu. Selecting the menu choice for a configuration object displays the properties dialog for the object. Selecting the menu choice for a container object displays its detail.

To use the project submenus:

- 1 Choose a submenu from the Project menu.
If the submenu has additional submenus, choose a submenu until you see a submenu that includes the names of all objects of that type.
- 2 Choose an object from the submenu.

Navigating Applications

To navigate applications, you can:

- [Use the Navigator.](#)
- [Search for objects.](#)


For information on creating and managing objects through the Project menu, see [Using the Project Menu.](#)

Using the Navigator

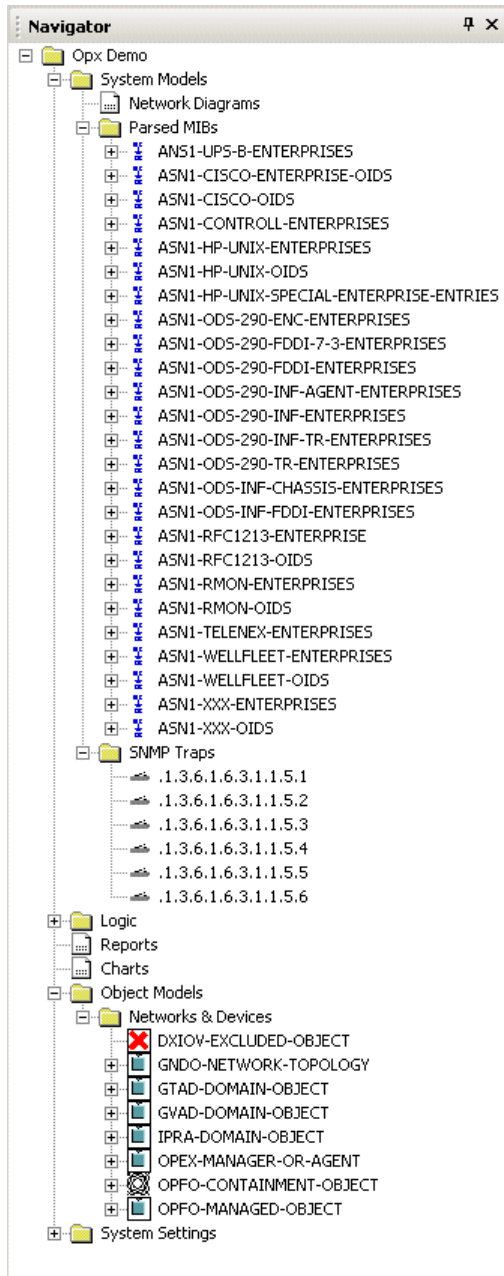
The Navigator displays all the elements of a project.

You can interact with objects in the Navigator, for example, showing its properties or going to the detail, depending on the type of object. You can also create new objects from the Navigator.

To display the Navigator:

- ➔ Choose View > Navigator or click the equivalent toolbar button () and expand the tree view to the desired level.

Here is the Navigator for the opx_demo application with the tree expanded:



To interact with objects in the Navigator:

➔ Right-click a node in the Navigator and choose the desired menu choice.

In addition to the menu choices that you normally get when you right-click the object, you can choose Go To to show the selected object. Depending on the type

of object, you might go to the object on a detail or you might go to the object in a repository.

You can also choose New Instance on the Network Diagrams folder to create a new domain mapnetwork diagram directly from the Navigator.

Searching for Objects

You can search for specific types of objects, by matching text in the label field and/or the target class, depending on the type of object. You can also go directly to named objects.

To search for objects:

- 1 Choose Tools > Search and choose a category of object to be found.
- 2 Enter the Keyword text to match and, depending on the type of object, optionally, the Target Class.
- 3 Configure Search By to search for the keyword only, class only, keyword or class, or keyword and class.
- 4 Click the Search button.

The search results include all objects whose label matches the specified text.

- 5 Select an object and click the Go To button.

An arrow appears next to the found object, if it exists; otherwise, the Search dialog display No Matches Found.

Depending on the type of object, you might go to the object on a detail or you might go to the object in a repository. You can interact with the object through its menu choices, for example, to go its detail or show its properties.

To go to a named object in the model:

- ➔ Enter the exact name of an object in the Go To type-in box on the toolbar:



A red arrow points to the named object on a workspace.

Interacting with Workspaces


You place all model objects on detail workspaces, which appear their own window. You display and interact with workspaces in these ways:

- [Display a detail workspace.](#)
- [Hide a workspace.](#)
- [Delete a workspace.](#)
- [Edit workspace properties.](#)
- [Scale a workspace.](#)
- [Shrink wrap a workspace](#) to fit the enclosed elements.
- [Show the superior object](#) for a workspace.
- [Print a workspace.](#)
- [Save a workspace as a JPEG file.](#)
- [Assign a background image to a workspace.](#)
- [Create and access top-level workspaces.](#)

Displaying a Detail Workspace

A number of objects define detail, which is a workspace associated with the object on which you place other objects.

To display detail for an object:

- ➔ Right-click the background of a workspace and choose Show Detail, choose View > Show Details, or click the equivalent toolbar button: ()

Hiding a Workspace


To hide a workspace:

- ➔ Right-click the background of a workspace and choose Hide or press Ctrl + right-click on the workspace.

Deleting a Workspace

Deleting a workspace permanently deletes it from the server, including all objects on the workspace.

To delete a workspace:


- ➔ Select a workspace and choose Edit > Delete, right-click the background of a workspace and choose Delete, or click the equivalent toolbar button: ()

Editing Workspace Properties

You can edit the name of the workspace, as well as the background and foreground colors, and the margins around the objects at the edges of the workspace. By default, the background color is white and the foreground color is black.

For information about configuring the background image, see [Loading Background Images](#).




To edit workspace properties:


- 1 Select a workspace and choose Edit > Properties, right-click the background of a workspace and choose Properties, or click the equivalent toolbar button: ()
 - 2 Configure the Names to be any text.
The text is converted to a symbol, with hyphens in place of spaces when you accept the dialog.
 - 3 Configure the Workspace Margin by entering the number of pixels.
 - 4 Configure the Foreground Color and Background Color by choosing a color.
- The name appears at the top of the workspace when you accept the dialog.

Scaling a Workspace

You can scale a workspace to fit the current window, or zoom a workspace in, out, or to a specific scale.

To scale a workspace:


- ➔ Choose View > Zoom In or Zoom Out, enter Ctrl + = to zoom in or Ctrl + - (minus) to zoom out, or click the equivalent toolbar buttons: ( )
- or
- ➔ Choose View > Zoom, then choose or enter a zoom scale, or enter a specific zoom scale in the zoom scale on the toolbar: (100% )
- or

→ Choose View > Zoom to Fit or click the equivalent toolbar button: ()

Shrink Wrapping a Workspace

When you move objects on a workspace beyond the visible borders, the borders adjust to fit the objects. When you move objects on a workspace such that the workspace contains extra space at its borders, you can adjust the borders by shrink wrapping the workspace. Shrink wrapping a workspace also adjusts the window size. You can resize the window to make it smaller to add scroll bars to the window.

To shrink wrap a workspace:

→ Select a workspace and choose Layout > Shrink Wrap or click the equivalent toolbar button: ()

This figure shows a workspace that has extra space around its borders:


This figure shows the result of shrink wrapping the workspace:

This figure shows the result of dragging the object on the workspace so it has extra space around its borders, then adjusting the window size to make it smaller, which adds scroll bars:

Showing the Superior Object of a Detail Workspace

You can show the superior object of a detail workspace.

To show the superior object of a detail workspace:


→ Right-click the background of a workspace and choose Go to Superior, or select a detail workspace and choose View > Go to Superior or click the equivalent toolbar button: ()

The workspace with the superior object is now on top with an indicator arrow next to the object.

Depending on the type of object, you might go to the object in a repository. You can interact with the object through its menu choices, for example, to show its properties.

Printing a Workspace

To print a workspace:

→ Choose File > Print, or enter Ctrl + P or click the equivalent toolbar button (), and configure the Print dialog.

Saving a Workspace to a JPEG File

To save a workspace to a JPEG file:

- ➔ Choose File > Save as JPEG and specify a file name.

Loading Background Images

You can load one or more JPEG, XMB, or GIF files as the background for a workspace.

To load a single background image:

- ➔ Choose Workspace > Load Background Image, navigate to the image to use as the background, and click Open.

To remove background images:

- ➔ Choose Workspace > Delete Background Image.

Creating and Accessing Top-Level Workspaces

Typically, you create new workspaces when you create network diagrams through the Project menu. However, you can also create new workspaces directly through the Workspace menu, which are top-level workspaces that you can access by name.


To create a new top-level workspace:

- 1 Choose Workspace > New.

The workspace is assigned a unique number, which starts with unnamed-workspace.

- 2 Configure the workspace properties as described in [Editing Workspace Properties](#).

To access the top-level workspace:

- 1 Choose Workspace > Get or click the equivalent toolbar button: ()

A list of all top-level workspaces available in the current user mode appears.

- 2 Select a workspace and click OK.

Using the Menus

The top-level menu bar consists of these menus:

Menu	Description
File	Standard file operations, and print and export operations for workspaces.
Edit	Standard editing operations for objects on workspaces.
View	Display the various toolboxes and toolbars, display the Navigator, zoom workspaces, show details, and show superior objects.
Layout	Standard layout operations for objects on workspaces, including align, distribute, rotate, reflect, order, nudge, as well as shrink wrapping workspaces.
Go	Standard browser navigation operations and interaction with the server.
Project	Manage system models, object models, reports, charts, system settings, and user preferences.
Workspace	Create new and get existing workspaces, and edit background images for workspaces.
Tools	Find model objects, show users, and switch user modes.
Window	Control window positioning and choose the active window.
Help	Display online help.

The following sections summarize each of these menus.

For information about how to use specific menu choices, see the referenced sections.

Using the File Menu

The File menu allows you to perform basic file and module operations.

Menu Choice	Description
New	Creates a new project. See Working with Projects .
Open	Opens an existing project, replacing the one currently loaded.
Save	Saves the top-level module of the current project.
Save As	Saves the top-level module of the current project to a user filename. You save models to filenames with a <i>.kb</i> extension.
Save as JPEG	Exports the currently selected workspace as a <i>.jpg</i> file.
Print	Prints the currently selected workspace to a postscript printer.
Close	Exits the client.

Using the Edit Menu

The Edit menu allows you to perform basic edit operations for objects.

Menu Choice	Description
Delete	Deletes the selected object.
Transfer	Transfers the selected object to the mouse. Click on a workspace to transfer the object.
Clone	Transfers the selected object to the mouse. Click on a workspace to clone the object.
Select All	Selects all objects on a workspace.
Properties	Displays the properties dialog for the selected object.
Colors	Changes the colors of the icon regions of the selected objects.

Using the View Menu

The View menu allows you to show and hide toolboxes and toolbars, and to control the zoom scale.

For details about each of the toolboxes, see [Using the Integrity Toolboxes](#).

The View menu contains the menu choices in the following table:

Menu Choice	Description
Toolbars > Standard	Toggles the Standard toolbar, which contains standard buttons for file and edit operations.
Toolbars > Layout	Toggles the Layout toolbar, which contains buttons for performing standard layout operations for objects on workspaces.
Toolbars > Web	Toggles the Web toolbar, which contains standard buttons for browsing HTML and text pages.
Toolbars > Integrity	Toggles the Integrity toolbar, which contains buttons that provide tools for Integrity users.
Status Bar	Toggles the status bar, which displays the connection status to the server.
Message Board	Displays the G2 Message Board, which displays text messages.
Message Browser	Displays a message browser of operator messages.
Navigator	Toggles the display of a tree view of all objects in the current project. See Navigating Applications .
Toolbox - Integrity	Toggles the display of the Integrity toolbox, which contains location and network containers, and network objects.
Toolbox - Integrity Export Import	Toggles the display of the Integrity toolbox, which contains tools for importing and exporting network diagrams.
Toolbox - SNMP Traps	Toggles the display of the SNMP Traps toolbox, which contains tools for SNMP trap processing.

Menu Choice	Description
Toolbox - Message Parsing Engine	Toggles the display of the Message Parsing Engine toolbox, which contains blocks for parsing message text.
Toolbox - OPAC	Toggles the display of the OPAC toolbox, which contains OPAC blocks.
Toolbox - ODiE Subscriber	Toggles the display of the ODiE Subscriber toolbox, which contains OpEx Dispatch Engine (ODiE) for handling events and responses to events.
Zoom	Scales the selected workspace.
Zoom In	
Zoom Out	
Zoom to Fit	
Hide	Hides the currently selected workspace.
Go to Superior	Displays the superior object of the currently selected workspace.
Show Details	Shows the detail workspace of the currently selected object.

Using the Layout Menu

The Layout menu allows you to interact with objects on workspaces. For details, see [Interacting with Objects](#).

Menu Choice	Description
Order > Bring to Front Send to Back	Controls the stacking order of selected objects on workspaces.
Nudge > Nudge Up Nudge Down Nudge Right Nudge Left	Micro-adjusts the position of selected objects in each direction.
Align or Distribute > Align Left Align Center Align Right Align Top Align Middle Align Bottom Distribute Horizontally Distribute Vertically	Aligns two or more selected objects along various axes. Distributes three or more selected objects vertically or horizontally.

Menu Choice	Description
Rotate or Flip > Normal 90 Clockwise 90 Counterclockwise 180 Flip Horizontally Flip Vertically	Rotates and reflects the selected objects.
Shrink Wrap	Adjusts the borders of the selected workspace to just fit the contained objects.

Using the Go Menu

The Go menu allows you to perform standard browser navigation and interact with the server.

Menu Choice	Description
Back	Provides standard browser operations for HTML and text pages.
Forward	
Stop	
Refresh	
Home	

Using the Project Menu

The Project menu allows you to interact with all the objects in the current project, as follows:

Menu Choice	Description
Initialize Application	Initializes all process maps in the application, which creates specific GEDP diagrams for each domain object with an associated generic diagram template, resets datapoint histories, compiles all SymCure diagrams, and clears all diagnoses from the various message browsers.
Uninitialize Application	
My User Preferences	Configures user preferences for the current user. See Configuring User Preferences .
Logic >	Creates and manages Integrity logic models that diagnose abnormal conditions. For information on the menu choice in the Diagnose menu, see Creating Generic Fault Models and the <i>SymCure User's Guide</i> .
Diagnose >	
Fault Models	
Diagnosis Managers	
Diagnostic Console	
Debug Specific Fault Models	
Import	
Enable Tuning	
Operator Actions	
Text Parsing	
Reports	Creates and manages a variety of reports.

Menu Choice	Description
Charts	Creates and manages various types of charts.
Object Models > Networks & Devices	Creates and manages networks and devices.
System Settings	Creates and manages the various system settings described below.
System Settings > Interfaces > SQL SMTP JMS SNMP HTTP Socket Manager	Creates and manages network and database interface objects for communicating with various types of external systems.
System Settings > Interface Pools > SQL SMTP JMS	Creates and manages network and database interface pools for communicating with various types of external systems.
System Settings > Message Browsers > Queues Events Messages Access Tables Templates	Creates and manages custom message browsers and queues.

Menu Choice	Description
System Settings > Users	Creates and manages user preferences. See Configuring User Preferences .
System Settings > System Performance	Enables, disables, and configures system performance metrics.
System Settings > Event and Alarm Metrics	Enables and disables event and alarm metrics.

Using the Workspace Menu

The Workspace menu allows you to interact with workspaces. For details, see [Interacting with Workspaces](#).

Menu Choice	Description
New	Creates a new workspace.
Get	Displays a list of named workspaces, which you can display.
Load Background Image Delete Background Image	Loads and deletes background images for the selected workspace.

Using the Tools Menu

The Tools menu allows you to browse objects in the model.

Menu Choice	Description
Search	Allows you to search for objects in a model by name or label. See Searching for Objects .

Menu Choice	Description
Show Users	Shows the users currently logged into the server.
User Mode >	<p>Changes the user mode. The default user mode is Modeler, which allows you to create models by copying, connecting, and configuring objects, and to run simulations. Operator mode allows end users to view models only. Developer mode allows developers to customize the application.</p> <p>Note: In general, you work in Modeler mode. Very occasionally, modelers need to switch to Developer, Administrator, or System Administrator mode to perform particular tasks.</p> <p>See Switching User Modes.</p>
Administrator	
System-Administrator	
Developer	
Modeler	
Operator	

Using the Help Menu

The Help menu allows you to access online help that displays as a window within the client:

Menu Choice	Description
G2 Help Topics	Display the G2 online help.
Integrity Help Topics	Displays the Integrity online help.
Server Information	Displays version information about the server.
About G2	Displays the G2 title block, which shows the current version.
About Integrity	Displays the Integrity title block, which shows the current version.

You can view PDF versions of the following guides:

- *Integrity User's Guide*
- *Integrity Utilities Guide*

To view the online manuals:

- ➔ Choose Start > Programs > Gensym G2 2011 > Documentation > G2 Integrity and choose the manual you want to view.

Using the Integrity Toolboxes

The Integrity toolboxes contain all of the objects that you use to create a model.

Integrity provides the following toolboxes:

- Toolbox - Integrity
See [Getting Started](#).
- Toolbox - Integrity Export Import
See Chapter 17 “Domain Map Export/Import (DXI3)” and Chapter 18 “Open View Map Importer (OVMAP)” in the *Integrity Utililies Guide*.
- Toolbox - SNMP Traps
See Part IV “SNMP-Bridges” in the *Integrity Utililies Guide*.
- Toolbox - Message Parsing Engine
See Chapter 14 “Message Parsing Engine (MPE)” in the *Integrity Utililies Guide*.
- Toolbox - OPAC
See Part I “OPAC Blocks Reference” in the *Integrity Utililies Guide*.
- Toolbox - ODiE Subscriber
See Chapter 13 “OpEx Dispatch Engine Reference (ODiE)” in the *Integrity Utililies Guide*.

To display and interact with the Integrity toolboxes:

- 1 Choose a toolbox from the View menu.


The toolbox appears with the first palette in the toolbox visible. The palettes are organized alphabetically. You access the various palettes in the toolbox by clicking the buttons at the bottom of the toolbox.

Here is the Location and Network Containers palette of the Integrity toolbox:



- 2 To access the various palettes in the toolbox, hover the mouse over a button to display its tool tip, then click the button to display the palette.

Depending on the size of toolbox, the toolbar at the bottom shows only a subset of the available buttons in the toolbox.

- 3 To display the additional buttons in the toolbox, click the configure button at the far right of the toolbar (), then choose a palette.
- 4 To configure the buttons that are visible in the toolbar and associated configuration menu, choose Add or Remove Buttons to display a list of all palettes, then choose a button to add or remove.

Once you have configured the buttons you want, you can expand the buttons to show their labels for some or all of the buttons.

- 5 To show button labels in the toolbox, drag the divider at the bottom of the toolbox up to expose the buttons with their labels.

Once you have configured the buttons you want to appear in the toolbox, you can auto hide the toolbox by clicking the pin in the upper right corner of the toolbox.

Note Do not close the toolbox or the toolbox reverts to the default set of buttons.

- 6 Click the pin to autohide the toolbox, and move the mouse over the tab to display the toolbox after it has been hidden.

You can display, configure, and autohide multiple toolboxes, as needed, each of which will have its own toolbox tab.

Using the G2 Toolbox

In general, you use the G2 toolbox when customizing models.

For details, see the *G2 Reference Manual*.

Interacting with Objects

You can interact with objects in a network diagram by using the Edit menu, the object's popup menu, and the Layout menu. Many of the menu choices have shortcuts and/or equivalent toolbar buttons.

When you create a process mapnetwork diagram, we recommend that first, you place the domain objectsnetwork objects on the workspace, then you align and distribute them, using buttons on the Layout toolbar, then you connect them, as needed.

You configure attributes of objects through properties dialogs.

Selecting Objects

To select one or more objects:

- Click an object to select it.

or

- Click and drag a rectangular area to select all the objects in the rectangle.

or

- Use Shift key and click on an object to add or remove it to or from an existing selection.

or

- Use the Alt key and click on a connected network of objects to select all the connected objects.

To select all objects on a workspace:

- Choose the Edit > Select All or enter Ctrl + A.

Cutting, Copying, Pasting, and Deleting Objects

When you copy an object, the new object has the same property values as the existing object. If the object has details, the new object has the same details. You can transfer objects from one workspace to another.


To copy and paste objects:

- Select one or more objects to copy, choose Edit > Clone, then click on any workspace to paste the selected objects to the workspace.



To cut and paste objects:

- Select one or more objects to cut, choose Edit > Transfer, then click on any workspace to paste the selected objects to the new workspace.

To delete objects:

- Select an object, then choose Delete from the Edit menu or from the popup menu, press the Delete key, or click the equivalent toolbar button (), then click Yes to confirm the deletion.







Controlling the Layout of Objects**To adjust the order of objects:**

- Select an object, then choose Layout > Order > Bring to Front or Send to Back or click the equivalent toolbar button: ( )



To rotate or flip objects:

- Select an object, choose Layout > Rotate or Flip, then choose the desired action from the submenu or click the equivalent toolbar button:      


To align objects:

- Select two or more objects, choose Layout > Align or Distribute, then choose the desired align action from the submenu or click the equivalent toolbar button: (     )

To distribute objects:

- Select three or more objects, choose Layout > Align or Distribute, then choose the desired distribute action from the submenu or click the equivalent toolbar button: ( )


To nudge an object up, down, right, or left:

- ➔ Select an object, choose Layout > Nudge, then choose the desired nudge action from the submenu; or hold down the Ctrl key while pressing the up, down, right, and left arrow keys to nudge the item in the desired direction; or click the equivalent toolbar button: 

For information on the Shrink Wrap toolbar button on the Layout toolbar, see [Shrink Wrapping a Workspace](#).

Displaying the Properties Dialog for an Object

To display the properties dialog for an object:

- ➔ Double-click the object.
or
- ➔ Select the object and press the F4 key.
or
- ➔ Choose Properties from the object's popup menu.
or
- ➔ Select the object, then choose Edit > Properties or click equivalent toolbar button: ()

Resizing an Object

You might need to resize an object.

To resize an object:

- ➔ Click an object to select it, and drag the selection handles to resize the object.

Editing Icon Color Regions

You can edit the color of any named region of any icon.

To edit icon colors:

- 1 Click an object to select it, and choose Edit > Colors.
- 2 Configure the color of the named icon region for the object, as desired.

Using the Toolbars

Integrity provides a number of toolbars that you can use to interact with models.

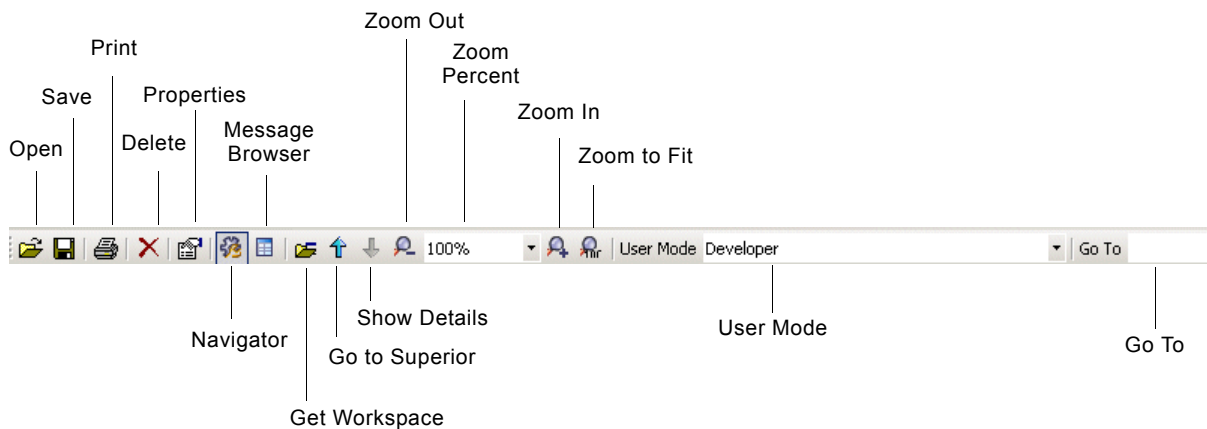
The toolbars are all docked, by default. You can drag the toolbar to a new location or off the toolbar to make it a floating toolbar.

The available toolbars are:

- [Standard toolbar](#)
- [Web toolbar](#)
- [Layout toolbar](#)
- [Integrity toolbar](#)
- [Status bar](#)

Standard Toolbar

The Standard toolbar contains many of the toolbar buttons that you need to work with the model:



To hide and show the Standard toolbar:

➔ Choose View > Toolbars > Standard.

For information on this button...

See...

Open [Opening a Project.](#)

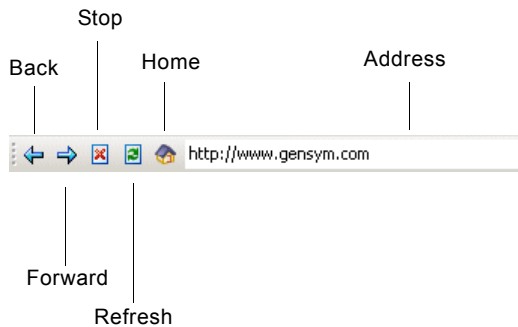
Save [Saving a Project.](#)

Print [Printing a Workspace.](#)

For information on this button...	See...
Delete	Cutting, Copying, Pasting, and Deleting Objects.
Properties	Displaying the Properties Dialog for an Object.
Navigator	Using the Navigator.
Get Workspace	Creating and Accessing Top-Level Workspaces.
Go to Superior	Showing the Superior Object of a Detail Workspace.
Show Detail	Displaying a Detail Workspace.
Zoom In, Zoom Out, Zoom Percent, and Zoom to Fit	Scaling a Workspace.
User Mode	Switching User Modes.
Go To	Searching for Objects.

Web Toolbar

The Web toolbar provides the standard browser navigation buttons and commands for browsing HTML pages:



To hide and show the Web toolbar:

➔ Choose View > Toolbar > Web.

You can go to any URL, including any HTML file on the World Wide Web or on the file system, or any RTF file.

To go to an HTML file on the World Wide Web, you use the standard HTTP protocol, for example, *http://www.gensym.com*.

To go to an HTML or RTF file on the file system, you use this protocol:

```
file:\<drive>:\<directory>\<filename>
```

For example, to go to the readme file, you would use:

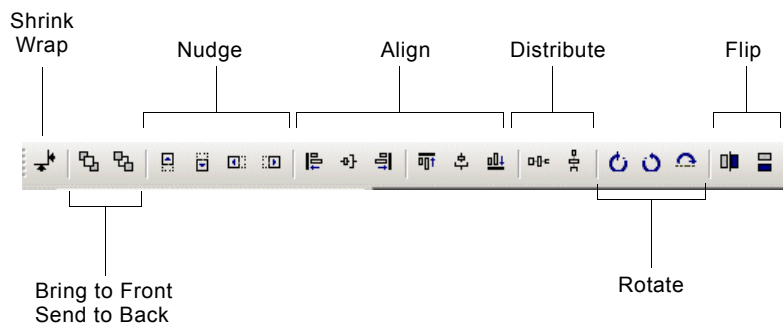
```
file:\C:\Program Files\Gensym\g2-2011\doc\integrity\integrity-readme.html
```

You navigate by using standard buttons in the Web toolbar or in the Go menu.

You configure the Home button URL in your user preferences. For more information, see [Configuring User Preferences](#).

Layout Toolbar

The Layout toolbar contains toolbar buttons that you need to control the visual layout of objects on a workspace:



To hide and show the Layout toolbar:

➔ Choose View > Toolbars > Layout.

For information on this button...

See...

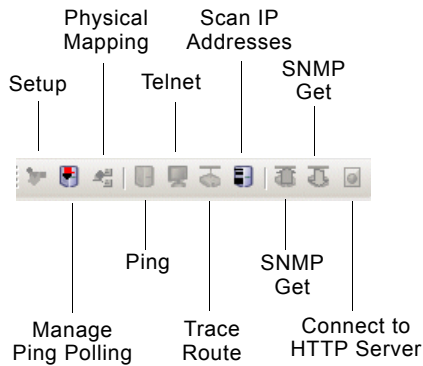
Shrink Wrap

[Shrink Wrapping a Workspace.](#)

Send to Front, Send to Back, Nudge, Align, Distribute, Rotate, Flip

[Controlling the Layout of Objects.](#)

Integrity Toolbar



To hide and show the Integrity toolbar:

➔ Choose View > Toolbars > Integrity.

Status Bar

The status bar shows various status information, such as the host and port of the client, the current file being loaded, and the progress bar.

By default, the status bar also shows the current message in the operator Message Browser. For information on how to disable this feature, see [Configuring User Preferences](#).

To hide and show the status bar:

➔ Choose View > Status Bar.

Switching User Modes

You build and run applications in one of four built-in **user modes**, or you can define your own user mode. The user mode determines what you can and cannot do when you create your application and run it. For example, the user mode determines whether you can move, edit, and delete objects, and whether you can use the full set of G2 features in your model. For example, the user mode determines the parameters that you can configure.

Integrity supports the following user modes for these classes of users:

This type of user...	Works in this user mode...	Which allows you to...
and end users	Operator	View pre-built applications without damaging them in any way. Operators cannot open, save, run, or configure applications.
who create applications	Modeler	This is the default user mode.
Integrity experts and G2 programmers	System-Administrator Administrator	Customizes the behavior of Integrity.

End users of fully developed applications generally work in Operator mode. Operator mode is restricted so that users may run a model but may not create, configure, or delete objects.

As a model developer, you will almost always be working in Modeler mode. This manual assumes you are working in Modeler mode, unless otherwise stated. Occasionally, as a model developer, you will also need to go into Developer mode to perform certain tasks.

If you are an expert who is customizing Integrity, you will be working mostly in Developer mode.

The user mode that is available to you depends on your login privileges.

To switch to a different user mode:

➔ Choose Tools > User Mode or configure the User Mode on the toolbar.

Configuring User Preferences

Integrity allows you to configure different levels of access and default behavior for different categories of users. When a particular user starts Integrity, the user preference associated with that user restricts the access and provides default behavior, as appropriate for the given user.

You can configure the following preferences:

- The default user mode, which determines the level of access to Integrity features.
- Subscription to queues.
- Message filter to subscribed queues, for filtering messages based on priority, process map, type, category, target, assigned to, age, and acknowledgement status.
- Acknowledgement and deletion permission and behavior in the Message Browser.
- Client disconnection, server shutdown, and modeling configuration permissions, and whether the user is an administrator.
- The default behavior for interacting with objects through menus and showing the logbook.
- Email and mobile email addresses for use with the JMail interface.

Specifying User Preferences for Different Types of Users

Integrity creates a default user preference for the server to determine the level of access and default behavior for all users that log into the server. Similarly, Integrity creates one user preference for each user associated with a G2 login account. The name of the user preference corresponds with the user name specified in the *g2.ok* file. For more information, see Chapter 62 “Licensing and Authorization” in the *G2 Reference Manual*.

If you are logged in as the user named **administrator**, you are automatically configured to be the Administrative User and can create and configure user preferences for all users. If you are logged in as any other user, you can only configure your own user preferences. You can be logged in either to your windowing system or to the Integrity server through a secure G2 as **administrator**.

We recommend that the user preference for the server provide access to all available features, and that it use either Modeler or Developer mode. The user preferences for the clients should provide appropriate levels of access and should use the appropriate user mode, depending on the type of user. For example, you might configure user preferences as follows for these types of users:

For this type of user...	Use this default user mode...	And provide these permissions and defaults...
Operators, who interact with messages only	operator	<ul style="list-style-type: none"> • Disconnect permission • Acknowledge message permission • Show message in operator mode by default • Subscribe to appropriate queues, depending on the model
Modelers, who create network diagrams	modeler	<ul style="list-style-type: none"> • Disconnect permission • Configuration permission • Acknowledge message permission • Delete message permission • Subscribe to Messages queue
Developers, who use G2 to customize models	developer	<ul style="list-style-type: none"> • Indicate items upon menu selection • Disconnect permission • Shutdown permission • G2 Logbook • Acknowledge message permission • Delete message permission • Subscribe to all queues

For this type of user...	Use this default user mode...	And provide these permissions and defaults...
Administrators, who configure user preferences for all users, using the Integrity user interface	system-administrator	The same as developers, plus Administrative User.
Administrators, who configure user preferences for all users, using G2's user interface	administrator	Note: You must log in as administrator to enable the Administrative User option.

Configuring User Preferences

In Modeler mode, you can configure these attributes for each user preference. For information about additional attributes that you can configure in administrator mode, see .

Attribute	Description
General	
User Name	The user name of the user that starts either the server or the client, which is read-only. If you are an administrative user, you can create new user preferences for specific users. For details, see .
Default User Mode	The default user mode for the specified user, which is <i>modeler</i> , by default. The options are: <i>operator</i> , <i>modeler</i> , <i>developer</i> , <i>system-administrator</i> , and <i>administrator</i> .
User Interface Theme	The Windows user interface theme. The default value is <i>window-theme-2003</i> .
Email Address Mobile Email	E-mail and mobile e-mail address of the specified user for sending email when a message occurs. For more information, see Delivering Messages by Email .

Attribute	Description
Home Process Map	A process map to use as the background in the operator interface. The default process map is default view , which is associated with the process map named guif-default-main-view . Click Select to display a list of all process maps in the KB and choose a map to use as the default background.
Telnet Command	The command for launching a Telnet session.
Default Web Location	The default URL when clicking the Home button in the Web toolbar.
Set Default User Mode	Whether the default user mode should be set upon startup.
Indicate Items	<p>Configures the behavior when choosing items from the Project menu. By default, Integrity displays the properties dialog or the model detail, depending on the type of item.</p> <p>Developers who are familiar with G2 and prefer to work with the iconic representations of items might want to enable the Indicate Items option, in which case, choosing items from the Project menu goes directly to the item.</p>
Extended Menus	Whether to display the complete list of objects in the Project submenus, the default. If your project has many domain models, for example, you might want to disable this option, in which case, selecting Project > System Models > Network Diagrams displays the Manage dialog for interacting with object.
Show Logbook	Whether to show the G2 Logbook when errors occur. By default, the G2 Logbook does not appear. Modelers or developers who are familiar with G2 might want to enable the Show Logbook option. We recommend that you disable this option for operators and modelers who are not familiar with G2.
Tabbed Mdi Mode	Whether to display workspaces in tabs in the window.

Attribute	Description
Restore Last Pane Settings	Whether to restore the settings for panes upon connection.
Message Browser	
Email Notification Mobile Email Notification	The format when sending e-mail and mobile e-mail messages. By default, the value is never , which means email messages are not sent. For details, see Delivering Messages by Email .
Modeler Browser	The browser to use in Modeler mode. The default is gevm-modeler-message-view-template , which is the browser that appears when you choose View > Message Browser.
Operator Browser	The browser to use in Operator mode. The default is gevm-operator-message-view-template , which is the browser that appears when you are in Operator mode.
Acknowledge Messages Upon Selection	Whether to acknowledge messages automatically when the operator selects a message in the Message Browser view of the operator interface. By default, messages are not automatically acknowledged. When Ack Msg Upon Selection is enabled, Ack Msg Permission must also be enabled.
Show Browser in Operator Mode	Whether to show the Message Browser by default view in the operator interface, or whether to show the process map view. By default, the Message Browser appears as the default view in the operator interface.
Enable Status Bar Message Browser	Whether to show the most recent message in the status bar.
Beep Enabled	Whether to enable beeping when new messages arrive in the Message Browser, as well as when they are acknowledged and deleted. By default, beeping is enabled.

To configure user preferences for yourself:

- ➔ Choose Project > My User Preferences and configure the user preferences, as needed.

For example, here is the default user preferences dialog appears for the user named nrs:

To configure user preferences for other users:

- ➔ Choose Project > System Settings > Users and choose the user whose preferences you want to configure.

Delivering Messages by Email

You can configure the user preference for individual users to provide an email address and a mobile email address, then configure rules for when to send email messages when an event occurs.

You can configure Integrity to format the message as short plain text, suitable for cell phones, for example, plain text with full message contents, or as an HTML document. You can also configure when to send a message, based on when it was created or updated, whether the user is currently connected to the server, and the priority of the message.

To deliver messages by email, you:

- [Start the G2 JMail Bridge process.](#)
- [Create, configure, and connect a JMail Interface object.](#)
- [Configure Integrity to send email messages.](#)
- [Configure startup parameter for sending email.](#)

Starting the G2 JMail Bridge Process

To deliver messages by email, you must start the G2 JMail Bridge process. You identify the host and port to which the bridge is connected for configuring in the JMail Interface object.

To start the G2 JMail Bridge process:

➔ Choose Start > Programs > Gensym G2 2011 > Bridges > G2 JMail Bridge.

The G2 JMail Bridge process appears in the command window.

To determine the bridge port:

➔ Open the command window for the bridge process.

The last line indicates the TPC/IP host and port number, for example:

```
TCP_IP:NSALVO-1165:22080
```

Creating, Configuring, and Connecting the JMail Interface Object

To deliver messages by email, you must create and configure a JMail Interface object, which specifies:

- A name.
- The host and port of the machine running the G2 JMail Bridge.
- Information about the SMTP mail server, including the user name, password, incoming and outgoing SMTP mail host, and SMTP protocol.

If the bridge process is running on the local machine, the host is localhost. The default port number is 22080, 22081, 22082, etc., depending on the number of clients that are currently connected on that port.

Note To configure a JMail Interface object, you must be in Developer mode.

Once you have configured the JMail interface object, you can connect it to the G2 JMail bridge process.

To create, configure, and connect a JMail Interface object:

- 1 Choose Tools > User Mode > Developer.
- 2 Choose Project > System Settings > Interfaces > SMTP > Manage and click the New button to create a new JMail Interface object.

Alternatively, you can choose View > Toolbox - G2, click the Network Interfaces tab, and create a JMail Interface object.

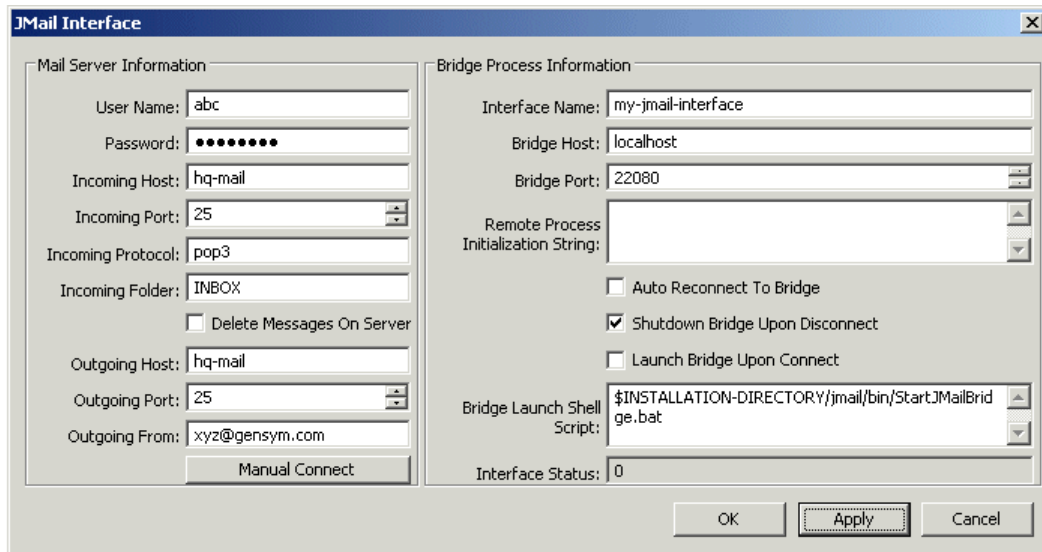
- 3 In the properties dialog for the JMail Interface object, configure the Interface Name attribute to be any symbol, for example, my-jmail-interface.
- 4 Configure the Bridge Host and Bridge Port to be the host and port of the machine on which you started the G2 JMail Bridge process.
- 5 Configure the following additional information:

Attribute	Description
User Name	The user name of the account to which email should be sent.
Password	The password of the user account to which email should be sent.
Incoming Host	The name of the host computer used for incoming email.
Incoming Port	The port number of the host computer used for incoming mail.
Incoming Protocol	The SMTP protocol that the incoming mail host uses. The default is pop3 .
Incoming Folder	The folder name of the user account to which to send email. The default is inbox .
Delete Messages on Server	Whether to delete the email message on the mail server after it is sent. By default, messages are not deleted.
Outgoing Host	The name of the host computer used for outgoing email.
Outgoing Port	The port number of the host computer used for outgoing mail.

Attribute	Description
Outgoing From	The name to use as the From address when the email message is sent, which cannot contain spaces.
Auto Reconnect to Bridge	Whether to automatically reconnect if the connection goes down.
Shutdown Bridge Upon Disconnect	Whether to shutdown the bridge when the connection is closed.
Launch Bridge Upon Connect	Whether to launch the bridge when a connection is made.
Bridge Launch Shell Script	Pathname to script for launching the bridge.

- 6 Click Apply to apply these values.
- 7 Click the Connect button in the dialog to connect the interface to the bridge.
- 8 Choose Tools > User Mode > Modeler to return to Modeler mode.

For example:



Configuring Integrity to Send Email Messages

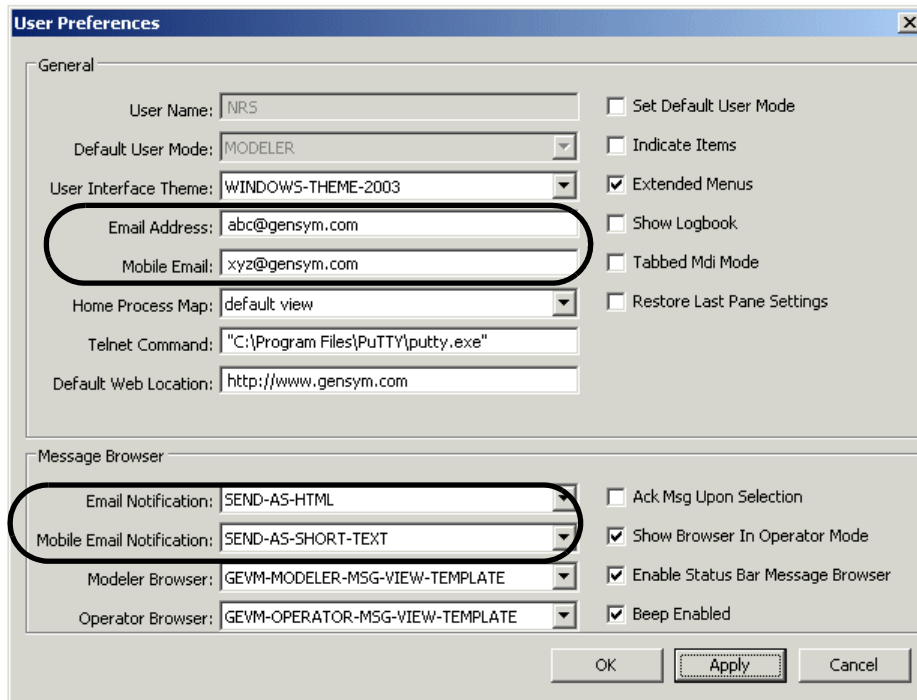
You configure Integrity to send email messages through the user preferences dialog.

To configure Integrity to send email messages:

- 1 Choose Project > My User Preferences.
- 2 Configure Email Address and/or Mobile Email.
- 3 Choose the rule to use for each of the configured email addresses, as follows:
 - **never** – Do not send e-mail messages. This is the default rule.
 - **send-as-text** – Send the message text and details as plain text.
 - **send-as-short-text** – Send the message text only as plain text.
 - **send-as-html** – Send the message text and details as HTML.
 - **only-high-priority-as-text** – Send the message text and details as plain text only if the priority is 1.
 - **only-high-priority-as-short-text** – Send the message text as plain text only if the priority is 1.
 - **only-high-priority-as-html** – Send the message text and details as HTML only if the priority is 1.
 - **if-not-connected-send-short-text** – Send the message text as plain text only if the user is not connected to the server.
 - **if-not-connected-send-as-text** – Send the message text and details as plain text only if the user is not connected to the server.
 - **if-not-connected-send-as-html** – Send the message text and details as HTML only if the user is not connected to the server.

When a message occurs, Integrity also sends an email to the specified addresses.

Here is the User Preferences dialog with both email addresses and rules configured:



Configuring Startup Parameter for Sending Email Messages

You can configure the following startup parameter in the configuration file:

`JMAIL-INTERFACE-NAME=none`

Specifies the default JMail interface to use for sending email messages.

For details about using the configuration file, see the *G2 Run-Time Library User's Guide*.

Customizing the Application

Describes how to create an operator menu and how to use initializations to customize your application.

Introduction **65**

Constructing an Operator Menu **65**

Defining Initializations **78**

Setting Preferences **81**

Creating New Palettes **87**

Customizing the User Interface Using Cyberformer **91**



Introduction

This chapter describes techniques you can use to customize the appearance of your application and to modify the user interface. You will learn how to create user-selectable menus and how to change the behavior and appearance of the system, using Initializations.

Constructing an Operator Menu

In your application, you might want to design a menu as a part of the operator interface. The G2 Menu System (GMS) provides a powerful set of tools that allow you to construct menus. Using GMS, you build a menu resource and then compile the menu resource to create the menu. A menu resource is built by cloning templates off the GMS palette and connecting them together. GMS allows you to

build multiple menus in a single application and to switch between the menus. Therefore, you can create a customized operator menu and still use the Integrity developers menu.

To create a menu resource you must:

- Select a workspace for the menu resource.
- Create and edit the initialization objects.
- Create the items contained in the menu.
- Link together the menu items.
- Compile the menu.

In this chapter we will present a tutorial that creates a simple menu using the *doc_demo.kb* application. To build complex menus, refer to the *G2 Menu System User's Guide* which is provided with the G2 documentation set.

The example in this chapter uses the operator menu implemented in the *doc_demo.kb* sample application.

To view the doc_demo operator menu:

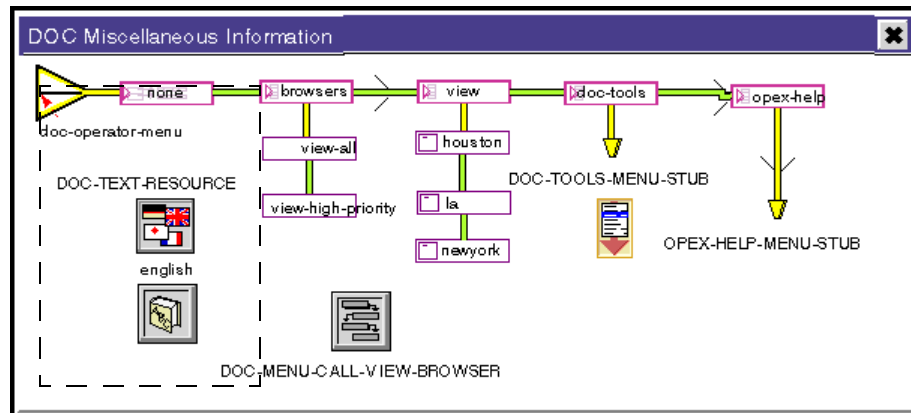
- 1 Load *doc_demo.kb*.
- 2 Click on the Gensym logo to the left of the File entry on the menu.
- 3 Choose Doc Demo Operator Menu from the list of available menus.

Select some of the options to see this menu. This menu provides an example of an operator interface.

To view the components of the doc_demo operator menu:

- 1 Using the Workspace Finder, locate and view the Docdemo-top-level workspace.
- 2 Select Miscellaneous Information from the top level workspace to display the workspace where the operator menu is defined.

The figure below shows the operator menu defined in `doc_demo`:



To create your own application operator menu, you must construct a menu resource similar to this one.

Selecting a Menu Template Workspace

To build a menu you must define a menu resource. In your application you should choose a workspace that can be accessed from the top-level workspace. In `doc_demo`, we used the Miscellaneous Information workspace. To keep this example simple, we will simply create a new workspace from the File menu.

To create a menu resource workspace:

- 1 Choose File > New > Workspace.
- 2 Select the background of the new workspace, select Name from the KB Workspace menu, then enter *menu-resource* to give the workspace a name.

Enabling List and Array Editing

Before you begin to create menu items, you should turn on the G2 facility that lets you edit lists and arrays in a spreadsheet format.

To enable list and array editing:

- 1 Using the Workspace finder, locate the GXL Spreadsheet workspace.
- 2 Click the check box Array and List Editing.
- 3 Select the background of the GXL Spreadsheet workspace and select Hide Workspace.

Note When the system is restarted, Array and List Editing is turned off. If you want to re-edit a menu resource after a restart, you must complete these steps again to turn on Array and List Editing.

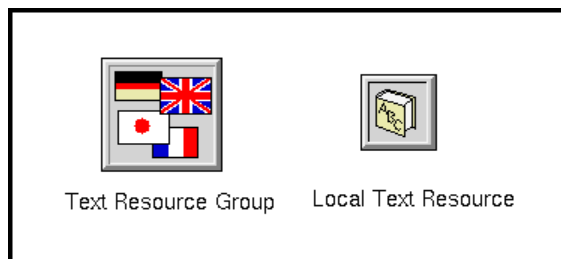
Creating a Text Resource and a Local Text Resource

Each menu resource must define a Text Resource Group and a Local Text Resource. These are part of the Gensym Foundation Resources (GFR). They provide multi-language capabilities for the menuing system.

To create a Text Resource Group and a Local Text Resource items:

- 1 Using the Workspace finder, locate the gfr-top-level workspace and display the Gensym Foundation Resources workspace.

The items you clone from this workspace are shown in the figure below:



Select Text Resource Group, move it to your menu workspace, then click to drop it on your menu resource workspace.

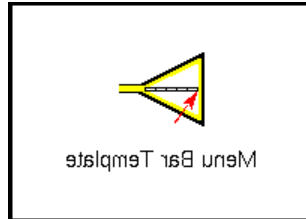
- 2 Select Local Text Resource, move it to your menu template workspace, then click to drop it on your menu resource workspace.

If your application supports multiple languages, you create a Local Text Resource for each language. In this example we will create only an English language resource.

- 3 Click on the background of the Gensym Foundation Resources workspace and select Hide Workspace.
- 4 Select the Text Resource Group item, select **name**, then type in the name *new-text-resource*.
- 5 Select the Local Text Resource item, select **table**, then enter the name of the Text Resource Group, *new-text-resource*, for the attribute **gfr-resource-group**.

Creating a Menu Bar Template

The menu bar template defines the name of the menu that appears on the list of menus under the Gensym logo. A menu bar template is shown in the figure below:



To create a menu bar template:

- 1 Using the Workspace finder, locate the gms-top-level workspace.
- 2 Select the Menu Bar Template from the Gensym Menu System workspace.
- 3 Move the cursor your menu resource workspace then click to drop the Menu Bar Template.
- 4 Select the menu bar template, select **table**, enter *new-operator-menu* as the value for the gms-label.

Configuring a Menu Template Item

After you create any type of menu template item, you must configure the item by:

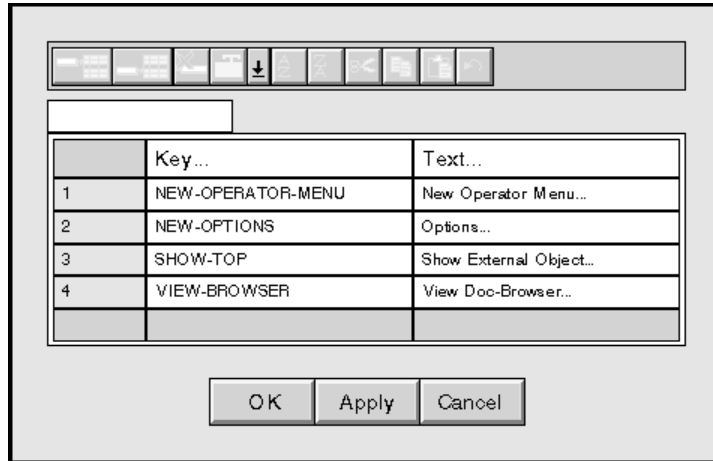
- Defining the gms-text-resource-group.
- Assigning a gms-user-key to the item.
- Defining a text string linked to the gms-user-key in each local text resource defined for the menu.

The procedure shown below shows how to configure the menu bar template item defined in this example. You can use the same procedure to configure any menu template item.

To configure the menu bar template:

- 1 Select the menu bar template then select **table** from the menu.
Enter the following values in the attribute table:
 - a *new-operator-menu* for the attribute gms-user-key.
 - b *new-text-resource*, for the attribute gms-text-resource-group.
- 2 Select the Local Text Resource and select **edit resource**.

If this selection does not appear, you need to turn on array editing as described in Enabling List and Array Editing on page 67. The local text resource text editor is shown in the figure below: The values shown are all defined in as part of this example.



Enter the value of the `gms-user-key`, `new-operator-menu`, in the column labeled Key..., `New Operator Menu` in the column labeled Text..., click OK, then click Yes.

Note Be sure to enter a carriage return after entering a value in a local text resource value to update the spreadsheet cell.

To add a new row in the Local Text Resource spreadsheet:

- 1 Click on the first column of an existing row.
- 2 Click one of the first two buttons on the left above the cells to insert a row above or below the current selection.

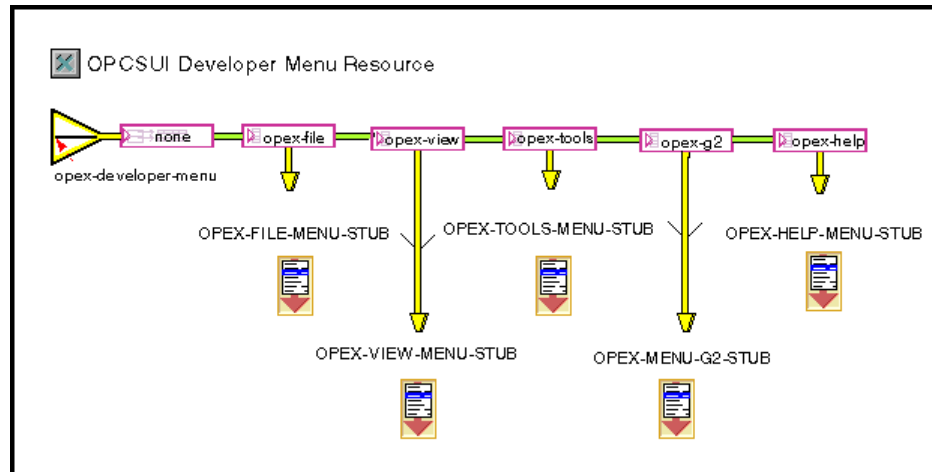
Adding the Menu Selection Icon

Click on the Gensym logo on the main menu to switch between the available menus in an application. This icon is created with a menu selection icon. You can clone this icon from the menu resource that defines the Integrity main menu.

To create the menu selection icon:

- 1 Choose Tools > Inspect.
- 2 Type in `go to opcsui-developer-menu`, click End, then click Hide on the workspace that displays the search results.

The figure below shows the Integrity main menu resource:



The menu selection icon is the rectangle on the top line, which contains the label `none`.

- 3 Select the menu selection icon from the OPSCUI Developer Menu Resource workspace, select `clone`, move the item to your menu resource workspace, then click to drop the item.

Tip To select menu template items, click on the left-hand border of the item. If you click on other areas, you may display the label edit box or table.

This item does not need to be reconfigured.

Cloning a Menu Group

The sample menu we are building will use part of the File menu defined on the Integrity main menu. You can clone any part of the Integrity menu resource and include it in your own menu definitions.

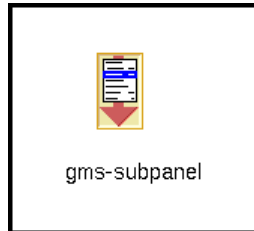
To clone a menu item from the Integrity main menu:

- 1 Select the item `opex-file` on the OPSCUI Developer Menu Resource workspace, select `clone`, move the item to your menu resource workspace, then click to drop the item.
- 2 Click on the yellow arrow under the item `opex-file` on the OPSCUI Developer Menu Resource workspace, select `clone`, move the item to your menu resource workspace, then click to drop the item.

This item is a `gms-submenu-connection-post`. It is used to connect a top-level menu definition to the items connected to a `gms-submenu-connection-post` with the same name at the other end.

- 3 Use the same cloning procedure to clone the `gms-subpanel` located under the label OPEX FILE MENU STUB.

A `gms-subpanel` is an object used to organize menu components. It provides a subworkspace to place sub-items of the menu. This item is shown below:



When an item is cloned, its attribute values are copied into the new item so you do not need to reconfigure the `gms-user-key` or the `gms-text-resource-group`. However, the name used for the `gms-submenu-connection-post` items is not copied so you need to enter a name for these items. Both connection posts should be assigned the same name.

To name a set of `gms-submenu-connection-posts`:

- 1 Select the `gms-submenu-connection-post`, select `name`, then enter the name for the connection post, which is *new-connection* for the example.
The other connection post is located on the subworkspace of the subpanel.
- 2 Click on the subpanel, and select `go to subworkspace`.
- 3 Click on the connection post at the top of the workspace, select `name`, then enter the same name as the name used for the connection post at the other end, *new-connection*.

In this example we do not want to include the File menu items `Modules...` or `Get Workspace` so we will delete them from the menu template items defined on the subpanel subworkspace.

To delete items from a cloned menu:

- 1 Select the `gms-submenu-connection-post` next to `opex-modules` and select `delete`.
- 2 Select the item `opex-modules` on the subpanel subworkspace, select `delete`, then click OK.
- 3 Select the `gms-separator-template` on the subpanel subworkspace, select `delete`, then click OK.

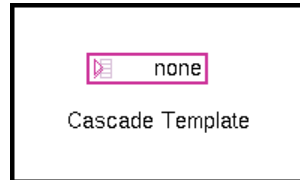
The separator template places across the submenu. The template item is a thick brown line.

- 4 Select the item `opex-get-workspace` on the subpanel subworkspace, select `delete`, then click OK.

- 5 Drag the connection to connect `opex-file` to `opex-clear-application`.
- 6 Click the X button to hide the subpanel subworkspace.

Creating a Cascade Menu Item

You use a `gms-cascade-template` to create a cascading menu item. This item is shown in the figure below:



If the `gms-top-level` workspace is not showing, use the Workspace finder to locate and view the `gms-top-level` workspace.

- 7 Select the cascade template on the Gensym Menu System workspace, move the item to your menu resource workspace, then click to drop the item.
- 8 Select the cascade template and select `table`.

Enter the following values in the attribute table:

- a `new-options` for the attribute `gms-label` and for `gms-key`.
 - b `new-text-resource` for the attribute `gms-text-resource-group`.
- 9 Configure the item as described in [Configuring a Menu Template Item](#) on page 69 entering `new-options` for the `Key...` and `Options` for the `Text...` in the local text resource.

Creating a Show Workspace Menu Item

Now we will add a new menu item that displays the `Docdemo-top-level` workspace. We will place this item under the main menu item `Options`.

The menu template used to create a menu item that displays workspace is a `gms-show-workspace-template`. This template is shown in the figure below:



To add a show workspace template:

- 1 If the `gms-top-level` workspace is not showing, using the Workspace finder, locate and view the `gms-top-level` workspace.
- 2 Select the show workspace template on the Gensym Menu System workspace, move the item to your menu resource workspace, then click to drop the item.
- 3 Select the show workspace template and select **table**.

Enter the following values in the attribute table:

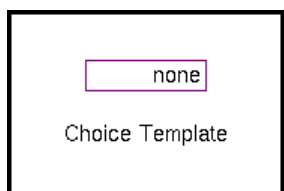
- a `show-top` for the attribute `gms-label` and for `gms-key`.
- b `new-text-resource` for the attribute `gms-text-resource-group`.
- c `docdemo-top-level` for the attribute `gms-display-target`.

You define the name of the workspace to display using the `gms-display-target` attribute. If you enter the name of an item, the subworkspace of the item is displayed.

- 4 Configure the item as described in [Configuring a Menu Template Item](#) on page 69 entering `show-top` for the Key... and `Show Top-Level Workspace` for the Text... in the local text resource.

Creating a Choice Menu Item

A choice menu item lets you call a procedure from a menu item. We will place this item under the show workspace item. The figure below shows a choice template:



The final item we will create in our example is a choice item that calls a procedure to display the doc-demo browser.

- 1 If the `gms-top-level` workspace is not showing, using the Workspace finder, locate and view the `gms-top-level` workspace.
- 2 Select the choice template on the Gensym Menu System workspace, move the item to your menu resource workspace, then click to drop the item.
- 3 Select the choice template and select **table**.

Enter the following values in the attribute table:

- a** *show-browser* for the attribute **gms-label** and for **gms-key**.
- b** *new-text-resource* for the attribute **gms-text-resource-group**.
- c** *new-call-browser* for the attribute **gms-activation-callback**.

In a choice template, the attribute **gms-activation-callback** defines the procedure called when you select the menu item. The procedure **new-call-browser** is described in the section *Displaying Browsers from a Menu* on page 75.

- 4** Configure the item as described in *Configuring a Menu Template Item* on page 69 entering *show-browser* for the *Key...* and *View Doc-Browser* for the *Text...* in the local text resource.

Displaying Browsers from a Menu

The procedure **doc-menu-call-view-browser** is defined in **doc_demo** to display the browser selected from the Operator Menu. This procedure is shown below:

```
doc-menu-call-view-browser (handle: integer, activation-path: item-or-value,
menu-index: integer )
item-selected: symbol;
win: class g2-window;

begin
item-selected = call gms-get-key-for-index (handle, menu-index);
win = call gms-get-window-for-handle (handle);
if item-selected = the symbol all
    then call opcsrui-view-browser (doc-browser, win)
    else call opcsrui-view-browser (doc-high-priority, win)
end
```

The procedure called to display the browser is:

```
opcsrui-view-browser ( browser-name: symbol, win: class G2-window )
```

Pass this procedure the name of the browser to display and the handle of the window where you want to display the browser. The window handle is retrieved by the GMS procedure **gms-get-window-for-handle**. This procedure returns the handle of the window containing the menu. **gms-get-key-for-handle** returns the menu key for the menu item selected.

Because our example directly calls one browser, we can use this procedure as a template.

To create a procedure to view the browser from the menu:

- 1 Select the background of the Menu Resource workspace, select new definition, procedure and procedure one more time.
- 2 Click to drop the definition of the procedure item on the workspace.
- 3 Select the procedure icon, select edit, then type in the following text:

```
new-call-browser (handle: integer, activation-path: item-or-value,  
menu-index: integer )  
win: class g2-window;  
  
begin  
win = call gms-get-window-for-handle (handle);  
call opcsrui-view-browser (doc-browser, win);  
end
```
- 4 Click End.

Connection Menu Items

Two types of connections are used for menu template items:

- Submenu connections - used to connect a cascade menu item to its sub-items. Submenu connections are green.
- Peer connections - used to connect a set of peer items together under the cascade menu item. Peer connections are yellow.

To create connection stubs on a menu template item:

- ➔ Select the menu template item and select add submenu stubs or add peer stubs depending on the type of stub you want to add.
- This adds menu stubs on all sides of the menu template item.

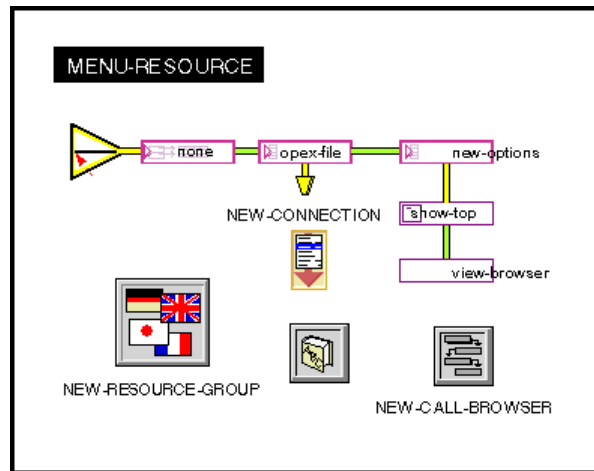
To remove unused stubs:

- ➔ Select the menu template item and select remove stubs.
- This remove stubs of both types of connections.

To connect the menu items:

- ➔ Click on the end of the connection stub, then drag the connection stubs to the item you want to connect, then click to connect the items.

Connect all of the menu items in the example. If you have followed along with this example, your completed menu should look like the one in the figure below:



Compiling the Menu

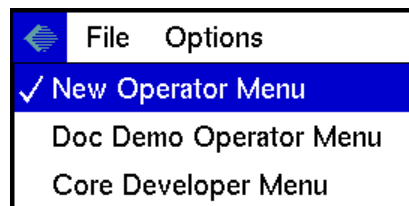
The final step to define the new menu is to compile the menu.

Tip Before compiling your menu, be sure that all the menu connections are firmly in place.

To compile a new menu resource:

- 1 Select the menu bar template. This is the triangular item at the left of the menu resource.
- 2 Select Compile all.

The top-level of the new operator menu defined in the example is shown in the figure below along with the expanded list of available menus:



If you get an error when you compile your menu, check each item and be sure that:

- The attribute values are correctly defined.
- An entry for the item is in the local text resource.
- All connections on the item are firmly in place.

After you identify and fix the error, try re-compiling the menu. If an error still occurs, the error might have corrupted the menu system and you might have to restart the system.

When you build your menu, it is best to build a small piece and then compile it to test it. If you build and compile incrementally, it is easier to find errors that might occur.

Defining Initializations

Initializations are objects that let you define initial values for any G2 item that can be assigned a value. Items assigned values are attributes of objects, G2 parameters, variables, arrays, and lists. You can use initializations to customize many aspects of the Integrity environment.

At startup, Integrity contains a number of predefined system initializations used to:

- Define the colors used to show the message priority and acknowledgment status of both the domain objects and the browser messages.
- Customize the way Integrity displays and handles error messages.
- Provide flexibility in defining how domain objects are retrieved.
- Allow custom procedures to be called as part of the execution of certain Integrity procedures.
- Define the workspaces used for certain operations within Integrity.

More than one initialization can target an item. When this occurs, the value of the initialization with the highest priority is used as the initial value of the item. Priorities range from 1 to 10, where 1 is the highest priority.

To view the items that have initializations defined:

➔ Choose Tools > Initializations from the Integrity main menu.

The function of the individual initializations is described in detail in the *Integrity Reference Manual*.

The figure below shows the current initializations table.

Type <	Target Name	Target Attribute	Module	Priority
Scalar	CDG-CORRELATION-MANAGERS-SERVER	NONE	ADV_DEMO	8
Scalar	CDG-DIAGNOSTIC-MESSAGES-SERVER	NONE	ADV_DEMO	8
Scalar	CDG-INCOMING-EVENTS-SERVER	NONE	ADV_DEMO	8
Scalar	ODG-ACTIVATE-INTERFACES-AT-STARTUP	null	ODG-CORE	10
Scalar	CDG-USER-EVENT-DELETION-METHOD	NONE	SVMDemo	9
Scalar	CDG-CORRELATION-MANAGERS-SERVER	NONE	SVMDemo	9
Scalar	CDG-DIAGNOSTIC-MESSAGES-SERVER	NONE	SVMDemo	9
Scalar	CDG-INCOMING-EVENTS-SERVER	NONE	SVMDemo	9
Scalar	CDG-INCOMING-EVENTS-SERVER	NONE	CDG_DEMO	3
Scalar	CDG-DIAGNOSTIC-MESSAGES-SERVER	NONE	CDG DEMO	3

Scalar Value
ADV-CORRELATION-MANAGERS

Procedure
null

Notes
OK

Description

Apply

Each item that has an initialization is shown in the scroll area. If the item targeted by the initialization is an object, the target attribute is also shown. Parameters, variables, arrays and lists do not have target attributes.

To view the value of an item with an initialization object:

➔ Click on the item in the scroll list.

The information shown includes:

- Description of the use of the initialization.
- Number of initialization objects defined for the selected item.
- Current value of the targeted item, this can be of the type scalar, vector, or GSI.

When an initialization is displayed in red with an exclamation point, it means the current value of the item is different than the value defined by the highest priority initialization item. Because item values can be changed as an application runs, this might not be a problem. The total number of items with values that do not match the initialization value is shown by the small red number above the scroll bar.

Creating an Initialization for a New Item

To create an initialization for a new item:

- 1 Click on the Initializations palette.
- 2 Click on the type of initialization to create. A dialog box appears, prompting you to specify the Destination Module.
- 3 Select the type of initialization object. Initialization objects can be one of:
 - a Scalar - a single value
 - b Vector - an array of values
 - c GSI - a special initialization used with interface objects
- 4 After selecting the correct module from the drop-down box, click OK. Integrity creates in the specified module an initialization. This initialization has the generic name of the type of initialization you selected.
- 5 Edit the initialization as discussed below and be sure to rename it to indicate its specific purpose. Be sure to specify the name of the target object. If the target is an object, specify which attribute of the object is the target of the initialization in Target Attribute.
- 6 Click OK.

Editing the Value of an Initialization

To edit the value of an initialization:

- 1 Choose Tools > Initializations from the Integrity main menu. The Current Initializations window appears.
- 2 Right-click on the name of the initialization object to edit.
- 3 Select View Properties. The properties dialog appears. You can change the properties of the initialization and set its value from this dialog box.

You can edit these values:

- Value - the value assigned to the target item on initialization. If there is more than one initialization for a specified target, the value of the initialization with the highest priority is assigned.
- Priority - the priority assigned to the initialization where 1 is the highest priority.
- Procedure - the name of an optional procedure which will be called when the target item is initialized.
- Description - a brief description of the function of the target item.

Note You cannot edit the initializations in system modules. You must define a new initialization for the same item.

Note When a new definition targets an item defined within Integrity, it overrides the system's default values.

Setting Preferences

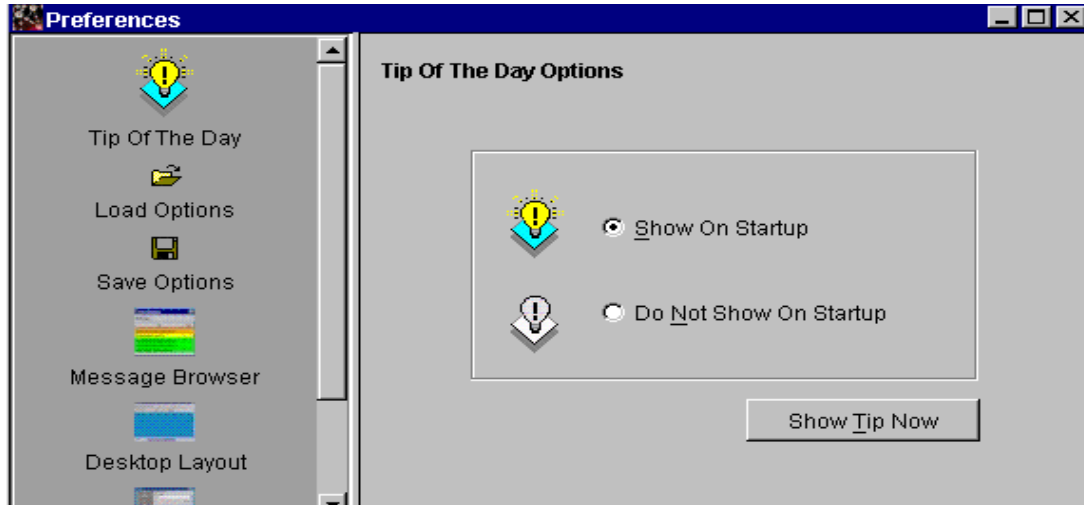
The Integrity Preferences window allows you to set preferences for using the Integrity client UI. To access the window, click on **Tools > Preferences**. The window is displayed.

The Preferences window allows you to set preferences for the following:

- Tip of the Day
- Load Options
- Save Options
- Message Browser
- Desktop Layout
- Finder Options
- Navigator Button

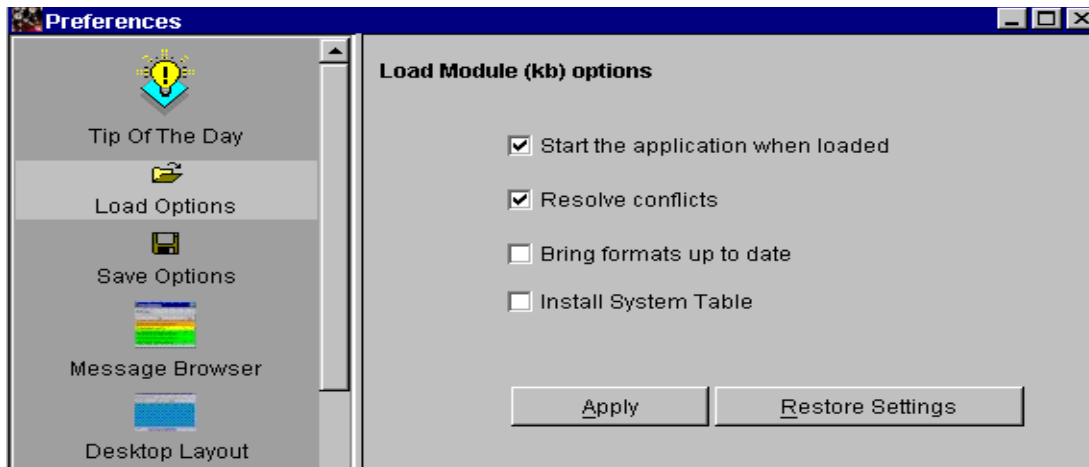
Tip of the Day Preferences

The window to set Tip of the Day preferences is shown below:



Load Options Preferences

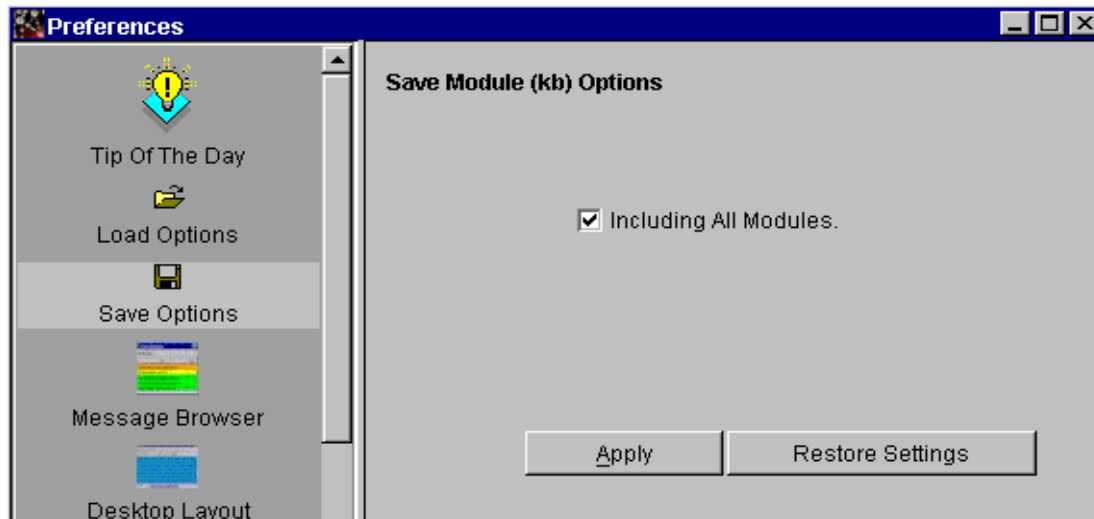
The window to set Load Option preferences is shown below:



This window allows you to set the options for loading a module.

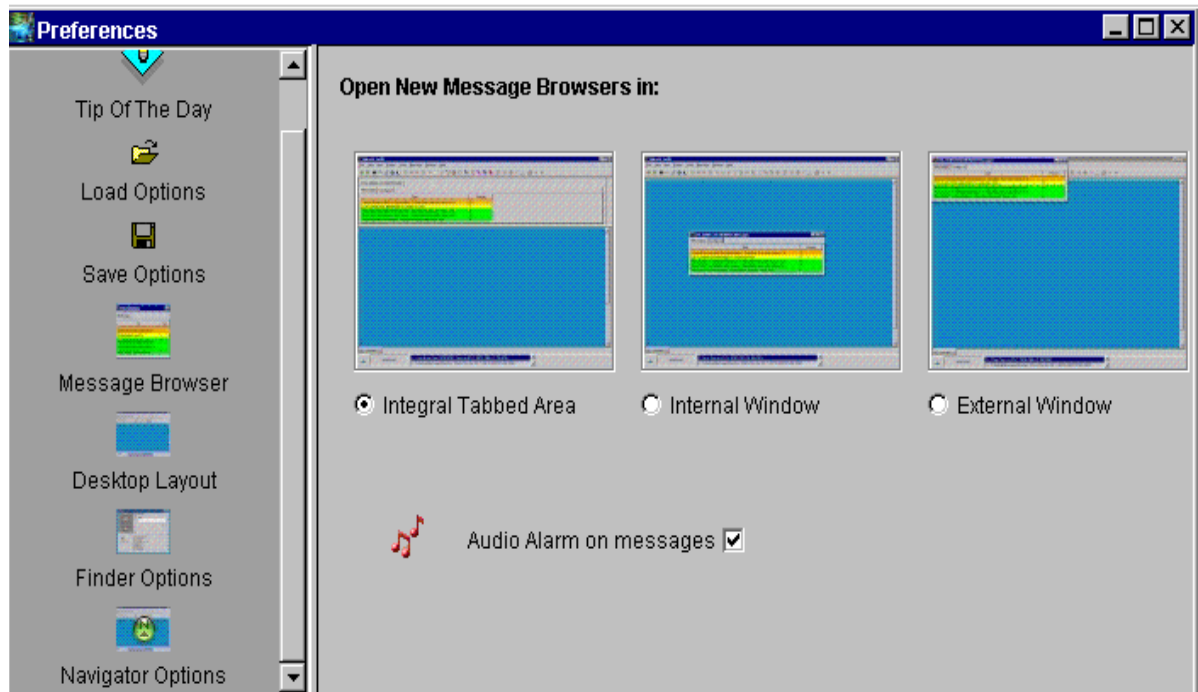
Save Options Preferences

The window to set save options preferences is shown below:



Message Browser Preferences

The window to set message browser preferences is shown below:



This window allows you set the way message browsers are displayed by checking the appropriate checkbox. You can display newly opened message browsers in:

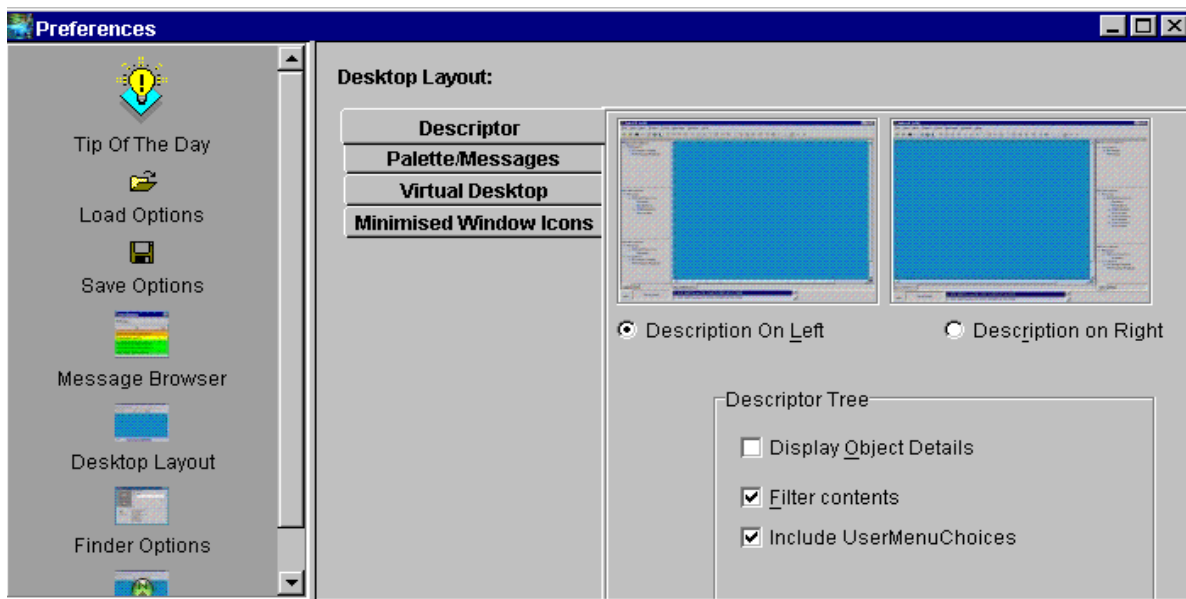
- The Integral Tabbed Area in the palettes/message browser area.
- An internal window
- An external window.

The Message Browser Preferences window also allows you select or deselect the audible alarm feature. When selected, this feature enables an audible alarm accompanies any messages of priority 1, 2, or 3.

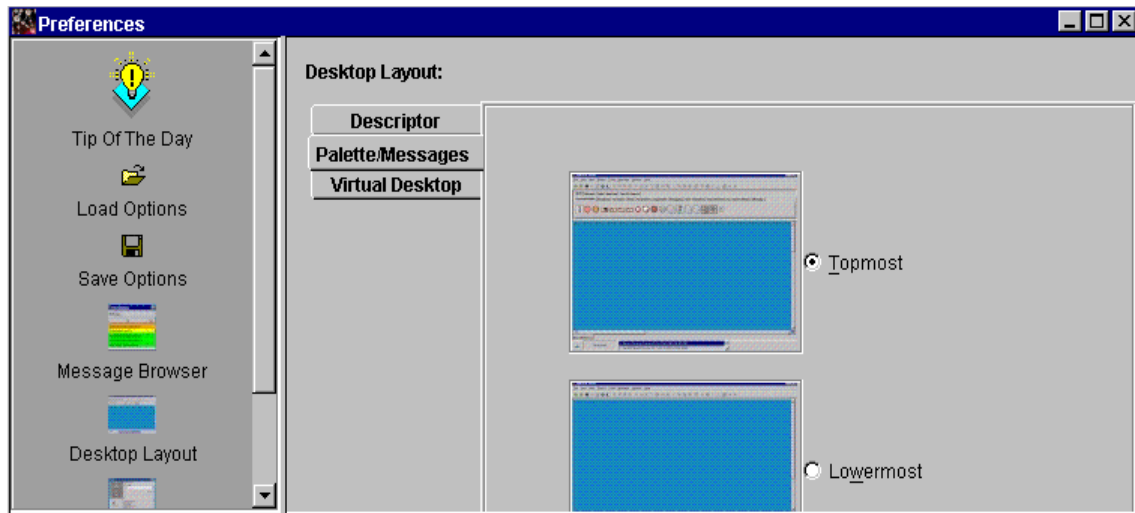
Desktop Layout Preferences

Desktop Layout Preferences allows you to customize how the Descriptors and Palette/Message area are positioned, and how to set the virtual desktop.

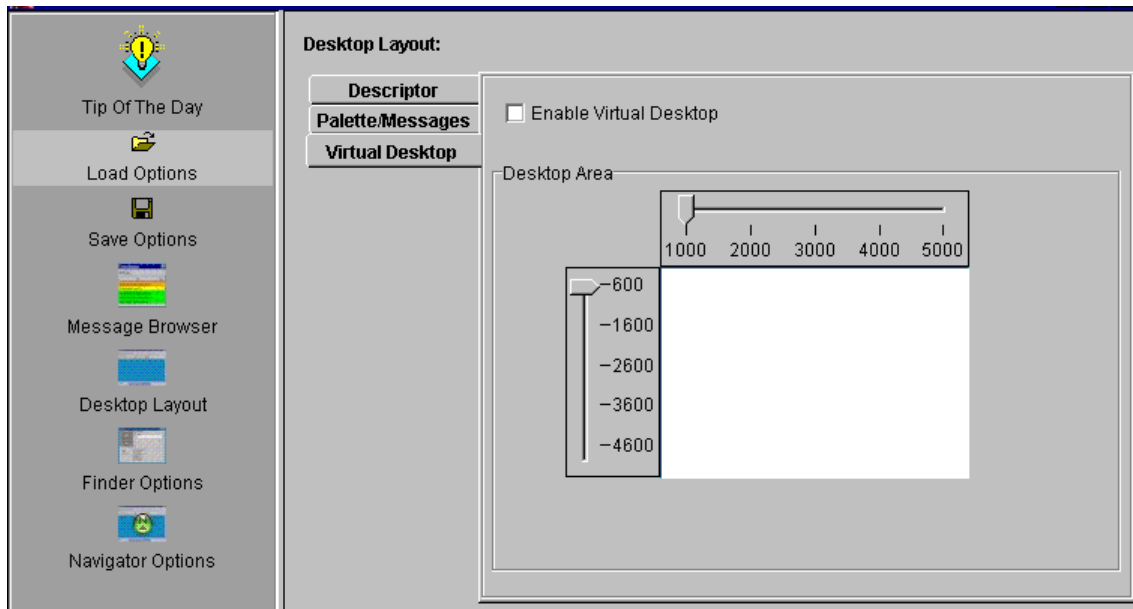
The window to set desktop layout descriptor preferences is shown below:



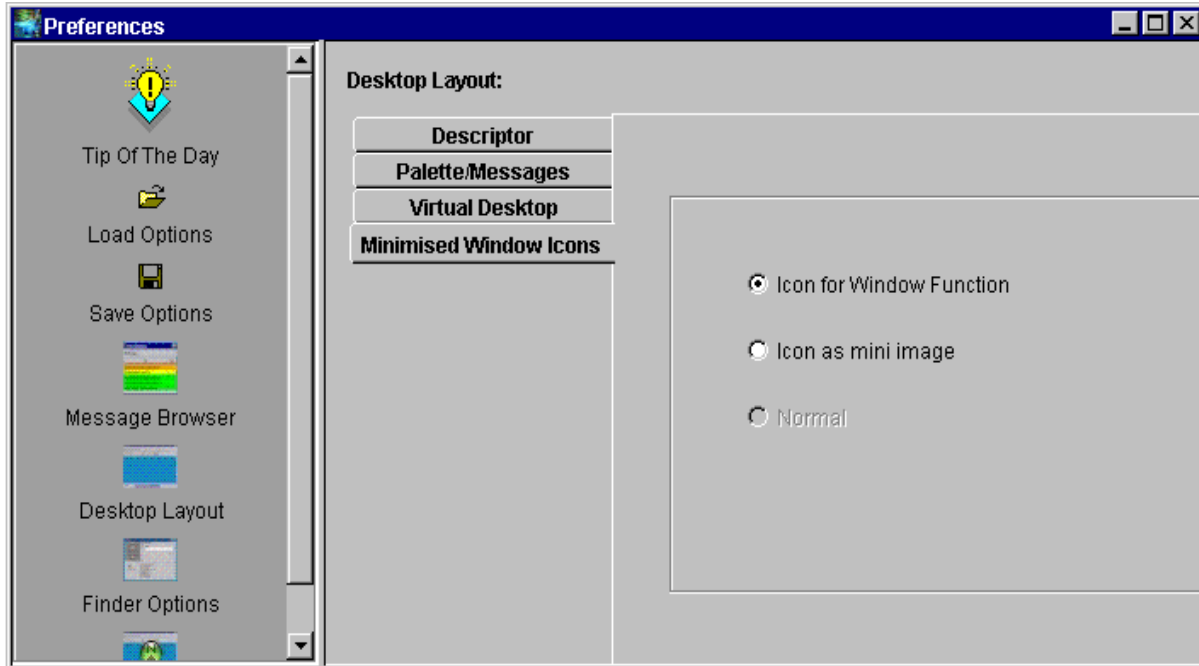
The window to set preferences for the desktop layout of the palette/messages area is shown below:



The window to set virtual desktop preferences is shown below:

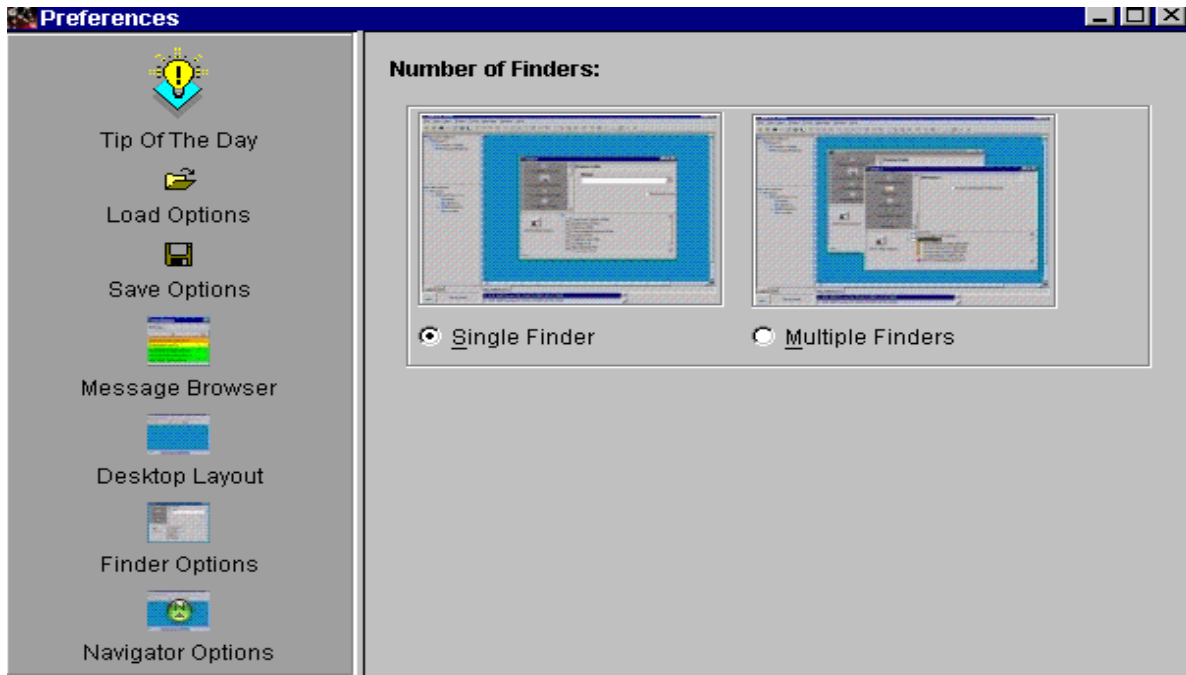


The window to set how minimized icons appear on the desktop is shown below:



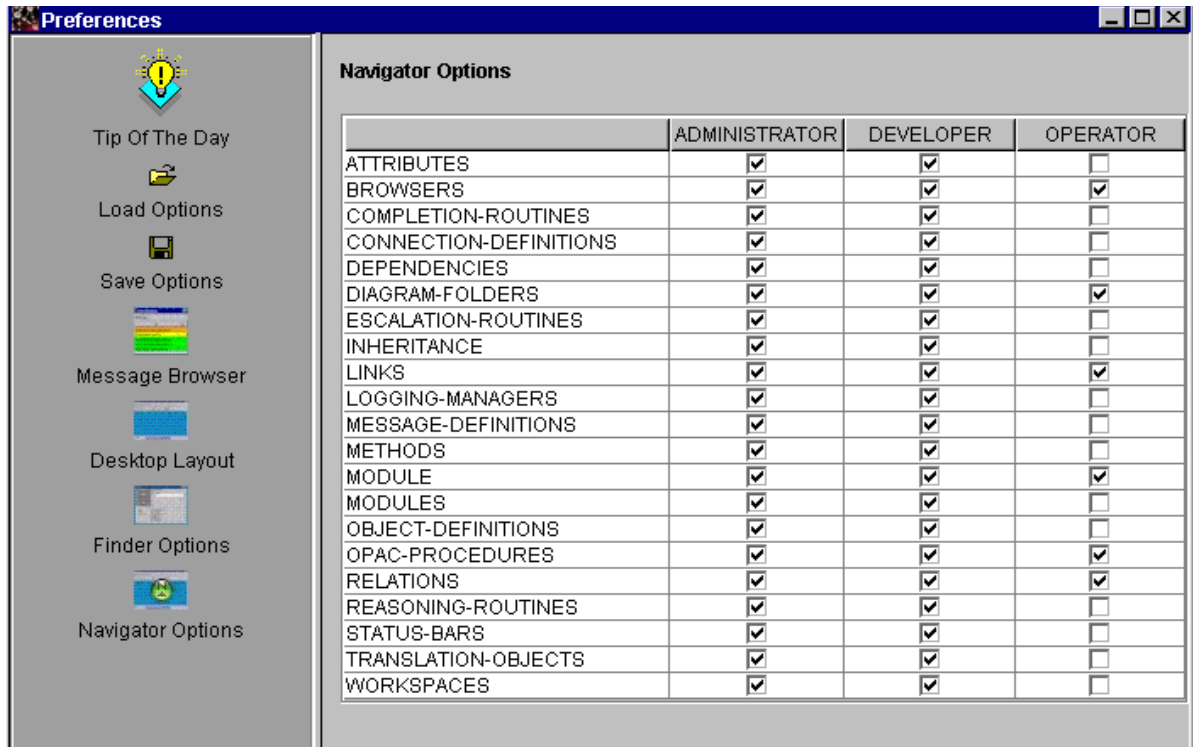
Finder Options Preferences

The window to set Finder options preferences is shown below:



Navigator Button Preferences

The Navigator preferences window allows you to set the options that are available to a particular user mode. The window to set Navigator preferences is shown below:

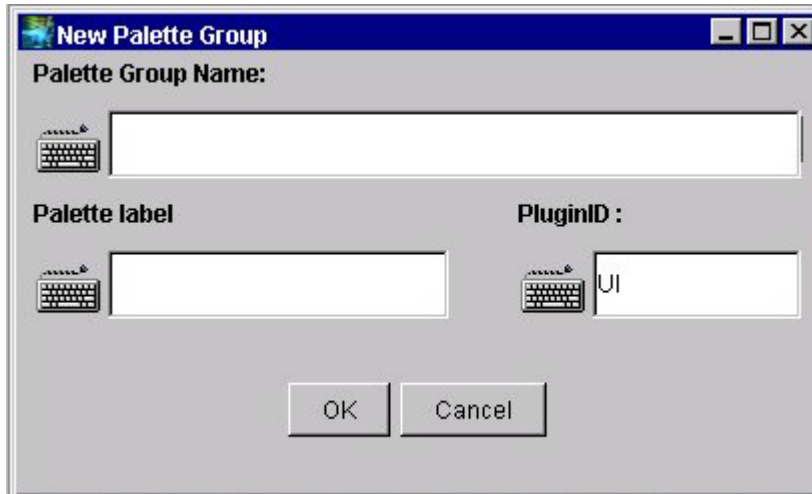


Creating New Palettes

New Palette Groups and Palettes can be added to the Integrity client UI. A Palette Group is a Tab component containing other Tab components comprising of class-definition icons.

Add a Palette Group

To add a Palette Group, move the mouse over the palette area and Right-Click. This will display the Palette Group menu. From this menu select *Add a new Group*. Once selected the New Palette Group dialog box will appear.

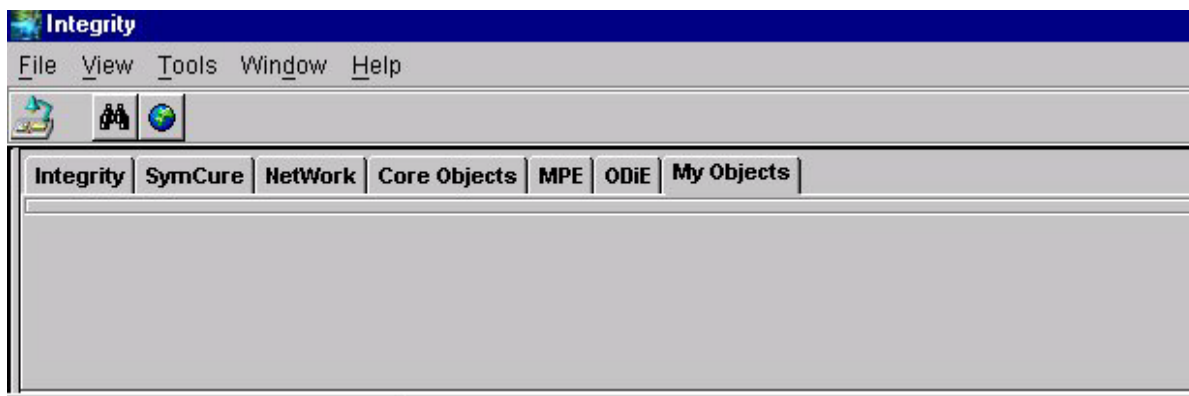


Enter the group name for the Palette Group Name. This will be the reference in the property files that are created and modified.

Enter a label for the Palette label. This will be displayed in the Tab area used for selecting the tab.

The PluginID contains the G2 module that is required to be loaded in order for the palette group to be accessible by the user. If this module is not required by your application, then the palette will not be loaded.

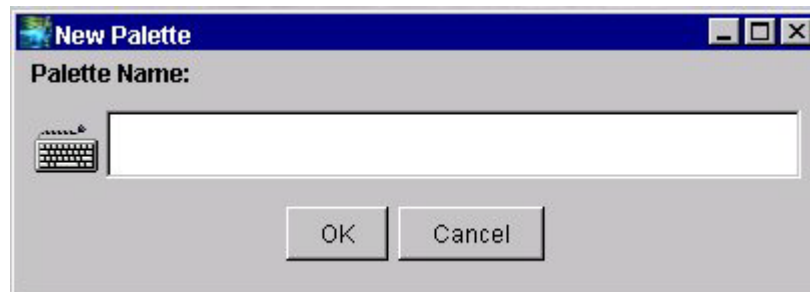
To create the Palette Group, select *OK* from the New Palette Group dialog. Once created the Palette Group is displayed in the palette area.



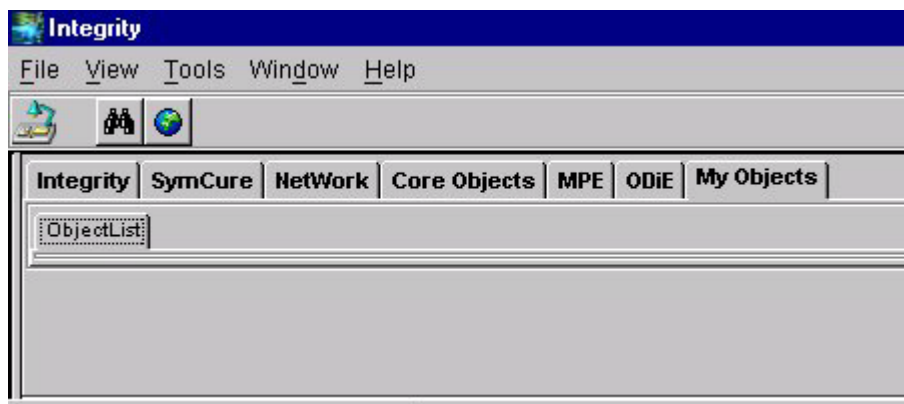
Add a Palette

A palette is a Tab component containing G2 class-definition icons. Once the palette and icons are created, users can drag the icons onto a workspace to create an instance of the class represented by the icon. To create a palette, position the mouse in the palette area and Right-Click. Select *Add Palette to Group* from the Palette Group menu.

The New Palette dialog will be displayed:



Simply enter the name for the palette in the Palette Name area. Once you select *OK* from the dialog the new palette will be created.



Adding Palette Items

Adding palette items will allow users to drag icons from the palette to a workspace to create an instance of a class. To create a palette item, move the mouse over the palette and Right-Click. Select *Edit palette* from the displayed menu. This will display the Palette Contents list.

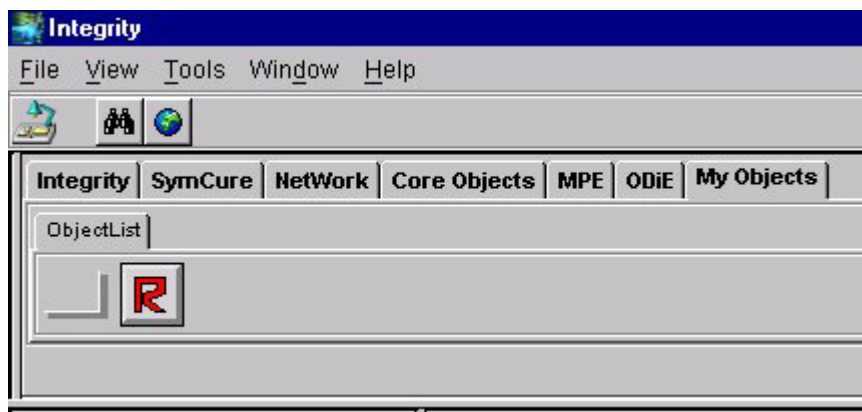
The Palette Contents list is divided into two lists. The list on the left are all classes defined in the connected G2. The list on the right is the list of classes to be displayed as your new palette. Some of these classes have icons displayed in the list along side the class name and some do not. Those that do, already have their icon as part of the client UI application. Once you add a class that does not have

an icon, displayed along side its class name, it will become part of the UI just as the others.

To create your list of palette items, select a class name from the Source List. Once an item is selected from the Source List the Right-Arrow-Button will be selectable. This allows you to select the button to move the selected class to the Palette Items list. The Up and Down arrows in the middle allow you to move the class names (icons) up or down to provide some type of ordering to your palette. The Left Arrow button allows you to remove the class name from the Palette List, and the Double-Left Arrow button allows you to remove all class names from the Palette List. Once all class names that you want are displayed in the Palette List, select the *OK* button to create the palette.

Note If a class name is not present in the Source List, make sure the class-definition exists in the connected G2 process.

After selecting OK, the class icons are placed in the palette.



The items on the palette are now ready to be dragged and placed on your workspace.

Property Files

The files listed below were either created or modified based on the addition of a group or palette:

- myobjectsPalette.properties
- userDefinedPalettes.properties

Note The file mentioned above, myobjectsPalette.properties, is listed because this is the example that was used for documentation purposes. The myobjects name comes from the information you entered on the previous dialogs.

If the object you placed on the palette are not part of the standard objects defined with Integrity, then a .jpg image is created and placed in the `%USERPROFILE%\opex\images\palettes` directory. This image is based on the icon-definition for the class-definition placed in your palette.

Customizing the User Interface Using Cyberformer

The cyberformer is an environment used to create and or modify the Integrity client UI. This includes the dialog title, icons, palettes, menu items and actions associated with the menu items.

Understanding Properties Files

The Integrity client UI allows the user to fully customize its features to meet customer-specific requirements. Instead of a hard-coded, unchangeable UI, Integrity uses an application generator which determines from a set of property files the features to build and present to the user as the Integrity client UI. (Even the name is an entry in a properties file). There is no class file (for example, `com.gensym.integrity.shell.Shell`) that serves as the UI.

The Integrity client UI builds from scratch each time you run it. Much of the form that it takes the behavior is determined by the contents of these properties files.

Origination and Purpose

Part of the function of the properties files is to allow you to define the context of an action without having to resort to hard coding it in java - its then the job of the CyberFormer class to decide figure out what bits of the UI is applicable at any one time.

What are Property files?

Think of these files as digital genes, the UI builder simply expresses what is encoded in these digital genes and creates what appears in the Integrity client UI. The purpose of this design is to allow users to alter the UI without too much difficulty or experience in Java programming.

The properties file include files that:

- Define the form of the UI - its menus / toolbars pallets etc.
- Define connection sessions/details
- Define layout/position of windows
- Define the contents of projects.

Although the format is basically the same, some files are simply ASCII text files designed to be edited by the user. Others contain binary information that should not be changed by direct editing, (basic tools are provided to manipulate some of these files).

Modify the digital genes and the result will be a different UI.

USER1 may not see exactly the same Integrity client UI as USER2 sees since either user may have made changes to his set of properties files.

Scope

Properties files affect:

- menus
- title bars
- tabbed dialogs
- actions related to menu items
- palettes
- labels next to the trees in the browser
- finders panel
- the "finder types" seen in the finder panel
- the properties seen under tools->properties
- the toolbar buttons
- the addition of Java Panels into the UI
- the footer at the bottom of the screen...
- the main panel seen in the UI
- the dynamic / class specific editor (allows you to specify a java class that supports visualization of the selected object -)
- the labels seen to the right of items in the trees - so you can display the attribute's to be part of the label.
- permitted G2 user modes
- login profiles
- projects and project contents
- NetSleuth connection details
- how the UI builds certain classes of objects
- the layout of the UI

- some behaviour of the finder
- the activation/size/behaviour of the Virtual Desktop

Location

The properties files are in the *USERPROFILE\opex* directory. The properties files must be in this directory. The application uses the files in this directory to load the UI. If you install the client for more than one user on the same machine, then an *opex* directory is created for each user under the user profiles directory and the properties files are copied to that directory.

For example: *C:\WINNT\Profiles\username\opex*.

If you need the original files as shipped, backup copies are included in *C:\gensym\OpEx30r0\Client\OPEXproperties*.

How the application loads the properties files:

When the application loads the properties files, it looks at the beginning of the *CLASSPATH* for the location of the files that it will use. If you want to modify a properties file that all users will share, you can put the location of this modified file in the *CLASSPATH*. For example, suppose you want several users to use one modified file. Logon as a user such as "common". Modify the properties file for the "common" user in the *opex* directory under the user profile directory for that user. For example, *c:\WINNT\Profiles\common\opex*. Then put this path in each user's *CLASSPATH*.

UI Structure

Instead of a hard coded UI the UI structure is in three parts.

- The Actions that perform some function.
- The menus and toolbars.
- Logic to determine applicability.

Rather than have each menu or menu item try and determine its applicability we have a central core which determines what is and what is not applicable at any given time.

How it determines applicability is largely controlled by ASCII text properties files, so again you can change the applicability without having to know about or change Java code.

Likewise the menus and the actions are all referenced from property files.

This is a simplified view, but basically it allows us to wire any action to any menu/toolbar button and to change the forms of the menus ... the hook between menu and action being soft wired at run time.

In fact the majority of the UI can be broken down and re assembled in a different configuration while the application is still running - allowing quite sophisticated changes to be made with minimum effort.

If an action is no longer applicable to a particular class of item, simply add that class to the exclusion key for the menu item, that's it - no delving into source code to figure out how the command file determines applicability.

Think of the central core as being like a switch center, or telephone exchange, it simply routes the input of the user, to a particular action... but with the ability to add new inputs and actions without taking the exchange down.

There are a number of keys which together combine to allow the UI to determine if a particular action is applicable at any given time (see Standard Menu Keys Standard Palette Keys).

While this scheme is okay for most customization, there may be times when you want to add checks that are not covered by this mechanism.

To allow you to easily slot in additional logic - we have the GoNoGo key; this allows you to write a Java class that holds all your additional logic... all it must do is implement the GoNoGo interface which basically returns true or false.

After all the "basic" checks for applicability have been run, if there is a go/nogo class associated with a particular action, the UI will consult that class before finally setting the action available or unavailable. The gonogo class has the last word on applicability, so if the "basic" checks (which are evaluated by the UI's inbuilt applicability logic) determine that an action is available, the gonogo class gives you the ability to override that value.

Functionality

This section describes the functionality of common properties files.

CyberFormer.properties

The most frequently modified file is the CyberFormer.Properties file. CyberFormer.properties is the "Granddaddy" of them all. It is the first one that the application calls on when loading the UI. It dictates what is on the menu, how the tabs are defined, how the toolbar is arranged, what modes allow various components to be seen, etc. Here is some code from this file to provide an example of what will get loaded and how these components are defined:

paletteImageFile=palette.images - This is used to indicate which serialized images to load.

palletGroups=_opacPallet _telecomsPallet etc, etc. Defines the palette groupings

Title=Integrity - this defines what is on the title bar.

Form :ASCII

Keys : 11 public

Title

String seen as the Application Name eg Integrity

TitleIcon icon to be used in the corner of frames

currentVersion String holding the current version of the constructed UI eg 3.0r1

panel The class of JPanel which will be instantiated as the main panel of the UI

footer The class of JPanel which will be instantiated as the footer of the UI

PanelAdditions : RESERVED

menubar A list of menu's which will be placed on the application frames menu bar

paletteImageFile The name of the image file holding the serialized images for the palettes

palletGroups A list of "Top level" palettes - each represents a distinct subsystem eg SYMCURE / G2

toolbarGroups A list of toolbar groups - each group represents a series of buttons which will appear in the toolbar of the application frame.

pluginRegistrationFile The name of the file which holds the list of currently registered subsystems

registeredPlugins.properties

Form : ASCII

Function : To allow actions to be attributed to a particular sub system - or plug-in

For example, menu items pertaining to SymCure are flagged as being part of the SYMCURE subsystem. Menus not part of a distinct subsystem are marked as belonging to the UI. Even if property files hold keys marked as belonging to SYMCURE, if SYMCURE is not a registered plugin, the UI builder will not include those menu, (this also extends to Finders)

Obsolete Property files:

DynamicFinderPanel

CdgExplorer

MsgBrowser

NeutralShellBuilder

opacPalette

opexPalette

runtimeWksp

systemWindowLocations

UIDescriptor

Some of the properties files create themselves:

- windowLocations.properties
- projects.properties

The advantage of this type of properties files is that, for example, if your windows have been moved to a point beyond where you can see them, you can delete windowLocations.properties file and it will recreate itself. Similarly, if something goes wrong with the projects, you can delete projects.properties file.

The registeredPlugins.properties file tells the UI what restrictions are allowed. For example, if you remove AUTODISCOVERY from this properties file, this user will not be able to access AUTODISCOVERY.

Here is a list of the remaining properties files.

- CdgExplorer.properties
- cdggui.properties
- CdgGuiLib.properties
- desktop.properties
- DynamicFinder.properties
- DynamicFinderPanel.properties
- EasyAccessUIJTreePanel.properties
- EditorDispatcher.properties
- g2control.properties
- g2corePalette.properties
- g2UserModes.properties
- gog2.properties
- LabelGeneratorDispatcher.properties
- lastActiveFile.properties
- lastLoadedFiles.properties
- lastOpenedFiles.properties
- LoginMangerFrame.properties
- LokupDriverDispatcher.properties

- `mainFrame.properties`
- `mibbrowser.properties`
- `MsgBroswer.properties`
- `NeutralShellBuilder.properties`
- `ObjectIndicatorImages.properties`
- `ObjMgrSupport.properties`
- `opacPalette.properties`
- `opexPalette.properties`
- `palette.properties`
- `pallet.properties` - outdated
- `projectmenu.properties`
- `PropertyViewer.properties`
- `RuntimeDynamicDisplay.properties`
- `runtimeWksp.properties`
- `sessionLibrary.properties`
- `symcurePalette.properties` - defines the palettes for the SymCure tab
- `systemWindowLocations.properties`
- `telcoPalette.properties`
- `ui.properties`
- `UIDescriptor.properties`
- `uitabs.properties`
- `userModes.properties` - defines the user modes
- `windowMenu.properties`

Keywords

In general, keywords exist in every properties file. Most keywords are pre-appended with a variable name that you define. The keyword is usually descriptive of its purpose. You assign values to the keywords.

To keep things simple, we will focus on the `CyberFormer.Properties` file and examine some of the keywords in this file.

`menubar` - allows you to create one or more new menu items.

To do this you declare and assign one or more variables to the keyword "menubar".

```
menubar=newitem1 newitem2.
```

Label - dictates what text will appear as a label for the user interface widget associated with that keyword. For example, if you create a new menu item and you want the menu item to appear as "Help If You Need It", you will assign a text value, "Help If You Need It" to the newitem1Label so that "Help If You Need It" appears on the menu bar. You do not need to put quotation marks around Help.

```
newitem1Label= Help If You Need It
```

Image - dictates the image that is associated with the item that you defined. Choose an image or create one. It can either be a .gif or .jpg file. All of the images that come with the Integrity package are located in jar files. This allows the image value to be a path images/nameOfImageFile.

```
newitem1Image=images/nameOfImageFile
```

Shortcut - defines the shortcut key for the item. Choose a character that exists in the label text.

```
newitem1ShortCut=H
```

IsPopup - tells the system to create a popup menu for this item when clicked on

```
newitem1IsPopup=true
```

ActionClass - tells the system to go to a particular java class that will cause an action to take place when the user chooses the menu item.

```
newitem1ActionClass=com.gensym.cdggui.ui.actions.g2server.  
ExportAllFileAction
```

Context - defines the modes that the user must be in to allow this item to function.

```
newitem1Context=ADMINISTRATOR DEVELOPER
```

Tooltip - determines the tool tip for the item

```
newitem1Tooltip=Create a new file
```

Cyberformer Reference

Syntax

Pound sign. Used for comments.

- Dash. Generates a horizontal line that separates menu items as they appear on any menu. Wherever you desire a separation on any menu, simply insert it in between any two variables that are listed in that menu.

_ Underscore. Pre-append it to any variable listed in any menu. It tells the system that this menu item will have a sub-menu associated with it. It generates an arrow to the right of the menu item represented by the variable pre-appended with this underscore.

Format:

Use "Title=" to create or modify the title that appears in the title bar of the client.

Title=OpEx

Choose an icon for the title bar. It can either be a .gif or .jpg file. Most of the image files are in the jar files. This allows the TitleIcon path to be images/.

TitleIcon=images/ico1.jpg

The icon used for the application when its iconified.

MDI=true

This indicates the current version of the client.

currentVersion= 3.5

This defines some of the components for the UI

opPanel=com.gensym.cdggui.ui.BaseOpPanel

adminPanel=com.gensym.cdggui.ui.BaseAdminPanel

panel=com.gensym.cdggui.ui.BaseIDEPanel

footer=com.gensym.cdggui.CdgComsIndicator

PanelAdditions=

MenuBar Declaration:

menubar=var1 var2 var3 var4 . . .

Where each variable reserves a space on the menu bar.

Menu item Label

var1Label=any text

var2Label=any text

where var1 and var2 are items on the menubar and "any text" is the label that will appear on the menu bar. (Do not put quotes around the text.)

Shortcut

var1Shortcut=any character

Choose a character that exists in the label text.

This is used to indicate which serialized images to load

```

paletteImageFile=palette.images
# Defines the palette groupings
palletGroups=_opacPallet _telecomsPallet _opexObjects _symcurePallet _
g2SystemPallet
#indicates which additional products are to be used
# G2 OPEX OPAC SYMCURE IPRA AUTODISCOVERY
pluginRegistrationFile=registeredPlugins
northToolBar=
centralToolBar=
southToolBar=
adminToolBar= pauseG2 resetG2 restartG2
# Defines the toolbar groupings
toolbarGroups=_uiControl _projects _finders _workspace _smhMessage _
symcureViewing _symcureChecking _symcureSetting
#_autoDiscovery
workspaceToolBarGroup= zoomIn zoomOut zoom2Fit zoom2FitAR
workspacePluginID=G2
projectsToolBarGroup=openThisProj closeThisProj addToProj rmFromProj
projectsPluginId=UI
autoDiscoveryToolBarGroup=discoveryTemplate - StartAutoDiscovery
StopAutoDiscovery StatAutoDiscovery PropertiesOfDiscovery - showWan
autoDiscoveryPluginID=AUTODISCOVERY
uiControlToolBarGroup=toggleDesc togglePallets tglControls
#tglControls
uiControlPluginID=UI
findersToolBarGroup= find viewObject props domainMap viewHistory
findersPluginID= UI
smhMessageToolBarGroup=viewMessages msgAck msgDel msgCom target
sender msgDet
smhMessagePluginID=OPEX
symcureSettingToolBarGroup=runTest falseValue trueValue
symcureSettingPluginID=SYMCURE

```

```

symcureCheckingToolBarGroup=configuration detectability isolatability getSig
symcureCheckingPluginID=SYMCURE
symcureViewingToolBarGroup=exploreDiagnosis mitiga exploreTests
conclusions
symcureViewingPluginID=SYMCURE
configureBlockLabel=Configure
configureBlockShortCut=V
configureBlockActiveWhenPaused=false
configureBlockAction=ConfigureBlock
configureBlockActionClass=com.gensym.cdggui.ui.actions.core.ConfigureBlock
configureBlockImage=images/ide.gif
configureBlockContext=com.gensym.classes.modules.opacore.
OpacLanguageClass
configureBlockGoNoGoClass=com.gensym.cdggui.ui.actions.core.ViewGoNoGo
configureBlockIsPopup=true
smhmessage= viewMessages - msgAck msgDel msgDet msgCom mitiga
exploreTests conclusions exploreDiagnosis
smhmessageLabel=Message
smhmessageShortCut=M
smhmessagePluginID=OPEX
getSigLabel=Get Signature
getSigShortCut=G
getSigAction=GetSignature
getSigActionClass=com.gensym.cdggui.ui.actions.symcure.GetSignatureAction
getSigTooltip=Graphical representation of the fault
#getSigUserModeContext=ADMINISTRATOR DEVELOPER
getSigImage=images/fsig.gif
getSigContext=com.gensym.classes.modules.cdg.CdgGenericFault
getSigIsPopup=true
tglControlsLabel=Messages // Palettes
tglControlsShortCut=s
tglControlsImage=images/a125.gif

```

tglControlsAction=TglControls
tglControlsActionClass=com.gensym.cdggui.ui.actions.core.TglControlsAction
tglControlsTooltip=Toggle Messages <-> Palettes
tglControlsUserModeContext= ADMINISTRATOR DEVELOPER
zoomInLabel=Zoom In
zoomInShortCut=I
zoomInImage=images/zin.gif
zoomInContext=com.gensym.classes.KbWorkspace
zoomInAction=ZoomIn
zoomInActionClass=com.gensym.cdggui.ui.actions.zoom.ZoomInAction
zoomInTooltip=Zoom In
zoomInIsPopup=true
zoom2FitARContext=com.gensym.classes.KbWorkspace
zoom2FitARLabel=Aspect Ratio 1:1
zoom2FitARShortCut=1
zoom2FitARImage=images/zfitar.gif
zoom2FitARTooltip=set aspect ratio of the workspace to 1:1
zoom2FitARAction=ZoomToFitAR
zoom2FitARActionClass=com.gensym.cdggui.ui.actions.zoom.
ZoomFitARAction
zoom2FitARIsPopup=true
zoom2FitContext=com.gensym.classes.KbWorkspace
zoom2FitLabel= Variable X:Y aspect ratio
zoom2FitShortCut=F
zoom2FitImage=images/zfit.gif
zoom2FitTooltip=Set the aspect ratio to fit in the window
zoom2FitAction=ZoomToFit
zoom2FitActionClass=com.gensym.cdggui.ui.actions.zoom.ZoomFitAction
zoom2FitIsPopup=true
zoomOutLabel=Zoom Out
zoomOutShortCut=O


```
zoomOutImage=images/zout.gif
zoomOutTooltip=Zoom Out
zoomOutAction=ZoomOut
zoomOutActionClass=com.gensym.cdggui.ui.actions.zoom.ZoomOutAction
zoomOutContext=com.gensym.classes.KbWorkspace
zoomOutIsPopup=true
## MENU ITEMS ##
viewObjectLabel=View Object
viewObjectShortCut=V
viewObjectActiveWhenPaused=false
viewObjectAction=ViewObject
viewObjectActionClass=com.gensym.cdggui.ui.actions.core.ViewAction
viewObjectImage=images/gotodot.gif
viewObjectExclude=java.lang.String com.gensym.util.Symbol com.gensym.util.
Sequence com.gensym.util.Structure com.gensym.classes.ModuleInformation
com.gensym.opex2000.project.Project
viewObjectGoNoGoClass=com.gensym.cdggui.ui.actions.core.ViewGoNoGo
viewObjectIsPopup=true
viewHistoryLabel=History
viewHistoryShortCut=H
viewHistoryActiveWhenPaused=false
viewHistoryAction=ViewHistory
viewHistoryActionClass=com.gensym.cdggui.ui.actions.core.
ShowHistoryAction
viewHistoryImage=images/showhist.gif
#snapshotExport
export= domainmapExport
exportLabel=Export
exportShortCut=E
exportImage=images/noimage.gif
exportUserModeContext=
allFileLabel=Save All file
```

allFileShortCut=A
allFileImage=images/save_as_all.gif
allFileAction=ExportAllFile
allFileActionClass=com.gensym.cdggui.ui.actions.g2server.ExportAllFileAction
moduleExportLabel=Save Module
moduleExportShortCut=M
moduleExportImage=images/modulesave.gif
moduleExportAction=ExportKb
moduleExportActionClass=com.gensym.cdggui.ui.actions.g2server.
ExportKbAction
moduleExportContext=com.gensym.classes.ModuleInformation
moduleExportUserModeContext=ADMINISTRATOR DEVELOPER
snapshotExportLabel=Snapshot
snapshotExportShortCut=S
snapshotExportImage=images/snap.gif
snapshotExportAction=ExportSnapshot
snapshotExportActionClass=com.gensym.cdggui.ui.actions.g2server.
ExportSnapshotAction
domainmapExportLabel=Domain Map
domainmapExportShortCut=D
domainmapExportImage=images/export-map.gif
domainmapExportAction=ExportDomainMap
domainmapExportActionClass=com.gensym.cdggui.ui.actions.opex.
ExportDomainMapAction
#snapshotImport
import= domainmapImport
importLabel=Import
importShortCut=I
importUserModeContext=ADMINISTRATOR DEVELOPER
importImage=images/noimage.gif
moduleImportLabel=Add Module
moduleImportShortCut=M

```
moduleImportImage=images/moduleadd.gif
moduleImportUserModeContext=ADMINISTRATOR DEVELOPER
moduleImportAction=ImportKb
moduleImportActionClass=com.gensym.cdggui.ui.actions.g2server.
ImportKbAction
snapshotImportLabel=Snapshot
snapshotImportShortCut=S
snapshotImportImage=images/snap.gif
snapshotImportUserModeContext=ADMINISTRATOR DEVELOPER
snapshotImportAction=ImportSnapshot
snapshotImportActionClass=com.gensym.cdggui.ui.actions.g2server.
ImportSnapshotAction
domainmapImportLabel=Domain Map
domainmapImportShortCut=D
domainmapUserModeContext=ADMINISTRATOR DEVELOPER
domainmapImportImage=images/import-map.gif
domainmapImportAction=ImportDomainMap
domainmapImportActionClass=com.gensym.cdggui.ui.actions.opex.
ImportDomainMapAction
renameLabel=Rename
renameShortCut=r
renameImage=images/noimage.gif
renameAction=Rename
renameActionClass=com.gensym.cdggui.ui.actions.g2server.
RenameObjectAction
renameGoNoGoClass=com.gensym.cdggui.ui.actions.g2server.
RenameObjectGoNoGo
propsLabel=Properties...
propsIsPopup=true
propsShortCut=P
propsImage=images/properties.gif
propsAction=G2Properties
```

```

propsActionClass=com.gensym.cdggui.ui.actions.g2server.
ShowPropertiesAction

propsGoNoGoClass=com.gensym.cdggui.ui.actions.g2server.PropertiesGoNoGo

propsUserModeContext=ADMINISTRATOR DEVELOPER

propsExclude=java.lang.String com.gensym.util.Symbol com.gensym.util.
Sequence com.gensym.util.Structure com.gensym.opex2000.project.Project

runTestLabel=Run Test

runTestShortCut=T

runTestImage=images/runtest.gif

runTestAction=RunTest

runTestActionClass=com.gensym.cdggui.ui.actions.symcure.RunTestAction

runTestContext=com.gensym.classes.modules.cdg.CdgSpecificTest

runTestIsPopup=true

webLinkPanelTabLabel=WebLink

webLinkPanelClass=com.gensym.cdggui.CdgHtmlPanel

webLinkPanelTabImage=images/bmcpnl.gif

# file Menu definition

file=_newInstance - _newProj openProj - save saveAs - saveProject removeProject
- load allFile moduleExport moduleImport unload - _import _export - rename -
print printPreview - exit

printPreviewLabel=Print Preview...

printPreviewShortCut=r

printPreviewImage=images/noimage.gif

printPreviewAction=PrintPreview

printPreviewActionClass=com.gensym.cdggui.ui.actions.core.
PrintPreviewAction

printPreviewContext=com.gensym.classes.KbWorkspace

printSetupLabel=Print Setup...

printSetupShortCut=s

printSetupImage=images/noimage.gif

printSetupAction=PrintSetup

#printSetupActionClass=com.gensym.cdggui.ui.actions.core.PrintPreviewAction

```

```
printLabel=Print
printShortCut=P
printImage=images/print.gif
printAction=Print
printActionClass=com.gensym.cdggui.ui.actions.core.PrintAction
printContext=com.gensym.classes.KbWorkspace
senderLabel=Goto Sender
senderShortCut=S
senderImage=images/sender.gif
senderActiveWhenPaused=false
senderAction=GotoSender
senderActionClass=com.gensym.cdggui.ui.GotoSenderAction
senderContext=com.gensym.classes.modules.smh.SmhTransientMessage com.
gensym.classes.modules.cdg.CdgSpecificFault com.gensym.classes.modules.cdg.
CdgSpecificOrPropagation com.gensym.classes.modules.cdg.
CdgSpecificAndPropagation
senderIsPopup=true
mitigaLabel=Suspect Faults
mitigaShortCut=F
mitigaImage=images/mfault.gif
mitigaWhenPaused=false
mitigaAction=SuspectMitigations
mitigaTooltip=See Suspect Faults
mitigaActionClass=com.gensym.cdggui.ui.actions.messagebrowser.
SuspectMitigationsAction
mitigaGoNoGoClass=com.gensym.cdggui.ui.actions.messagebrowser.
CheckForDiagnosis
mitigaContext=com.gensym.classes.modules.smh.SmhTransientMessage com.
gensym.classes.modules.cdg.CdgCorrelationManager
mitigaIsPopup=true
exploreTestsLabel=Candidate Tests
exploreTestsShortCut=T
exploreTestsImage=images/mtests.gif
```

exploreTestsWhenPaused=false
exploreTestsTooltip=See Candidate Tests
exploreTestsAction=Tests
exploreTestsActionClass=com.gensym.cdggui.ui.actions.messagebrowser.
ExploreTestsAction
exploreTestsContext=com.gensym.classes.modules.smh.SmhTransientMessage
com.gensym.classes.modules.cdg.CdgCorrelationManager
exploreTestsIsPopup=true
msgAckLabel=Acknowledge
msgAckrShortCut=A
msgAckImage=images/ack.gif
msgAckActiveWhenPaused=false
msgAckTooltip=Acknowledge the message
msgAckAction=MsgAck
msgAckActionClass=com.gensym.cdggui.ui.actions.messagebrowser.
AckMsgAction
msgAckContext=com.gensym.classes.modules.smh.SmhTransientMessage
msgDelLabel=Delete
msgDelShortCut=D
msgDelImage=images/delete.gif
msgDelActiveWhenPaused=false
msgDelAction=DelMsg
msgDelTooltip=Delete the message from the browser
msgDelActionClass=com.gensym.cdggui.ui.actions.messagebrowser.
DelMsgAction
msgDelGoNoGoClass=com.gensym.cdggui.ui.actions.messagebrowser.
DelMsgGoNoGo
msgDelContext=com.gensym.classes.modules.smh.SmhTransientMessage
msgDetLabel=Detail
msgDetShortCut=T
msgDetImage=images/moreinfo.gif
msgDetActiveWhenPaused=false

msgDetAction=MsgDetail
msgDetTooltip=Get further details on this message
msgDetActionClass=com.gensym.cdggui.ui.actions.messagebrowser.
MsgDetailsAction
msgDetContext=com.gensym.classes.modules.smh.SmhTransientMessage
exploreDiagnosisLabel=Diagnosis Detail
exploreDiagnosisShortCut=F
exploreDiagnosisImage=images/diagdet.gif
exploreDiagnosisActiveWhenPaused=false
exploreDiagnosisAction=Diagnosis
exploreDiagnosisTooltip=Get the Diagnosis Detail
exploreDiagnosisActionClass=com.gensym.cdggui.ui.actions.messagebrowser.
ExploreDiagnosisAction
exploreDiagnosisGoNoGoClass=com.gensym.cdggui.ui.actions.messagebrowser.
CheckForDiagnosis
exploreDiagnosisContext=com.gensym.classes.modules.smh.
SmhTransientMessage com.gensym.classes.modules.cdg.
CdgCorrelationManager
exploreDiagnosisIsPopup=true
msgComLabel=Add Comments
msgComShortCut=T
msgComImage=images/addcom.gif
msgComActiveWhenPaused=false
msgComAction=MsgComment
msgComTooltip=Add a Comment to the Message
msgComActionClass=com.gensym.cdggui.ui.actions.messagebrowser.
MsgCommentsAction
msgComContext=com.gensym.classes.modules.smh.SmhTransientMessage
conclusionsLabel=Known Faults
conclusionsShortCut=F
conclusionsImage=images/nfaults.gif
conclusionsActiveWhenPaused=false
conclusionsAction=Conclusions

conclusionsTooltip=See Known Faults
conclusionsActionClass=com.gensym.cdggui.ui.actions.messagebrowser.
ConclusionMitigationsAction
conclusionsGoNoGoClass=com.gensym.cdggui.ui.actions.messagebrowser.
CheckForDiagnosis
conclusionsContext=com.gensym.classes.modules.smh.SmhTransientMessage
com.gensym.classes.modules.cdg.CdgCorrelationManager
conclusionsIsPopup=true
newApplicationLabel=New Application
newApplicationShortCut=N
newApplicationImage=images/closebrowser.gif
newApplicationAction=New Application
newApplicationActionClass=com.gensym.cdggui.ui.actions.opex.
NewApplicationAction
newApplicationUserModeContext=ADMINISTRATOR
dynamicViewerLabel=Dynamic Viewer
dynamicViewerAction=DynamicEditor
dynamicViewerImage=images/newclass.gif
dynamicViewerShortCut=V
dynamicViewerActionClass=com.gensym.cdggui.ui.actions.core.
ClassSensitivePanelAction
openProjLabel=Open Project
openProjAction=OpenProj
openProjImage=images/opproj.gif
openProjShortCut=O
openProjActionClass=com.gensym.opex2000.actions.project.OpenProjectAction
openProjContext= com.gensym.opex2000.project.Project
addToProjLabel=Add To Project
addToProjAction=AddToProject
addToProjImage=images/addprj.gif
addToProjShortCut=A
addToProjActionClass=com.gensym.opex2000.actions.project.
AddToProjectAction


```

addToProjGoNoGoClass=com.gensym.opex2000.actions.project.
AddToProjectGoNoGo

addToProjIsPopup=true

#addToProjExclude=java.lang.String com.gensym.util.Symbol com.gensym.util.
Sequence com.gensym.util.Structure

addToProjContext=com.gensym.classes.ModuleInformation com.gensym.classes.
KbWorkspace

rmFromProjLabel=Remove From Project
rmFromProjAction=RemoveObjectFromProject
rmFromProjImage=images/rmprj.gif
rmFromProjShortCut=r
rmFromProjActionClass=com.gensym.opex2000.actions.project.
RemoveFromProjectAction
rmFromProjGoNoGoClass=com.gensym.opex2000.actions.project.
RemoveFromProjectGoNoGo
rmFromProjIsPopup=true
rmFromProjContext=com.gensym.classes.ModuleInfo com.gensym.classes.
KbWorkspace
#rmFromProjExclude=java.lang.String com.gensym.util.Symbol com.gensym.
util.Sequence com.gensym.util.Structure
newProj=newPersonalProject newGroupProject
newProjLabel=New Project
newProjShortCut=P
newProjImage=images/newprj.gif
newPersonalProjectLabel=Personal Project
newPersonalProjectShortCut=P
newPersonalProjectImage=images/smallpeprj.gif
newPersonalProjectAction=NewPersProj
newPersonalProjectActionClass=com.gensym.opex2000.actions.project.
NewPersonalProjectAction
newGroupProjectLabel=Group Project
newGroupProjectShortCut=G
newGroupProjectImage=images/smallgrprj.gif

```

newGroupProjectAction=NewGrpProj
newGroupProjectActionClass=com.gensym.opex2000.actions.project.
NewGroupProjectAction
newGroupProjectUserModeContext=ADMINISTRATOR DEVELOPER
newInstance= newApplication - newModule newWS - newItem newObject
newMessage
#- discoveryTemplate
newInstanceLabel=New
newInstanceShortCut=N
newInstanceImage=images/new.gif
discoveryTemplateLabel=Discovery Job
discoveryTemplateShortCut=O
discoveryTemplateImage=images/newdisc.gif
discoveryTemplateAction=NewDiscovery
discoveryTemplateActionClass=com.gensym.opex2000.actions.nm.
NewDiscoveryAction
newObjectLabel=Object...
newObjectShortCut=O
newObjectImage=images/new.gif
newObjectAction=NewObjectInstance
newObjectActionClass=com.gensym.cdggui.ui.actions.core.NewObjectAction
newObjectContext=com.gensym.classes.KbWorkspace
newMessageLabel=Message...
newMessageShortCut=M
newMessageImage=images/mb.gif
newMessageAction=NewMessageInstance
newMessageActionClass=com.gensym.cdggui.ui.actions.core.
NewMessageAction
newMessageContext=com.gensym.classes.KbWorkspace
newItemLabel=Item...
newItemShortCut=I
newItemImage=images/blueball.gif

```
newItemAction=NewItemInstance
newItemActionClass=com.gensym.cdggui.ui.actions.core.NewItemAction
newItemContext=com.gensym.classes.KbWorkspace
newWSLabel=Workspace
newWSShortCut=W
newWSImage=images/new_wksp.gif
newWSAction=NewWorkSpace
newWSActionClass=com.gensym.cdggui.ui.actions.core.NewWSAction
newModuleLabel=Module
newModuleShortCut=D
newModuleImage=images/modules.gif
newModuleAction=NewModule
newModuleActionClass=com.gensym.cdggui.ui.actions.core.NewModuleAction
newModuleUserModeContext=ADMINISTRATOR DEVELOPER
fileLabel=File
fileShortCut=F
filePluginID=UI
connection= openConnection closeConnection
connectionLabel=Connection
connectionShortCut=C
openConnectionLabel=Open
openConnectionShortCut=O
closeConnectionLabel=Close
closeConnectionShortCut=C
opex=relink initializations
opexLabel=OpEx
opexShortCut=O
relinkLabel=Relink Escalation Procedures
relinkShortCut=R
relinkAction=EscalationSpecifications
```

relinkActionClass=com.gensym.cdggui.ui.actions.opex.
EscalationSpecificationsAction
relinkImage=images/relink.gif
initializationsLabel=Initializations...
initializationsShortCut=i
initializationsAction=Initializations
initializationsActionClass=com.gensym.cdggui.ui.actions.opex.
InitializationsAction
initializationsImage=images/devinit.gif
methodsLabel=Methods
methodsAction=GetMethods
methodsImage=images/methods.gif
methodsActionClass=
#This is a temporary feature untill I hook in to mode changes.
#But it will be useful when putting the tool in menu edit mode
reloadLabel=Load Properties
reloadShortCut=R
reloadAction=reloadProperties
reloadActionClass=ReloadPropertiesAction
reloadImage=images/reload.gif
reloadUserModeContext=ADMINISTRATOR
openLabel=Load
openShortCut=L
openImage=images/open.gif
openAction=LoadKb
openActionClass=com.gensym.cdggui.ui.actions.g2server.LoadKbAction
loadLabel=Load
loadShortCut=L
loadImage=images/open.gif
loadAction=LoadKb
loadActionClass=com.gensym.cdggui.ui.actions.g2server.LoadKbAction

```
loadUserModeContext=ADMINISTRATOR DEVELOPER
unloadLabel=Remove Module
unloadShortCut=R
unloadImage=images/modulerem.gif
unloadAction=UnloadKb
unloadActionClass=com.gensym.cdggui.ui.actions.g2server.UnloadKbAction
unloadContext=com.gensym.classes.ModuleInformationImpl
unloadUserModeContext=ADMINISTRATOR DEVELOPER
#new=newClassDefinition
saveLabel=Save
saveShortCut=S
saveImage=images/save.gif
saveActionClass=com.gensym.cdggui.ui.actions.g2server.SaveModuleAction
saveAction=Save
saveAsLabel=Save As...
saveAsShortCut=A
saveAsImage=images/noimage.gif
saveAsActionClass=com.gensym.cdggui.ui.actions.g2server.
SaveModuleAsAction
saveAsAction=SaveAs
saveProjectLabel=Save Project As...
saveProjectShortCut=P
saveProjectImage=images/saveproj.gif
saveProjectActionClass=com.gensym.opex2000.actions.project.
SaveProjectAsAction
saveProjectAction=SaveProjectAs
saveProjectContext=com.gensym.opex2000.project.Project
removeProjectLabel=Remove Project
removeProjectShortCut=R
removeProjectImage=images/delproj.gif
removeProjectActionClass=com.gensym.opex2000.actions.project.
RemoveProjectAction
```

removeProjectAction=RemoveProject
removeProjectContext=com.gensym.opex2000.project.Project
RemoveProjectGoNoGoClass=com.gensym.opex2000.actions.project.
RemoveProjectGoNoGo
exitLabel=Exit
exitShortCut=x
exitAction=Exit
exitActionClass=com.gensym.cdggui.ui.actions.core.ExitAction
exitTooltip=Closedown this client
exitImage=images/exit.gif
project=descProj openThisProj closeThisProj
projectLabel=Project
projectShortCut=p
descProjLabel=Describe Project
descProjAction=DescribeProject
descProjShortCut=D
descProjActionClass=com.gensym.opex2000.actions.project.
ProjectDescribeAction
descProjContext=com.gensym.opex2000.project.Project
descProjIsPopup=true
openThisProjLabel=Make Active project
openThisProjAction=ProjectOpen
openThisProjImage=images/opproj.gif
openThisProjShortCut=A
openThisProjActionClass=com.gensym.opex2000.actions.project.
ProjectOpenAction
openThisProjContext=com.gensym.opex2000.project.Project
openThisProjIsPopup=true
closeThisProjLabel=Deactive project
closeThisProjAction=ProjectClose
closeThisProjImage=images/opproj.gif
closeThisProjShortCut=D

```

closeThisProjContext=com.gensym.opex2000.project.Project
closeThisProjActionClass=com.gensym.opex2000.actions.project.
ProjectCloseAction
closeThisProjIsPopup=true
#tglControls
view =viewHistory domainMap zoomIn zoomOut zoom2Fit zoom2FitAR - log
messages - toggleDesc togglePallets tglControls [X]toolbarDisplay
[X]statusDisplay [X]tglNavigatorMode - _desktopTabs
toolbarDisplayLabel=ToolBar
toolbarDisplayAction=ToggleToolBar
toolbarDisplayImage=images/tools.gif
toolbarDisplayShortcut=B
toolbarDisplayActionClass=com.gensym.cdggui.ui.actions.core.TglToolBar
statusDisplayLabel=Status Bar
statusDisplayAction=ToggleStatusBar
statusDisplayImage=images/statbari.gif
statusDisplayShortcut=S
statusDisplayActionClass=com.gensym.cdggui.ui.actions.core.TglStatusBar
desktopTabs=FILE::uitabs::desktopTabs
desktopTabsLabel=Desktop Tabs main
desktopTabsShortcut=k
desktopTabsImage=images/ltabs.gif
uiMode=FILE::g2UserModes::uiMode
uiModeLabel=Mode
uiModeShortcut=M
uiModeImage=images/mode.gif
window=FILE::windowMenu::window
windowLabel=Window
windowShortcut=w
logLabel=Logbook
logShortcut=L
logImage=images/lb.gif

```

logAction=ShowLogBook
logActionClass=com.gensym.cdggui.ui.actions.g2server.ShowLogBookAction
viewMessagesLabel=View Messages
viewMessagesTooltip=View the messages of the selected message browser
viewMessagesShortCut=s
viewMessagesImage=images/dlg.gif
viewMessagesAction=ViewMessages
viewMessagesContext=com.gensym.classes.modules.scroll.ScBrowserTemplate
viewMessagesActionClass=com.gensym.cdggui.ui.actions.messagebrowser.
ViewMessagesAction
viewMessagesActiveWhenPaused=false
viewMessagesGoNoGoClass=com.gensym.cdggui.ui.actions.messagebrowser.
ViewMessagesGoNoGo
viewMessagesIsPopup=true
messagesLabel=Message Board
messagesShortCut=M
messagesImage=images/mb.gif
messagesAction=ShowMessageBoard
messagesActionClass=com.gensym.cdggui.ui.actions.g2server.
ShowMessageBoardAction
messagesActiveWhenPaused=false
domainMapLabel=Domain Map
domainMapShortCut=D
domainMapImage=images/webLink.gif
domainMapAction=GetDomainMap
domainMapActionClass=com.gensym.cdggui.ui.actions.opex.
GetDomainMapAction
domainMapActiveWhenPaused=false
viewLabel=View
viewmage=images/view.gif
viewShortCut=V
viewPluginID=UI


```

diagramFolderShortCut=D
diagramFolderLabel=Diagram Folder
diagramFolderImage=images/dfldr.gif
diagramFolderAction=newDiagramFolder
diagramFolderActionClass=com.gensym.cdggui.ui.actions.symcure.
NewDiagramFolderAction
diagramFolderContext=com.gensym.classes.modules.cdg.CdgDiagramFolder
com.gensym.classes.KbWorkspace
diagramFolderIsPopup=true
diagramFolderPluginID=SYMCURE
newGenericEventShortCut=E
newGenericEventLabel=Event Node
newGenericEventAction=newFault
newGenericEventActionClass=com.gensym.cdggui.ui.actions.symcure.
NewFaultAction
newGenericEventContext=com.gensym.classes.modules.cdg.CdgDiagramFolder
com.gensym.classes.KbWorkspace
newGenericEventIsPopup=true
newNodeViewLabel=Node View
newNodeViewShortCut=V
newNodeViewAction=newNodeView
newNodeViewActionClass=com.gensym.cdggui.ui.actions.symcure.
NewViewAction
newNodeViewContext=com.gensym.classes.modules.cdg.CdgDiagramFolder
com.gensym.classes.KbWorkspace
newNodeViewIsPopup=true
explainContext=com.gensym.classes.modules.cdg.CdgSpecificEvent
newClassDefinitionLabel=Class
newClassDefinitionShortCut=C
newClassDefinitionImage=images/newclass.gif
#newClassDefinitionAction=newClass
#newClassDefinitionActionClass=com.gensym.cdggui.ui.actions.core.
NewClassDefinitionAction

```

```

# HELP MENU
help=about
helpLabel=Help
helpShortCut=H
helpPluginID=UI
aboutLabel=About
aboutShortCut=A
aboutAction=About
aboutActionClass=com.gensym.cdggui.ui.actions.core.AboutAction
aboutImage=images/ico1.jpg
# ITEM MENU
#
# The _ tells the system that the menu is a sub menu
#
# _specify - addToProj rmFromProj
item= viewObject props configureBlock - sender target - trueValue falseValue
runTest explorer - configuration detectability isolatability getSig - addToProj
rmFromProj
#item= viewObject props - sender target - trueValue falseValue runTest
explorer - configuration detectability isolatability getSig - descProj addToProj
rmFromProj
#status
goBackLabel=Go Back
goNextLabel=Go Next
goBackShortCut=B
goNextShortCut=N
goBackImage=images/back.gif
goNextImage=images/next.gif
goBackAction=back
goBackActionClass=com.gensym.cdggui.ui.actions.core.HistoryBackAction
goNextAction=next
goNextActionClass=com.gensym.cdggui.ui.actions.core.HistoryNextAction

```

```

itemLabel=Item
itemShortCut=I
itemPluginID=UI
wizards=newApplication diagramFolder newGenericEvent newNodeView
wizardsLabel=Wizards
newApplication=W
wizardsWhenPaused=false
wizardsImage=images/create.gif
# The () prefix tells the system to build radio buttons
# the one with (*) is the defdault selection
#
specify= ()trueValue (*)falseValue
specifyImage=images/ballcyan.gif
specifyLabel=Specify Value
specifyShortCut=S
specifyContext=com.gensym.classes.modules.cdg.CdgSpecificEvent
trueValueLabel=Specify True
trueValueShortCut=T
trueValueContext=com.gensym.classes.modules.cdg.CdgSpecificEvent
trueValueAction=SpecifyTrue
trueValueActiveWhenPaused=false
trueValueActionClass=com.gensym.cdggui.ui.actions.symcure.
SpecifyTrueAction
trueValueImage=images/ballred.gif
trueValueIsPopup=true
falseValueLabel=Specify False
falseValueShortCut=F
falseValueContext=com.gensym.classes.modules.cdg.CdgSpecificEvent
falseValueActiveWhenPaused=false
falseValueAction=SpecifyFalse

```

falseValueActionClass=com.gensym.cdggui.ui.actions.symcure.
SpecifyFalseAction
falseValueImage=images/ballgreen.gif
falseValueIsPopup=true
explorerLabel=Diagnosis Details
explorerShortCut=D
explorerActiveWhenPaused=false
explorerAction=UploadToNewBrowser
explorerTooltip=Get Diagnosis Details
explorerActionClass=com.gensym.cdggui.ui.actions.symcure.
UploadToNewBrowserAction
explorerImage=images/mtb.gif
explorerContext=com.gensym.classes.modules.cdg.CdgCorrelationManager
gotoLabel=Goto
gotoShortCut=G
gotoAction=GotoItem
gotoActionClass=com.gensym.cdggui.ui.actions.core.GotoItemAction
gotoImage=images/goto.gif
gotoActiveWhenPaused=false
gotoAdditionalSupportClass=com.gensym.cdggui.ui.indicators.core.
GotoItemSupport
gotoRequiresSelection=true
#means that if the selection is null, dont apply this option
gotoExclude=java.lang.String com.gensym.util.Symbol com.gensym.util.
Sequence com.gensym.util.Structure null.class
targetLabel= Goto Target
targetShortCut=T
targetAction=GotoTarget
targetActiveWhenPaused=false
targetImage=images/target.gif
targetActionClass=com.gensym.cdggui.ui.actions.opac.GotoTargetAction

```

targetContext=com.gensym.classes.modules.cdg.CdgSpecificFault com.gensym.
classes.modules.cdg.CdgSpecificOrPropagation com.gensym.classes.modules.
cdg.CdgSpecificAndPropagation com.gensym.classes.modules.smh.
SmhTransientMessage

targetIsPopup=true

senderLabel=Goto Sender

senderShortCut=S

senderImage=images/sender.gif

senderActiveWhenPaused=false

senderAction=GotoSender

senderActionClass=com.gensym.cdggui.ui.actions.opac.GotoSenderAction

senderContext=com.gensym.classes.modules.cdg.CdgSpecificFault com.gensym.
classes.modules.cdg.CdgSpecificOrPropagation com.gensym.classes.modules.
cdg.CdgSpecificAndPropagation com.gensym.classes.modules.smh.
SmhTransientMessage com.gensym.cdggui.ui.OpExInitializationsFrame

senderIsPopup=true

detectabilityLabel=Check Detectability

detectabilityShortCut=D

detectabilityActiveWhenPaused=false

detectabilityImage=images/detect.gif

detectabilityAction=Detectability

detectabilityContext=com.gensym.classes.modules.cdg.CdgGenericFault com.
gensym.classes.modules.cdg.CdgDiagramFolder

detectabilityActionClass=com.gensym.cdggui.ui.actions.symcure.
DetectabilityAction

detectabilityIsPopup=true

isolatabilityLabel=Check Isolatability

isolatabilityShortCut=I

isolatabilityAction=Isolatability

isolatabilityImage=images/iso.gif

isolatabilityActiveWhenPaused=false

isolatabilityActionClass=com.gensym.cdggui.ui.actions.symcure.
IsolatabilityAction

isolatabilityContext=com.gensym.classes.modules.cdg.CdgDiagramFolder

```

```

isolatabilityIsPopup=true
configurationLabel=Check Configuration
configurationShortCut=C
configurationActiveWhenPaused=false
configurationImage=images/check.gif
configurationAction=Configuration
configurationActionClass=com.gensym.cdggui.ui.actions.symcure.
ConfigurationAction
configurationGoNoGoClass=com.gensym.cdggui.ui.actions.symcure.
CheckConfigGoNoGo
configurationIsPopup=true
configurationExclude=java.lang.String com.gensym.util.Symbol com.gensym.
util.Sequence com.gensym.util.Structure com.gensym.classes.modules.cdg.
CdgSpecificFault com.gensym.classes.modules.cdg.CdgCorrelationManager
com.gensym.classes.modules.cdg.CdgSpecificOrPropagation com.gensym.
classes.modules.cdg.CdgSpecificAndPropagation com.gensym.classes.
KbWorkspace
configurationContext= com.gensym.classes.modules.cdg.CdgDiagramFolder
com.gensym.classes.modules.cdg.CdgGenericFault com.gensym.classes.modules.
cdg.CdgGenericSymptom com.gensym.classes.modules.cdg.CdgGenericTest
//SYMCURE SPRCIFICS
//EOF
# MENU
tools= find clean - _session - relink initializations - properties
#tools= find clean _opex - properties
#- showInternals
systemLabel=System Tables
systemShortCut=T
systemImage=images/ide.gif
#_connection
session= _uiMode _g2
sessionLabel=Session Details
sessionShortCut=D
pallets= addgroup addpalette

```

```
palletsLabel=Palettes
palletsShortCut=P
propertiesLabel=Default Properties
propertiesImage=images/pallets.gif
propertiesAction=UIProperties
propertiesShortCut=D
propertiesActionClass=com.gensym.cdggui.ui.actions.core.UIPropertiesAction
MIBBrowserLabel=MIB Browser
MIBBrowserImage=images/pallets.gif
MIBBrowserAction=ShowMibBrowser
MIBBrowserShortCut=M
MIBBrowserActionClass=com.gensym.opex2000.actions.adventnet.
ShowMibBrowserAction
discovery=StartAutoDiscovery StopAutoDiscovery StatAutoDiscovery
PropertiesOfDiscovery
discoveryLabel=Discovery
discoveryShortCut=D
discoveryPluginId=AUTODISCOVERY
showWanLabel=WAN
showWanTooltip=W
showWanTooltip=Show WAN Schematic
showWanAction=ShowWAN
showWanImage=images/wan.gif
showWanActionClass=com.gensym.opex2000.actions.nm.ShowWANAction
showWanPluginId=AUTODISCOVERY
#showWanGoNoGO=
showWanUserModeContext= ADMINISTRATOR DEVELOPER
StartAutoDiscoveryLabel=Start Auto Discovery
StartAutoDiscoveryImage=images/run.gif
StartAutoDiscoveryTooltip=Start Auto Discovery
StartAutoDiscoveryShortCut=S
```

StartAutoDiscoveryAction= StartDiscovery
StartAutoDiscoveryActionClass=com.gensym.opex2000.actions.nm.
StartDiscoveryAction
#StartAutoDiscoveryGoNoGO=
StartAutoDiscoveryUserModeContext= ADMINISTRATOR DEVELOPER
StopAutoDiscoveryLabel=Stop Auto Discovery
StopAutoDiscoveryImage=images/stop.gif
StopAutoDiscoveryTooltip=Stop Auto Discovery
StopAutoDiscoveryShortCut=T
StopAutoDiscoveryAction= StopDiscovery
StopAutoDiscoveryActionClass=com.gensym.opex2000.actions.nm.
StopDiscoveryAction
#StopAutoDiscoveryGoNoGO=
StopAutoDiscoveryUserModeContext= ADMINISTRATOR DEVELOPER
StatAutoDiscoveryLabel=Auto Discovery Status
StatAutoDiscoveryImage=images/wanstat.gif
StatAutoDiscoveryTooltip=Auto Discovery Status
StatAutoDiscoveryShortCut=u
StatAutoDiscoveryAction=DiscoveryStatus
StatAutoDiscoveryActionClass=com.gensym.opex2000.actions.nm.
DiscoveryStatusAction
#StopAutoDiscoveryGoNoGO=
StatAutoDiscoveryUserModeContext= ADMINISTRATOR DEVELOPER
PropertiesOfDiscoveryLabel=Properties
PropertiesOfDiscoveryImage=images/properties.gif
PropertiesOfDiscoveryTooltip=See Properties
PropertiesOfDiscoveryShortCut=p
PropertiesOfDiscoveryAction=DiscoveryProperties
PropertiesOfDiscoveryActionClass=com.gensym.opex2000.actions.nm.
DiscoveryPropertiesAction
#PropertiesOfDiscoveryGoNoGO=
PropertiesOfDiscoveryUserModeContext= ADMINISTRATOR DEVELOPER


```
configureLabel=Configure Palettes
configureImage=images/pallets.gif
configureAction=ConfigurePalettes
configureActionClass=com.gensym.cdggui.ui.actions.core.
ConfigurePalettesAction
addgroupLabel=Add Palette Group
addgroupAction=AddPaletteGroup
addgroupActionClass=com.gensym.cdggui.ui.actions.core.
AddPaletteGroupAction
addpaletteLabel=Add Palette
addpaletteAction=AddPalette
addpaletteActionClass=com.gensym.cdggui.ui.actions.core.AddPaletteAction
controlLabel=Server Control
controlShortCut=S
controlAction=G2ServerControl
controlImage=images/led.gif
controlUserModeContext= ADMINISTRATOR DEVELOPER
controlActionClass=com.gensym.cdggui.ui.actions.g2server.
G2ServerControlAction
showInternalsLabel=Details
showInternalsShortCut=D
showInternalsAction=ShowInternals
showInternalsImage=images/led.gif
toolsLabel=Tools
toolsShortCut=T
toolsPluginID=UI
tglNavigatorModeLabel=Navigator Mode
tglNavigatorModeShortCut=g
tglNavigatorModeAction=ToggleNavigatorMode
tglNavigatorModeImage=images/nav.gif
tglNavigatorModeActionClass=com.gensym.cdggui.ui.actions.core.TglNavMode
toggleDescLabel=Toggle Description
```

toggleDescShortCut=p
toggleDescAction=ToggleDesc
toggleDescImage=images/a123.gif
toggleDescActionClass=com.gensym.cdggui.ui.actions.core.TglDescAction
togglePalletsLabel=Toggle Controls
togglePalletsShortCut=N
togglePalletsAction=TogglePallets
togglePalletsImage=images/a124.gif
togglePalletsActionClass=com.gensym.cdggui.ui.actions.core.TglPalletsAction
findLabel=Find
findShortCut=F
findAction=Find
findImage=images/find.gif
findActionClass=com.gensym.cdggui.ui.actions.core.FindAction
notepadLabel=Notepad
notepadShortCut=N
notepadAction=Notepad
notepadImage=images/np.gif
notepadActionClass=com.gensym.cdggui.ui.NotepadAction
cleanLabel=Clean Up Model
cleanShortCut=C
cleanAction=Clean
cleanImage=images/clear.gif
cleanActionClass=com.gensym.cdggui.ui.actions.symcure.CleanAction
cleanContext=com.gensym.classes.modules.cdg.CdgDiagramFolder
cleanIsPopup=true
optionsLabel=IDE Options
optionsShortCut=I
optionsAction=IDE
optionsImage=images/tools.gif
optionsActionClass=com.gensym.cdggui.ui.actions.core.IDEOptionsAction

```

#
# edit Menu definition
edit=cut copy paste delete
editLabel=Edit
editShortCut=E
cutLabel=Cut
cutAction=cut-to-clipboard
cutImage=images/cut.gif
cutShortCut=T
copyLabel=Copy
copyAction=copy-to-clipboard
copyImage=images/copy.gif
copyShortCut=C
pasteLabel=Paste
pasteAction=paste-from-clipboard
pasteImage=images/paste.gif
pasteShortCut=P
deleteLabel=Delete
deleteAction=Undo
deleteShortCut=t
deleteAction=Delete
deleteActionClass=com.gensym.cdggui.ui.DeleteAction
deleteImage=images/delete.gif
g2Label=Servertt
g2ShortCut=S
g2Image=images/sd.gif
g2=FILE::g2Control::g2
g2PluginID=G2
#####
## PALLETS                                     ##
#####

```

opacPalletTabLabel=OPAC
opacPallet =FILE::opacPalette::opacPallet
opacPalletPluginID=OPAC
telecomsPalletTabLabel=Network
telecomsPallet=FILE::telcoPalette::telecomsPallet
telecomsPalletPluginID=OPEX
symcurePallet=FILE::symcurePalette::symcurePallet
symcurePalletTabLabel=SymCure
symcurePalletPluginID=SYMCURE
opexObjects=FILE::opexPalette::opexObjects
opexObjectsTabLabel=OpEx
opexObjectsPluginID=OPEX
g2SystemPallet=FILE::g2corePalette::g2SystemPallet
g2SystemPalletTabLabel=Core G2 Objects
g2SystemPalletPluginID=G2
newTooltip=Create a new file
openTooltip=Open folder

Getting Started

Describes how to start building Integrity applications, the objects created in a new application, and the Integrity menu system; also how to interact with G2 objects. It also provides a tutorial on how to build a sample application.

Introduction	131
Creating a New Application	132
Using the Integrity Setup Dialog	133
Building a Simple Domain Map	139
Working with G2 Objects	144
Other Integrity Modules	144
Adding Integrity Functionality to an Existing Application	145
Out-of-Box Functionality	145



Introduction

To create an Integrity application, you must first install G2, the Integrity Module, and any other modules you have purchased. See the installation sections in the “Overview” chapter of this manual for information regarding the installation of these products.

This chapter provides a tutorial that leads you through the creation of a new Integrity application. The Integrity menu system and the application objects created in a new application are described at the end of this chapter.

Creating a New Application

Before creating a new application, an understanding of modular structure is required. Ensure that consistent modularization within an application is maintained. For more information on consistent modularization and module hierarchies, refer to the section “Creating a Module Hierarchy” in the *G2 Reference Manual*.

The top-level module is the module that provides all the functionality for running the Integrity UI. This module includes all of the necessary modules to develop and run your application

Package	Top-Level Module
Integrity package	<i>integrity.kb</i>

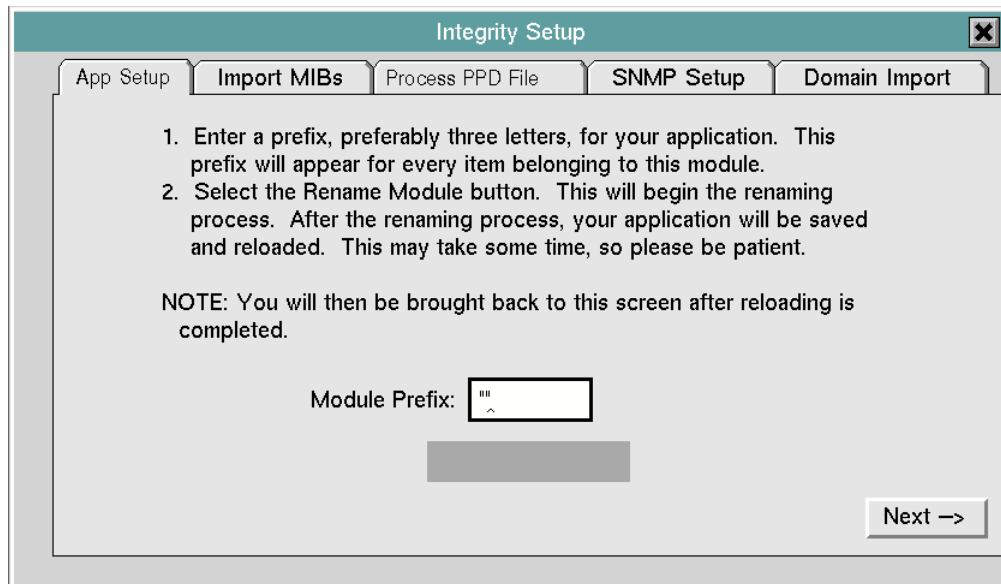
To create a new application:

- 1 Choose File > New.
All libraries currently installed have a check mark next to the library name.
- 2 Review the checked libraries and ensure the Integrity and SNMP Interfaces libraries are selected; deselect any other library you do not wish to be included in your application.
- 3 Enter a name for your application in the Project Name text box.
- 4 Click OK.

Once the new application has been created and saved, it is loaded into the G2 server. You are now ready to start building your application domain objects and other items.

Using the Integrity Setup Dialog

The Integrity Setup Dialog helps you to get your application up and running. To access the Integrity Setup dialog, first switch to Developer mode, then click the Setup button in the Integrity toolbar. Here is the Setup dialog:

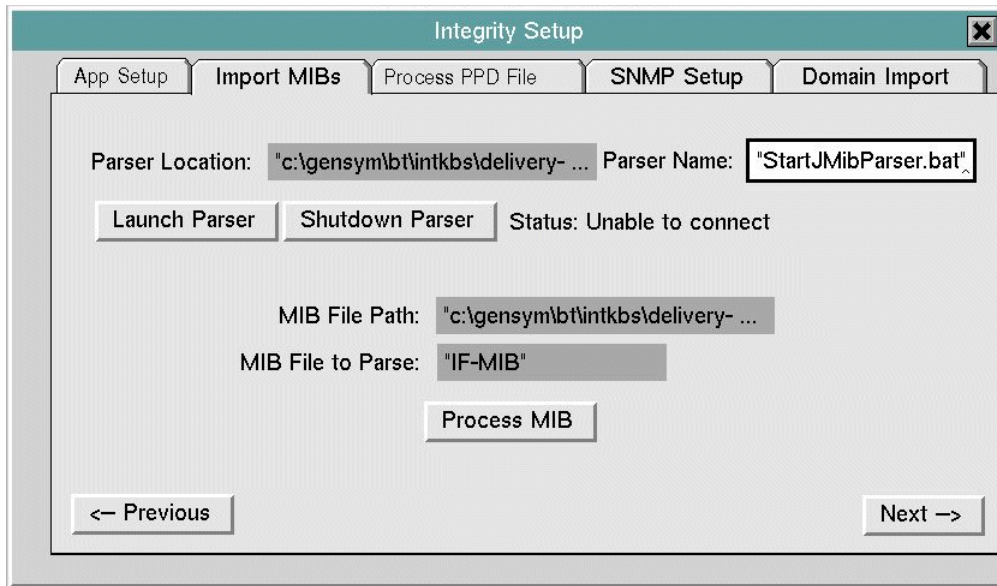


The Setup dialog helps you set up these aspects of your application: naming your application, importing MIBs, optionally processing the PPD file, starting the necessary SNMP bridges, and importing a domain from an SQL-compliant database. The sections below describe these four steps.

Importing Management Information Base (MIBs)

The Import MIBs feature allows you to import MIBs of the equipment you want to monitor and manage. The importer is based on the AdventNet libraries and requires the g2MibParser bridge to be running. If the MIB you are importing requires other MIB files, the Import MIBs feature imports those as well. Note that

the supporting MIB files must be located in the same directory as the imported MIB file. Here is the Import MIBs tab page:



Before processing the MIB file, the wizard must start the g2MibParser bridge. The Parser value in the dialog should already display the correct path for the MIB Parser batch file.

To launch the MIB parser bridge, click the Launch Parser button. When the bridge has been launched, the status message indicates that the bridge is connected.

To parse MIB files, specify the MIB File Path location and the MIB File to Parse by clicking the "..." button to the right of the File text box. This will launch a file selection dialog allowing you to navigate to the MIB file. Click the Process MIB button to process the selected MIB file. Repeat this process for each MIB file that you need to parse.

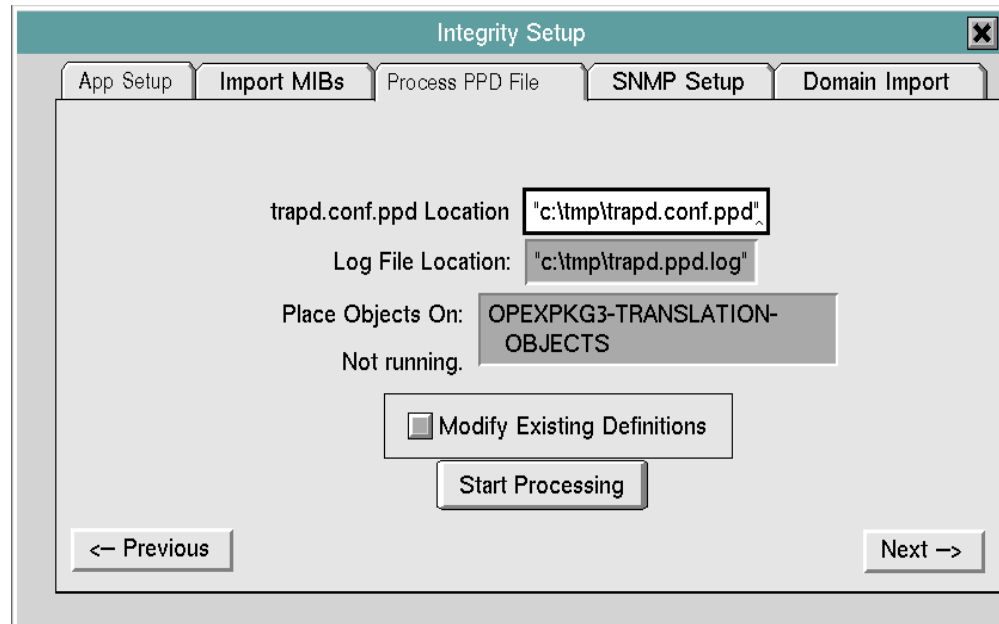
Note The MIB File Path must be specified as an absolute path.

The wizard places the parsed information on a subworkspace of a gmib-mib-reader object located on the Application Objects > Translation Objects workspace. If you need to parse this file again, you can repeat the above process or you can choose gmib parse mib on the gmib-mib-reader object.

Once you have processed all MIB files, click the Process PPD File tab.

Process PPD File

For HP OpenView users, you can optionally process the *trapd.conf* file to produce the *trapd.conf.ppd* file. This file contains trap information and message format specifications. If you do not use HP OpenView, you can also use this feature to create the file by hand. Consult the SNMP User's Guide for details. This figure shows the Process PPD File tab page:



To process the *trapd.conf.ppd* file, specify the name and location of the file. If the file has already been processed, enable the Modify Existing Definitions option to modify definitions instead of creating new ones. To process the file, click the Process PPD File button.

Note The wizard deletes all existing trap instances of the object before creating new object definitions.

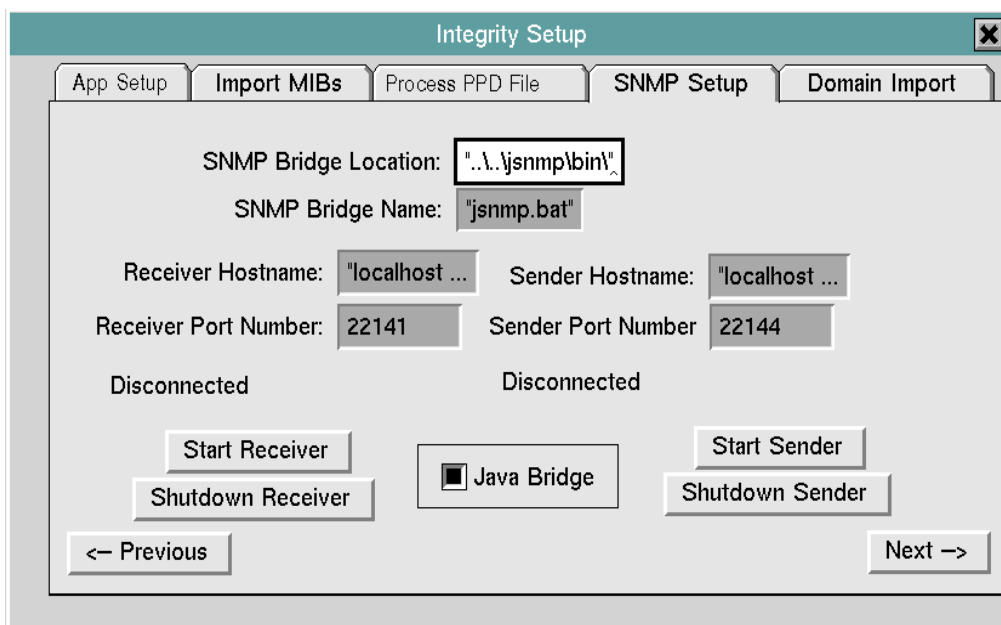
Once the *trapd.conf.ppd* file is processed, click the SNMP Setup tab.

SNMP Setup

The SNMP Setup feature allows you to start both the Sender SNMP bridge and the Receiver SNMP bridge. You can start and shutdown each bridge process separately.

Note If the bridge is installed on a remote machine, then you must start the bridge and connect manually. The GSI variables for starting and connecting the bridge are located in the Navigator under System Settings > Interfaces > SNMP.

Here is the SNMP Setup tab page:



The SNMP Bridge Location text box should already have the complete path and startup batch file; however, if this needs to be changed, click the "..." button to select the *StartJsnmpBridge.bat* file.

Specify the Receiver and Sender Hostname and Port Number to be the host name and port number for each bridge process.

Note If you are using the Java SNMP Generic Bridge, enable the Java Bridge option to enable the automatic processing of trap information. For more information, see [Out-of-Box Functionality](#).

To start the SNMP Receiver bridge, click the Start Receiver button. To start the SNMP Sender bridge, click the Start Sender button. The bridge status appears above each button. To shut down either bridge, click the appropriate Shutdown button.

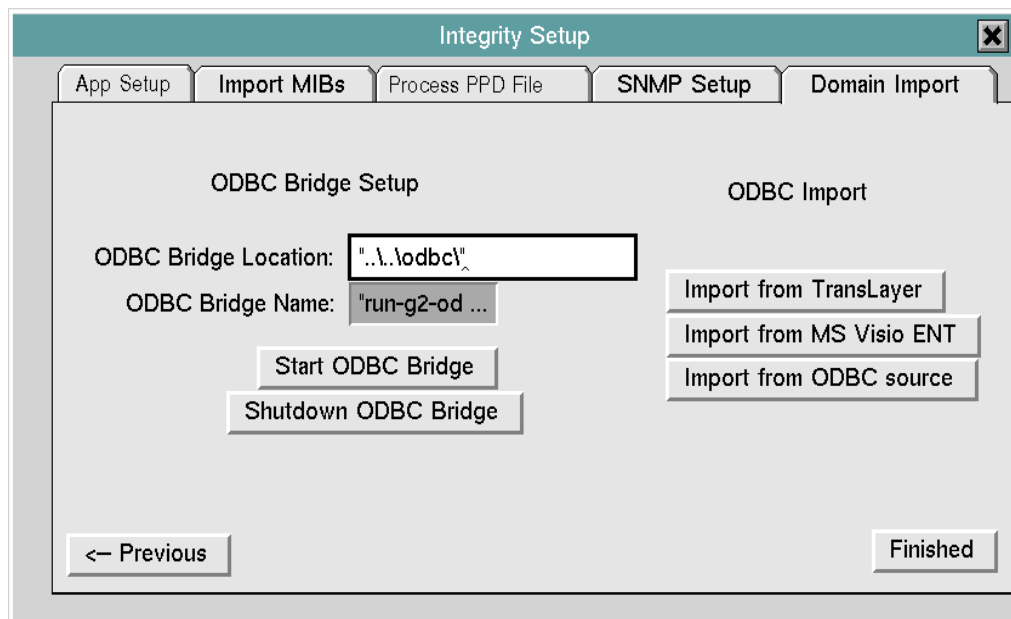
Once the bridges are running and connected, click the Domain Import tab.

Domain Import

The Domain Import feature uses the G2-ODBC Bridge to import domain information from an SQL-compliant database. You have three options for importing domain information: Regenative's Translayer, Microsoft's Visio Enterprise Network Tools, and an ODBC database. For information about the required format of the ODBC database, see the DXI3DB chapter in the *Integrity User's Guide*. Another option of importing your domain is through the import of an HP OpenView saved exported file. If the `ovet_topoquery getAllNodes -ShowIF` command was also used to export HP OpenView's Layer 2 information, this file can also be imported.

Note If the bridge is installed on a remote machine, then you must start the bridge and connect manually.

The Domain Import tab is shown below:



The Domain Import tab is divided into two sections: ODBC Setup and Import and HPOV Setup and Import.

ODBC Setup and Import

Before you can import the domain information, you must create an ODBC data source for the database to import. Refer to Microsoft Help for information on how to create an ODBC data source.

Once the ODBC data source exists, you must start the G2-ODBC Bridge. Specify the ODBC Bridge Location and ODBC Bridge Name to refer to the directory location of the ODBC bridge executable, which is named *run-g2-odbc.bat* and is located in the *odbc\bin* directory of the G2 installation directory. Be sure to specify the ending separator in the path name, for example, *C:\Program Files\Gensym\g2-2011\odbc\bin*.

Click the Start ODBC Bridge button to start the bridge. To shut down the bridge, click the Shutdown ODBC Bridge button.

Import from Translayer

This feature allows you to import from an ODBC data source that was populated by Regenative's Translayer product. Translayer's auto-discovery discovers network items, WMI information, and Windows user accounts. All of this information is imported into Integrity.

Import from MS Visio ENT

This feature allows you to import from an ODBC data source that was populated by Microsoft's Visio Enterprise Network Tools product. Visio ENT's auto-discovery discovers network items and WMI information. All of this information is imported into Integrity.

Importing from other ODBC Sources

Allows you to import from an ODBC data source populated by means other than Translayer or Microsoft Visio.

Configuration of the ODBC Import

Specify the Bridge Host and Bridge Port of the G2-ODBC Bridge process. Specify the Username and Password for accessing the ODBC data source. Specify the Data Source to be the name of the ODBC data source you created through Windows.

To connect to the ODBC bridge and import the data, click the Start Domain Import button. When Integrity connects to the bridge, a workspace appears showing readout tables of the types of information being imported. The instances that are created are placed in a repository object.

HPOV Setup and Import

This section allows you to import a domain map based on an export of an HP OpenView map. If the Layer 2 information was also exported, this information can also be imported and used by Integrity. To be compatible with the Layer 2 import facility of Integrity, be sure the *ovet_topoquery* command uses the *getAllNodes* and *-ShowIF* command arguments.

Here is a description of the information required by HPOV Setup and Import:

- **Translations Location** – Specifies the location of translation objects used to translate between the HPOV specified fields and Integrity objects. You should not have to create additional translation objects for this section because Integrity already provides these translations.
- **Default Class To Create** – If no translation object exists for the imported object, an instance of this class will be created.
- **Domain Map Destination** – Specifies the destination of the imported objects.
- **New Class Destination** – Specifies the location of new classes created by the import process.
- **Export Retrieve Procedure** – Specifies the retrieve procedure to be called to generate the HPOV export file. This is only used when running the G2 server on the same machine as HPOV.
- **Export Retrieve Command** – Specifies the export command to issue to generate an export file from HPOV. This is only used when running the G2 server on the same machine as HPOV.
- **Exported File Already Exists** – If selected, the Export Retrieve Procedure and Command will not be executed and will import the file specified in the Export File Location.
- **Export File Location** – Specifies the direct path of the export HPOV file.
- **Import Layer 2** – If selected, the Layer 2 File Location is processed to import HPOV Layer 2 information.
- **Layer 2 File Location** – Specifies the exported Layer 2 file. This file must be generated by the following command: `ovet_topoquery getAllNodes -ShowIF`.

To start the import, click the Start HPOV Domain Import button. During the import, a progress notification dialog is displayed and is dismissed when the import process has completed.

Building a Simple Domain Map

The first step to build an Integrity application is to build a domain map. You can obtain a domain map for use in Integrity by:

- Building the domain map manually.
- Import an existing domain map, for example, from HP OpenView.

- Import an SQL-compliant database that contains your domain objects, including:
 - A pre-built database.
 - Auto-discovery results from a TransLayer session.
 - Auto-discovery results from a Microsoft Visio Enterprise Network Tools session.

Building a domain map manually helps you to understand how Integrity works with the domain map. In this tutorial you will build the domain map manually by:

- 1 Creating subclasses from the Integrity Foundation Classes.
- 2 Creating instances of those classes.
- 3 Placing the instances on a domain map.

Creating Domain Map Subclasses

Domain objects are created from subclasses of the Integrity Foundation Classes `opfo-managed-object` and `opfo-containment-object`, both of which are subclasses of `opfo-domain-object`.

- A **containment object** is an object whose sole purpose is to contain other objects on its subworkspace. Examples of typical containment objects are states, cities, and rooms.
- A **managed object** is an object that represents an external object managed by the application. Examples of managed objects are routers, computers, and sensors. Managed objects can also contain other objects but it is not their sole purpose.

In this example, you must create a subclass:

- `test-computer` - used to create the domain objects that represent the external objects you are managing in the application. This is a subclass of `opfo-managed-object`.

First, create the `TEST-TOP-NODE` subclass.

To create the subclass:

- 1 Display the Toolbox - G2 palette by selecting View > Toolbox G2.
- 2 Select the Definitions and Relations palette and select Class Definition. Move your mouse and drop it on a new workspace you have created.

- 3 Right-click on the new object and select **table** (or double-click on the new object) to display the table.
- 4 Enter the class name for the new object subclass, `TEST-COMPUTER`, then enter `opfo-managed-object` for the Direct Superior Classes.

Creating Domain Objects

Now that you have defined the application subclass, you are ready to create instances and place them on the domain map.

To create a domain map container:

- 1 From the Navigator, select System Models. Right-click on Network Diagrams and select “New Instance...”. This will create an instance of the `gndo-network-topology` class
- 2 When you create a new instance, the properties dialog is displayed for the new object.

a test-containment-object	
Opfo external name	""
_opfo highest message priority	99999
_opfo acknowledgement status	unacknowledged

- 3 Enter a name of `a-top-node` for the Domain Object Name and select the OK button. After pressing the OK button, you will notice in the Navigator the addition and new name of the `gndo-network-topology` object.
- 4 From the Navigator right-click on `a-top-node` and select Show Details. this is the subworkspace of `a-top-node`.

You defined the subclass `test-computer` to create instances of managed objects. At this point you can now create instances of your `test-computer` class and place them on the `a-top-node` subworkspace.

Working with Modules

G2 allows you to organize your knowledge base into high-level units called **modules**. To gain a thorough understanding of modules and how they are used in G2, refer to the *G2 Reference Manual*. The menu option Modules provides facilities for working with G2 modules.

Creating Modules

To create a new module in an application

- 1 Choose File > KB Modules > New.
- 2 Type in the name of the new module then click OK.

Merging Modules

To merge a module into an application:

- 1 Choose File > KB Modules > Merge.
- 2 Locate the module by using the file selection button "... " to the right of the File Name text box.
- 3 Select any of the following options:
 - Resolve Conflicts Automatically - When this option is selected, the merging routine automatically checks the existing application for naming conflicts which can occur as a result of the objects coming into the application from the new module. It is best to select this option.
 - Bring Formats up to Date - This option applies the formats of the current version of G2 to all objects merged into the application. It is generally not recommended that you select this option unless you want to mix items developed under different G2 versions.
 - Install System Tables of Merged KB - When you want to bring a module into an application and designate it as the top-level module of the application you should select this option.

Renaming Modules

To rename a module:

- 1 Choose File > KB Modules > Rename.
- 2 Enter a name for the module in the New Name text box.
- 3 Select a module from the Module To Rename list and click OK.

Saving Modules

To save an individual module:

- 1 Choose File > KB Modules > Save.
- 2 Enter or select the File name, and select the Module to save.

- 3 Select any of the following options from the bottom of the file selection dialog:
 - Including all required modules – Saves all modules required by the module you selected above.
 - Save all modules to one file – Saves all modules regardless of the selected module above into a single file.

Deleting Modules

To delete a module from an application:

- 1 Choose File > KB Modules > Delete.
- 2 Select the name of the module you want to delete from the dialog window.
- 3 Select any of the following options:
 - Delete Associated Workspaces – This option deletes all workspaces associated with the module you choose to delete. This option is usually selected.
 - Remove References to Module in Hierarchy – This option removes all references to the deleted module in other modules in the module hierarchy. If deleting a module leaves another module that is not required by any other modules, a warning message is posted.

G2 Mode

G2 allows you to declare distinct categories of usage, called **user modes**, for your application. Each user mode represents a style of interaction with the application's knowledge. The meaning of each style depends on how your application organizes its knowledge and the user interface to it.

Five modes are defined on the Tools > User Mode menu item:

- Administrator
- System-Administrator
- Developer
- Modeler
- Operator

Administrator and System-Administrator modes offer the least restrictions on what you can do in the application. Each mode becomes successively more restrictive. In general, you use Developer mode when you are developing your application. In this mode, all buttons function as buttons. In Administrator mode, when you click on a button you view its item menu. So, when you want to modify or define behaviors of buttons, you will need to switch to Administrator mode.

The behavior of Operator mode is defined by the application. The behavior of the different modes and the details of item and instance configuration is described in the *G2 Reference Manual*.

Working with G2 Objects

G2 provides a flexible programming environment. You can customize the appearance and user interface of the all the workspaces and items you create in your application. For a complete description of the G2 environment, see the *G2 Reference Manual*.

Other Integrity Modules

When you create a new application, these additional modules are included in your application.

These modules are:

- Integrity family
 - *symcure.kb* — The Integrity SymCure Reasoning Module, which is a graphical environment for building causal directed graph models for reasoning about events. This module is included in the Reasoner and Premium Integrity bundles.
 - *gsnmp.kb* — The Integrity SNMP Module, which is an internal bridge KB for use with the SNMP bridges.
- Demonstration knowledge bases, which demonstrate Integrity functions. It can be useful to include demo knowledge bases in an application because functionality from a demo can be used as the basis of similar functionality in your application. When you just want to view the demonstration systems, it is better to load them as the main application as described in [Building a Simple Domain Map](#).

Demonstration knowledge bases include:

- *opx_demo.kb* demonstrates features of Integrity.
 - *dxi_demo.kb* demonstrates features of the domain map importer.
 - *opac_demo.kb* demonstrates features of the OPAC module.
 - *svcmdemo.kb* provides an example of using SymCure as a basis of Service Level Agreement application.
- Application libraries - The modular structure of Integrity lets you design applications so that functions can be encapsulated in modules and reused in other applications. You can create your own library of modules, which you can merge into applications as needed.

When you create a new application, select the libraries you want to include from the dialog. For more information about the module structure of G2, refer to the *G2 Reference Manual*.

Note If all the modules required by your Integrity application do not reside in a single directory, then you must create a module search path or module map as described in the *G2 Reference Manual*.

Adding Integrity Functionality to an Existing Application

If you are adding Integrity functionality to an existing G2 application, you can take two approaches depending on the needs of your application:

- Create a new Integrity application as described in the Introduction and merge in your application as one of the required modules of the new application. This is the recommended procedure.
- Load your existing application, then merge in the Integrity module, *integrity.kb*. If you take this approach, you have to define the dependencies between your top-level module and the new modules you have merged. For information on module dependencies, refer to the *G2 Reference Manual*.

Out-of-Box Functionality

Integrity provides a limited set of functionality without any configuration or setup by the user. This functionality is provided to allow you to start using the features of Integrity right after installation. The out-of-the-box functionality includes auto-clearing and time-based events.

Auto-Clearing

The auto-clearing feature allows one trap to clear another trap. Auto-clearing traps are provided for LinkDown and LinkUp events. When Integrity receives a trap, it searches based on the `the-gmib-clears-for` relation. This relation is created when the G2 MIB Parser reads in MIB information. This relation can be used to relate any two `gmib-trap-properties` together. When the method that searches for the LinkUp and LinkDown `gmib-trap-properties` runs, it also looks for a procedure named by the symbolic parameter `gmib-clears-for-procedure`. This allows you to customize how any two `gmib-trap-properties` objects are related.

Time-Based Events

Time-based events uses ODiE to create time-based events. Integrity creates an ODiE subscriber object that listens for events. When the subscriber receives an `snmp-event`, it calls the OPAC procedure named `pps-time-based-threshold-filter`, which checks the event history to see if it matches the time and count threshold set by the OPAC procedure. By default, the time threshold is 5 minutes, and the count threshold is 3, which means if the same `snmp-event` is received more than three times within a five minute period, an SMH message is created.

For information on ODiE, see the *Integrity Utilities Guide*.

Handling Events

Describes how you interface an Integrity application to a set of external objects and how the Integrity application handles events.

Introduction **147**

Setting up an External Interface **148**

Processing Unsolicited Events in the External Bridge **150**

Interpreting the Event in the Internal Bridge **152**

Automatic Trap Processing **154**



Introduction

The previous chapter showed how you construct a domain map to represent your external objects. The objects in the domain map can represent actual physical objects, software processes, databases or any other collection of items linked together to form a system. An Integrity application gathers information about the objects in the domain map by:

- Receiving unsolicited information directly from the external objects, their agents, or from a manager process linked to the objects.
- Polling the status of the external objects.

This chapter describes how information relating to the external objects enters Integrity, how the information is related to the domain map, and how procedures are called to take appropriate action for a particular type of event.

Setting up an External Interface

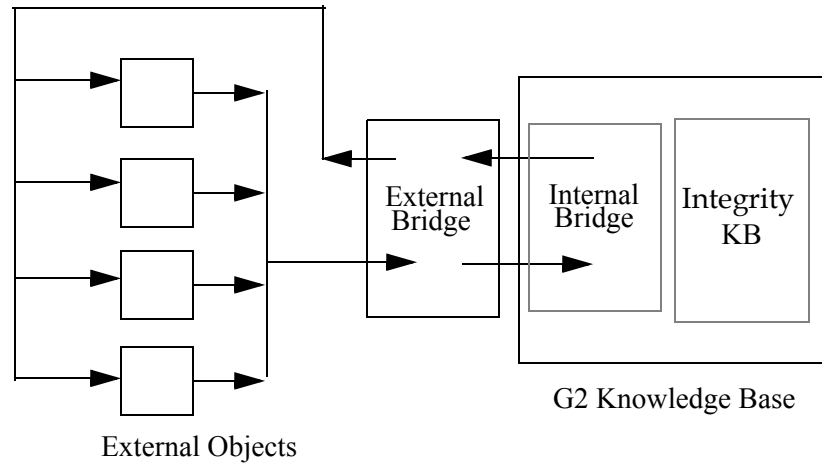
An event is a stream of data that contains information regarding the status of some external object. Events are either unsolicited or solicited. A **solicited event** is one that occurs in response to a request for information. An **unsolicited event** is one that is directly generated by the external object or an agent for that object to notify a management process of a change in the status of the object. The object originating an event is called the **sender**. The object an event is about is called the **target**. In some cases, the sender of the event is also the target of the event, but this is not always true. For example, one object can be configured to send information about objects contained within it, or a device can send a message about another device after failing in an attempt to communicate with the device. The description of the type of event from among the possible events the sender can generate is called the **category**. The syntax and codes used to represent these events depends on the protocols adopted by the type of object sending the event.

The application can receive a solicited or unsolicited event directly from an external object or through a manager layered between the external object and the application.

You interface Integrity to external objects, processes, and systems, using software called a **bridge**. You create a bridge using the G2 Gateway. Standard bridges exist for many devices. The G2-SNMP Bridge is designed to interface Integrity to systems that use the SNMP protocol.

Each bridge contains an external part, usually written in C or C++, and an internal part that resides within the Integrity application. The internal and external parts of the bridge communicate by means of a **remote procedure call (RPC)**. When a bridge process receives incoming information, the external bridge calls the internal RPC to place the information into the Integrity application. When a bridge process sends requests, the internal RPC calls a procedure in the external part of the bridge and passes it the data forming the request. You can define more than one bridge process to run at the same time.

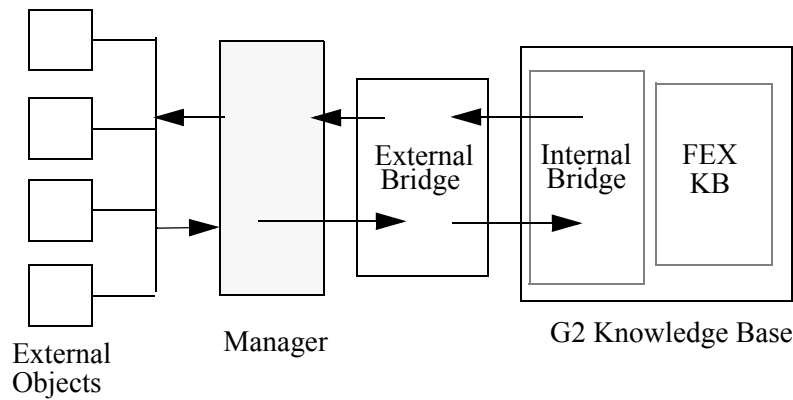
The figure below shows a generalized view of an Integrity application interfaced directly to a group of external objects:



The basic components of an Integrity application include:

- A group of external objects.
- An external bridge to move information to and from the objects to the Integrity application.
- An internal bridge to complete processing of the incoming information and relate the events to the Integrity domain objects.
- An Integrity knowledge base that contains:
 - domain objects related to the external objects
 - a message information base of past events
 - knowledge allowing you to reason about the events based on the histories and the relationships among the domain objects

In addition to these basic components, other components, such as a manager, can be layered between the external objects and the external bridge, as shown in the diagram below:



No assumptions are made as to the physical locations of any of these components. For example, the external objects can be software processes, and the entire system can be running on a single machine, or the external objects can be individual pieces of equipment managed by a central computer. The system configuration is entirely application-dependent and does not effect the basic operation of the application.

The details of how to construct a G2 Bridge are beyond the scope of this document. For information on how to construct a bridge and hook it up to external objects, refer to the *GSI Bridge Developer's Guide*. Users of the G2 SNMP Bridge should refer to the *G2-SNMP Installation and Operation Guide*. The rest of this chapter describes the functionality that you incorporate into your bridge to receive unsolicited events from a group of external objects and how to relate these events to the domain objects in the Integrity application.

Processing Unsolicited Events in the External Bridge

The external bridge process receives unsolicited events from the external objects, their agents, or an event manager, and should perform the following tasks:

- Parse and decode the events to determine the sender, target and category of each event.
- Apply low-level filtering to incoming events.
- Pass the parsed and decoded events along to the internal bridge via a remote procedure call (RPC).

This describes an ideal situation. Keeping as much of the parsing as possible in the external bridge maximizes the efficiency of the system. Unfortunately, in some cases, the complexity of the events passed can not make this feasible. This complexity can result from complicated protocols used by some types of object or from an application that models diverse objects, each using widely varying event protocols.

Parsing in the External Bridge

To extract the sender, target, and category for an event you need to:

- Parse the information that forms the event into the individual parameters the event defines. This includes translating coded fields in the event into meaningful text.
- Decode the names passed for the **sender** and the **target** into the external names used for the domain objects. The external names are stored as the value of the domain object attribute `Opfo-external-name`. Ideally, you can use external names that directly map to the sender and target parameters passed with the event.
- Determine a **category** name for the event. The **category** name must be chosen so it is easily parsed from the event and uniquely identifies the particular type of event or a grouping of similar events.

When the complexity of the system makes it too difficult to determine all of this information at a low level, the call to the RPC passes as much information as possible to the internal bridge.

Once the event is in the internal bridge, you can use the high-level functionality of G2 to perform other parsing and decoding tasks. Parsing and decoding that is highly individualized to a particular class of object can be done in routines related specifically to that one object class as described in [Defining Completion Routines](#).

When deciding whether to parse and decode in the external bridge or within the application itself, you must take into account the demands of the volume of incoming messages and the complexity of the parsing required.

Performing Low-Level Filtering in the Bridge

Low-level filtering does not depend on any previous events that have occurred or on any of the relationships in the domain map. It is filtering purely based on eliminating events that are sent from certain objects or are about certain targets or categories of events. When you decide to filter an event in the bridge, you can decide to simply discard the event and have it move no further along in the system, or you might decide to log the event to a text file.

Doing some filtering in the external bridge is particularly important in systems with a large volume of incoming messages. This filtering can significantly improve the throughput and efficiency of the system.

Interpreting the Event in the Internal Bridge

The call you make to the RPC from the external bridge brings the information contained in the event into the Integrity environment. The form of the information you pass as arguments to the RPC will depend on how much parsing and decoding you could complete in the external bridge. In the case where all parsing and decoding is done externally, the call to the RPC can specifically pass the sender, target, and category associated with the event along with any other informational arguments defined as part of the event.

The job of the internal bridge is to:

- Attempt to complete any parsing and decoding not done in the external bridge to determine the sender, target, and category of the event.
- If the sender and target are known, locate these objects in the object domain.
- Locate the appropriate completion routine to complete the reception of the event. A completion routine completes the reception of the event and initiates other handling required as a result of the event. Completion routines are described in [Defining Completion Routines](#).

Relating an Event to the Domain Objects

An Integrity application contains a representation of each of the external objects in a domain map. When you build the domain map, you assign each object an external name as described in [Naming Domain Objects](#). Whenever possible, the external name you assign to the domain objects should match the name of the target and sender passed in the event.

If the names passed with the event do not match the external names used in the domain map, the parsing and decoding routines must extract the external names from the information passed. Some situations might only require a simple transformation such as concatenating text to the event string or stripping text off an event string. Other event information can be more complex and can require more complicated parsing and decoding routines.

Once you have extracted the external name, you can retrieve the domain object, using this procedure:

```
devu-domain-object-lookup
  (ext-name: text)
```

For example, an event might send the text string “Router-Hous1&ModemA&Noisy” to the external bridge. In the external bridge you would parse this into:

```
sender-string = Router-Hous1
target-string = ModemA
category = noisy
```

Next you would pass these strings to the internal part of the bridge and match the sender and target of the event with their domain objects:

```
sender, target: class opfo-managed-object;
sender = call devu-domain-object-lookup (sender-string);
target = call devu-domain-object-lookup (target-string);
```

If you want to retrieve domain objects, using criteria other than the `opfo-external-name`, you can create your own lookup procedure and define it as an alternate domain object lookup method, using the initialization item `smh-alternate-object-lookup-procedure`. For more information on how domain objects are retrieved and writing custom access methods, see `devu-alternate-object-lookup-procedure` and `devu-domain-object-lookup` in the *Integrity Utilities Guide*.

Defining Completion Routines

The purpose of a completion routine is to complete the reception of the incoming event. This can include further parsing of an event, creating Integrity messages or calling reasoning routines applied to the event. Your application must provide some method for selecting a completion procedure based on the type of incoming event.

The complexity of the completion routine depends on:

- The amount of parsing and decoding completed in the bridge. If all decoding is done in the bridge, the completion routine does not need to provide this functionality.
- How incoming events are handled. For example, some applications can automatically create an Integrity message for each event. Others can apply further reasoning routines to decide the outcome of the event. The event can be logged to a text file, discarded, or represented as an Integrity message.
- The complexity of the reasoning routines applied to the incoming events. The completion routine calls the reasoning routines applied to the event.

You can define your completion routines many different ways. One approach might be to create only one completion routine and call it regardless of the event type. This is possible when all of the events are structured in a similar manner and do not need a lot of special-purpose parsing. At the opposite extreme, you might create a completion procedure for each unique type of event that can occur. This is usually impractical in applications of any significant size and unnecessarily complex.

A useful technique is to define the completion routine based on the sender or category of the event. Because each type of sender has its own particular protocols for the events it generates, you will find it useful to use the class of the sender to determine which completion routine to call. The G2 SNMP Bridge uses the category of the event to determine the completion routine. The category in that

case combines information that relates both to the sender and the type of the event.

Automatic Trap Processing

For automatic trap processing, use the Java SNMP bridge and the current version of the GTRAP and GMIB modules. To avoid having to create your own completion procedures, use the Java SNMP bridge, because this process is automatically handled for you.

Processing The Trap

During the initial setup of the application using the Setup Wizard, the Import MIBs tab and the SNMP Setup tab play a roll in setting up the automatic trap processing. The Import MIB tab imports mibs by creating gmib-trap-properties objects and post import, defines a relationship between LinkUp and LinkDown traps by default. This relationship allows the LinkUp trap to automatically clear the LinkDown trap. Other trap properties can be related this way to provide additional clearing of traps.

The gmib-trap-properties play a role in defining what varbinds are associated with the traps and allows the processing of those varbinds. It also contains the default gtrap-trap-receiver class to create.

When the SNMP Setup tab is used there is a check-box for the Java Bridge. If this is set some internal settings are made. These settings include setting the oxsj-create-trap-mib-receiver truth-value to false. It also sets the oxsj-process-trap-structure-proc to the oxs2-default-process-trap-structure method.

The oxs2-default-process-trap-structure method finds the associated gmib-trap-properties object and creates a gtrap-trap-receiver based on the trap-class of the trap properties object. It then translates the non-default varbinds and searches for a clears-for relationship. The clears-for relationship is described above. Finally, it generates an event to be automatically processed by the generic trap completion procedure.

Generic Trap Completion Procedure

The generic trap completion procedure can automatically process traps. By utilizing this method it can process traps that have the clears-for relationship defined. This will automatically clear the message that was posted for a previously received trap. If there is no clears-for relationship defined for the newly received trap, a SMH message is created and posted to the applications message server. In addition to a message being created an ODiE event is created and posted. These ODiE events are utilized to provide out-of-box capabilities.

Creating a Domain Map

Describes how to build a domain map to represent your external objects in the Integrity application.

- Introduction **155**
- The Components of a Domain Map **156**
- Defining Domain Map Subclasses **158**
- Manually Building the Domain Map **164**
- Importing and Exporting a Domain Map **171**



Introduction

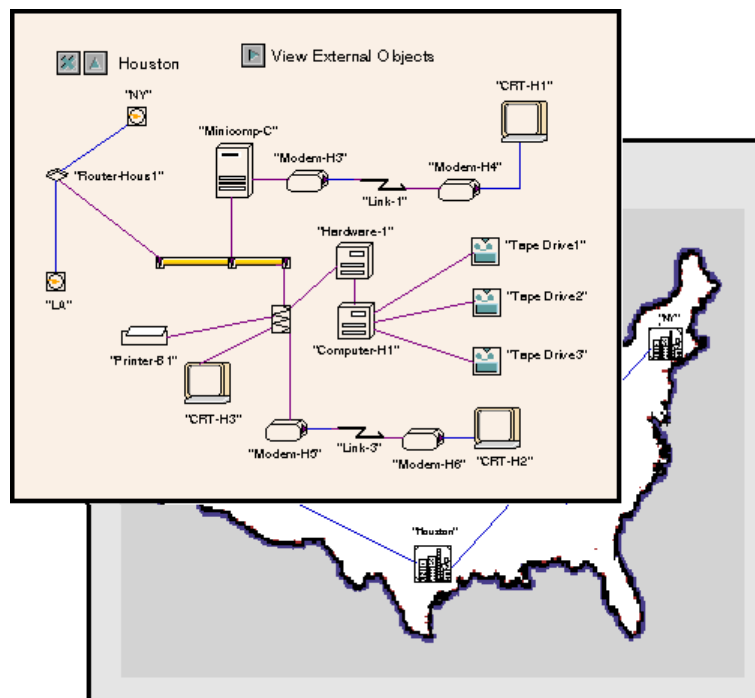
The first step to build an Integrity application is to build a domain map. The domain map models objects, connections, and containment relationships in the real, external world. This makes it easy to build and maintain extremely large collections of objects and still be able to fine tune and customize object placement.

Domain map objects, connections, and relationships are represented by G2 Integrity classes and objects. Map building and editing can be performed manually, by running a discovery job, and/or by reading in object information from an external text file. Understanding how to manually create a domain map helps you to fully understand how Integrity works.

The Components of a Domain Map

The diagram that follows shows a section of two subworkspaces representing a domain map. This figure shows examples of the following domain map building blocks:

- **Containment objects** are objects that represent virtual external objects whose sole purpose is to contain other objects. Houston, LA, and NY are examples of containment objects.
- **Managed objects** are the objects that have messages directed against them. Router-Hous1 and Printer-B1 are examples of domain objects.
- **Connections** link objects together. All the lines linking objects on the domain map are connections.
- **Connection Posts** connect objects that reside on different workspaces. Houston-to-NY and Houston-to-LA are examples of connection posts.



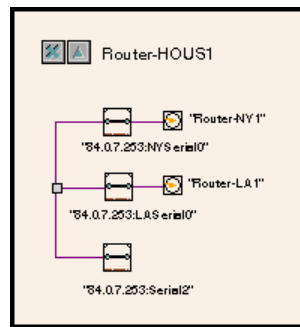
Containment Objects

Containment Objects are the domain objects that represent virtual external objects whose sole purpose is to contain other objects. You place the objects contained on a subworkspace of the containment object. Containment objects can contain domain objects of all kinds, including other containment objects. These nested objects create a containment hierarchy.

On the domain map shown in the preceding figure, Houston, LA and NY are containment objects. Containment objects are instances of a subclass of `opfo-containment-object`. The prefix `opfo` indicates Integrity Foundation classes, which are base classes needed in most applications. How to create new containment object subclasses is described in [Defining Domain Map Subclasses](#).

Managed Objects

Managed objects are objects that are not pure containment objects. Managed objects are instances of a subclass of `opfo-managed-object`. Although a managed object is not used only for containment, it can contain other objects on its subworkspace. For example, on the domain map shown in the preceding figure, the object `Router-Hous1` is a managed object that contains other objects. The objects contained in `Router-Hous1` are shown in the following figure:



You can add subworkspaces to domain objects at any time.

Managed objects are generally the active objects with attributes representing state values managed by your application. They represent real-world entities that have behavior determined by a set of states and failure modes that change over time.

How to create new managed object subclasses is described in [Defining Domain Map Subclasses](#).

Connections and Connection Posts

Connections link objects together. They are instances of the connection class or one of its subclasses. The domain map in the preceding figure uses several classes of connections to represent wires, phone lines, and ethernet wires.

You use Connection Posts to connect objects that reside on different workspaces. The connection post labeled LA, connects Router-Hous1 to Router-LA1 on the subworkspace of the object LA. Two types of connection post are used in Integrity applications:

- **dxi-linking-connection-post** is a connection post class provided as part of the Integrity Foundation Classes. Connections across subworkspaces built using the domain map importer use this class of connection post. When you click on a **dxi-connection-post**, the connection post at the other end is displayed. The *opexpkg3.kb* automatically creates a subclass of a **dxi-linking-connection-post** and places it on the Object Definitions workspace. The subclass created is named **xxx-connection-post**, where **xxx** is the prefix you assign to your application at startup.
- You define G2 connection posts in the G2 programming environment. You can create objects that automatically create subworkspaces containing G2 connection posts.

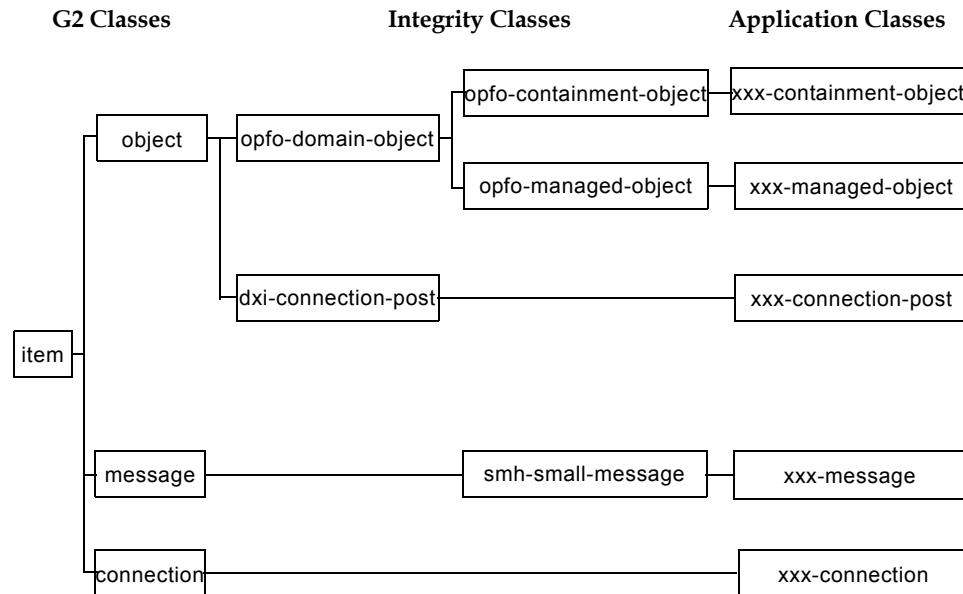
Defining Domain Map Subclasses

Before you build or import a domain map, you must create subclasses to represent the unique types of objects you are modeling and the connections that exist between them. This section describes how to define a new class, its attributes, and the icon used to display instances of the class.

You build your application subclasses by using these classes:

- **opfo-containment-object** is a class of objects whose sole purpose is to contain other objects on their subworkspaces.
- **opfo-managed-object** is a class of objects that are not pure containment objects.
- **connection** is a class of objects used to connect together domain objects. Subclasses of the basic class can define different types of wires and connections such as phone lines and ethernet cables.

The following figure shows a section of the class hierarchy in an Integrity application:



An `item` is the highest level class in G2. Subclasses of `item` include `object`, `message`, and `connection` subclasses. These subclasses are all built into G2. The next set of subclasses shown are part of the Integrity Foundation classes. `opfo-domain-object` is the superior class to both `opfo-containment-object` and `opfo-managed-object`. Another Integrity object subclass used in the domain map is the `dxi-connection-post`.

The domain object classes should be designed to take advantage of the relationships among the domain objects. For example, you might create a class for modems and have subclasses for different types of modems used. The modem class can contain attributes and behaviors (methods) common to all modems, while the specialized subclasses can contain additional attributes and model the behavior unique to that modem.

Viewing Attributes of a Subclass

Each class has attributes associated with the class.

To view the attributes of a class:

- 1 Navigate to the name of the class, then select it.
- 2 Double-click on it to display it on a workspace.
- 3 Right-click on the object to display its menu.
- 4 Click on Properties.

The attribute **Class-specific-attributes** defines attributes defined specifically for the class. Subclasses can inherit the attribute definitions from their superior class.

You can also view a table that contains the attributes of a class by selecting the icon of an instance of the class.

The basic Integrity classes **opfo-managed-object** and **opfo-containment-object** have the attributes shown in the following table:

Attribute	Description
opfo-external-name	Name you enter to refer to the object.
_opfo-highest-message-priority	Highest priority message targeting the object. This is set by the domain object alarm propagation methods, as initiated by the message system. The underscore before the name defines this as a read-only attribute.
_opfo-acknowledgment-status	Acknowledgment status of messages targeting the object. Set to unacknowledged when there are unacknowledged messages targeting the object. Otherwise acknowledged . This is set by the domain object alarm propagation methods, as initiated by the message system. The underscore before the name defines this as a read-only attribute.

Adding Attributes to a Subclass

You might have a type of object with specific attributes that you want to save along with the object.

To add a new attribute to a class definition:

- 1 Click **View Object** to display the workspace containing the class definition.
- 2 Select the object-definition icon of the class to display the menu and select **table**.
- 3 If the table is too small to read, use **CTRL+B** to expand it.
CTRL+S shrinks it. These keys can be used to expand and shrink any workspace.
- 4 Click over the text in the right-hand column next to **Class-specific-attributes** then select **edit** from the menu.

If you click on the cell but not directly on the text, you must select **edit** from a menu to display the edit box.

- 5 Type the name of any new attributes separated by a semi-colon.

For complete information about defining attributes specific to a class, refer to the *G2 Reference Manual*.

After you add attributes to a class definition, they appear on the tables for the instances of the class.

Advanced techniques for managing and customizing object classes are described in the *G2 Reference Manual*.

Displaying Attributes for a Subclass

You can set up your subclass to display the value of an attribute. Usually the *opfo-external-name* is displayed next to each instance of a domain object and containment object.

To display the external name of the domain object:

- 1 Select the object-definition of the class, then select **table** from the menu.
- 2 Click over the text in the right hand column next to **Attribute-displays**.

If you click on the cell but not directly on the text, you must select **edit** from a menu to display the edit box.

- 3 Type in *opfo-external-name* at standard position.

Use this technique to display any attribute in a class.

Creating Icons for Domain Object Classes

Every class has an icon description. When you create and place an instance of the class on a workspace, this icon becomes visible and provides direct access to that instance. When you create a new class, the class initially inherits the icon description of its superior class.

In most applications, you will want to modify the descriptions for your objects to better distinguish between them visually. You can do this by creating completely new icon descriptions, editing existing descriptions, or copying descriptions from other class definitions.

You can define and edit an icon either graphically or by using a text description:

- To edit an icon description graphically, select the object-definition and select **edit icon** from the menu.
- To edit an icon's text description, select the object-definition, select **table** from the menu, and then click on the value assigned to the attribute icon description

in the table. You can use the G2 editor to cut and paste these descriptions from other object-definitions as well.

Icon definitions can also include images imported from GIF files.

You define icons to contain multiple graphical layers. You can assign each one of these layers a different name and a different color.

To define an icon region:

- 1 Select the class object-definition.
- 2 Select **edit icon** from the menu.
- 3 Click **New**. This creates a new icon area.
- 4 Click over the text in the right hand column next to **Region**.
If you click on the cell but not directly on the text, you must select **edit** from a menu to display the edit box.
- 5 Type in the name of the region.
- 6 Draw the region in the edit rectangle to the right of the display of the icon definition rectangles.

Note Icon descriptions for domain classes you define within Integrity applications must contain definitions for two special-purpose regions: **alarm-region** and **acknowledgment-region**. These regions display the alarm priority and acknowledgment status on the domain map.

For a complete description of defining icons, refer to the *G2 Reference Manual*.

Creating Patterns for Connections

When you define subclasses of connection, you should also define how the connection appears. This is called the cross-section pattern.

To define a cross-section pattern for a connection:

- 1 Go to the object-definition defining the connection subclass.
- 2 Select the object-definition to display the menu.
- 3 Select **table** from the menu.
- 4 Click over the text in the right-hand column of the table next to **Cross-section-pattern**.
If you click on the cell but not directly on the text, you must select **edit** from a menu to display the edit box.
- 5 Type in the cross section in the edit window.

Here is an example of a cross-section pattern: 1 black, 3 gold, 1 black. This specifies a connection appearance that is five lines thick, gold with black borders, where the black borders are each one pixel thick. You can also display the available G2 colors on the color palette for changing workspace colors.

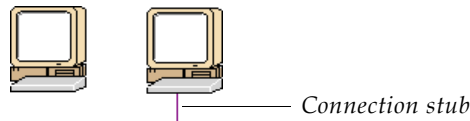
You can also define the length of the stub by editing the value of `stub-length` in the definition table of the connection subclass.

These operations are described in the *G2 Reference Manual*.

Adding Connection Stubs to Class Definitions

Most domain objects are connected to at least one other domain object. The definition of a domain class can include the definition for one or more default connection stubs. Connection stubs let you associate the proper connection class for the object, simplifying creating connections on the domain map.

The figure that follows shows an item without a connection stub and the same item with a stub. The stub provides a handle you can grab with the mouse and extend to another object.



To specify default connection stubs for new instances of any class:

- 1 Select the object-definition.
- 2 Select **table** from the menu.
- 3 Click on the right-hand column in the row containing the definition for **Stubs**.
- 4 Type in the definition for the stub.

At the bottom of the edit box, prompts appear to guide you through the syntax required to define a stub. The stub definition specifies the connection class, the location of the stub, and whether the stub is diagonal or orthogonal. Stub definition is described in the *G2 Reference Manual*.

Importing Class Definitions

Instead of creating new class definitions manually, you might want to use classes that have already been defined in other applications. You can also decide to create module libraries that contain class definitions that you commonly use.

You can use classes defined in another module several ways:

- Merge the module into your application and use the classes directly from the merged in module. This is the simplest technique, but you are left with the

overhead of everything contained in the module. You can delete unwanted items then re-save the module with a new file name. Merging modules is described in [Merging Modules](#).

- Transfer selected classes onto the workspace of a module in your application. This must be done with caution because when the classes are removed from the merged in module, any objects depending on the classes will generate inconsistency errors.
- Clone class definitions onto a workspace in your application. When object definitions are cloned, the attributes, configuration and icon definitions are all copied to the new definition, but the superior class and the name of the class must be redefined in the cloned version.

When you transfer or clone new class definitions to a module in your application, be sure to place it on subworkspaces of a workspace containing the other class definitions in the application. This will make it easy to locate.

Manually Building the Domain Map

When you manually build a domain map, you must create all of the classes needed for the domain objects and connections, create the instances and subworkspaces, place them on workspaces, and make any needed connections.

When you import a domain map from a text file, the map importer creates the domain objects based on information in the file. Imported domains reflect the hierarchy and connectivity of the domain. Imported domains do not preserve the connection class, icon scale, or icon rotation. Automatic builds are described in [Importing a Sample Domain Map](#).

Even if you are reading most of your domain information from an external file, it is still useful to be familiar with the principles of manually building domain maps as you might want to fine tune or edit domain maps manually after a build.

The steps that follows describe how you manually build a new domain map. Each of these steps is described in detail in the sections that follow.

To manually build a domain map:

- 1 Create a new application framework as described in the Introduction.
- 2 Define a class for each unique type of object you will need for your domain map. If just one or two standard connection classes are used (highly recommended), then container-object class definitions should generally be given G2's subworkspace-connection-post capability. Define a class for each unique type of object you are modeling on the domain map.
- 3 Define the types of connections used between domain objects.
- 4 Create an instance of a containment subclass, place it on a workspace, and then create its subworkspace. This is your top-level domain object. Place

subsequent containment or domain objects on the new subworkspace until you have created a complete containment hierarchy. By having a single top-level object to represent your entire domain, you simplify specification of imports and exports of the domain map. If your domain has many disparate portions, you can use a very general, abstract containment object for your top level.

- 5 Create instances of domain objects and place them on the appropriate workspaces in the containment hierarchy.
- 6 Connect the domain objects on each workspace to reflect the connectivity of the domain objects.
- 7 Place connection posts on workspaces as needed to connect items that reside on different workspaces.

The rest of this chapter describes the specific techniques for building the domain map and importing map items. It can be helpful to create a new application and follow along as the steps to creating a domain map are described.

Creating Domain Objects

Once you have created a class hierarchy to represent the different types of domain objects in your application, you are ready to create instances of those classes and place them on a domain map. The domain objects on the map are related to each other by the containment hierarchy, by connections you place between objects on the same workspace, and by connection posts with which you link objects on different workspaces.

When you begin your domain map, you generally create a top-level object that contains all the other domain objects. This simplifies importing and exporting maps, and helps navigation as users will know that all domain objects can be found underneath a single, top-level containment object, even if it has just an abstract name like “all-domain-objects”.

The top-level object is subclassed from the `opfo-containment-object` class. In your new application create a top-level object and place it on the domain map.

To create and place a top-level object on the domain map:

- 1 Create a container object using the Navigator by right-clicking on Network Diagrams and choosing New Instance.
- 2 Enter a name for the container and click OK.
- 3 Right-click the new container object you just created and choose Show Details to view its subworkspace.
- 4 Select the object-definition of the class you defined and choose Create Instance.

- 5 Drag the icon onto the subworkspace you created above and drop it by clicking on the workspace.
- 6 Name the item as described in [Naming Domain Objects](#).

In some cases, you may need to create a subworkspace for an item. Below are the steps to create subworkspaces.

To create a subworkspace of an object:

- 1 Select the object to display its menu.
- 2 Select create subworkspace from the menu.

Once you have created a subworkspace for the top-level object, place the next level of domain objects on the subworkspace. Create the domain objects by using the technique described above for creating the top-level object or use the palette objects to create a new item.

To clone objects from a palette:

- 1 Display the palettes by choosing View > Toolbox - Integrity.
- 2 Display the workspace where you want to create the new instance.
- 3 Click the mouse button to select the palette item, move the mouse to the workspace where you want to create it, and click the workspace.
- 4 Release the mouse button to drop the item on the workspace.
- 5 Name the item as described in [Naming Domain Objects](#).

Naming Domain Objects

You must assign all domain objects a value for their `opfo-external-name` attribute. This is the value you use whenever you want to refer to the domain object by name. When you want to retrieve an item using its name, you pass the `opfo-external-name` to the procedure `devu-domain-object-lookup`. Integrity does not use the G2 name facility because it does not support embedded blanks, use upper and lower case for display purposes, or allow special purpose characters often found in names used for external objects.

To assign an opfo-external-name:

- 1 Right-click the item and choose Properties.
- 2 Enter a name in the Domain Object Name text box.

If the Domain Object Name text box is read-only, then you will not be able to change the name because the object already has a name.

- 3 Click the OK button.

- 4 If the external name was not defined to be displayed for all instances of a class and you want the external name to be displayed next to the item, click on `Opfo-external-name` in the left column of the table. When the menu appears, select `show attribute display`.

Defining attribute displays for all instances of a class is described in [Displaying Attributes for a Subclass](#).

Connecting Domain Objects

After you create the domain objects, you need to connect the objects together by using the connection subclasses you have defined. You make a connection between two objects by using connection stubs. You can add stubs to a class of domain objects as described in [Adding Connection Stubs to Class Definitions](#). You can also add stubs to instances of domain objects as described in the sections that follow.

Adding and Deleting Connection Stubs from Instances

The stub definitions in a domain object-definition provide default connections as a starting point. You can add or delete stubs to or from specific objects once they are created.

To add a connection stub to an instance of an object

- 1 Select an item that uses the type of connection you want to use, select `clone` to make a copy of the item, then click to drop the item on the workspace next to the item you want to provide with a connection stub.
- 2 Click on the end of the unused stub connected to the cloned object, then drag the end into the item that you are giving the stub, and click to anchor the connection.
- 3 Click on the middle of the connection between the objects to display the connection menu, then select `delete`.
- 4 Select the cloned item, which is no longer needed, and select `delete`.

If you already have an object on the same workspace that has an unused stub of the right class, you do not need to clone an object. In this case you can use the unused stub and follow steps 2 and 3 shown in the preceding steps.

Another technique for creating stubs is to create special objects that provide stubs of several classes. This method is described in [Creating a Connection Configuration Object](#).

To delete an unused connection stub:

- ➔ Click on the unused stub and drag it into the center of the object.

Creating a Connection Configuration Object

It is sometimes convenient to create a special-purpose object specifically designed to provide connection stubs for your application. This is called a **connection configuration object**.

To create a connection configuration object:

- 1 Create a subclass of the top-level object in your module.
- 2 Define a connection stub for each connection subclass you use in your application.

Defining connection stubs for classes is described in [Adding Connection Stubs to Class Definitions](#).

To connect two objects by using a connection configuration object:

- 1 Create an instance of the connection configuration object.
- 2 Connect the connection configuration object to one of the objects you want to connect, using the stub for the desired class of connection.

- 3 Delete the connection you have just made.

This leaves behind a connection stub on the object.

- 4 Delete the connection configuration object.

Using a Stub to Create a Connection

Once you have attached the proper class of connection stub to an object, either by using a class definition, or by placing a stub on the instance, you can use the stub to form a connection of the class defined for the stub.

To use a stub to form a connection:

- 1 Click on the end of the stub and drag it into the center of the object you want to connect.
- 2 Click once to make the connection and a second time to secure the connection.

It is only necessary for a stub to be on one of the objects to be connected.

If neither object has a stub, you need to create an instance of an object with the proper stub, connect it to one of the objects, and then delete the connection you just made. To delete a connection, click on the connection, then select **delete** from the item menu. The stubs from the connection remain on both objects after you delete the connection.

Using Connection Posts

When you need to connect two objects that reside on different workspaces, you use connection posts. An Integrity application uses two types of connection posts:

- A `dxi-linking-connection-post` is a special class of connection provided in Integrity. You can click on one end of a `dxi-linking-connection-post` to display the `dxi-linking-connection-post` at the other end.
- A G2 connection post is a connection post that is part of the G2 environment. You can define classes so every instance of the class automatically creates a subworkspace that contains a G2 connection post. The system assigns the names to the connection posts.

To create and use a DXI-linking-connection post:

- 1 Create a connection class by creating a class-definition with a direct superior class of `dxi-linking-connection`.
- 2 Select the object-definition for the connection post class and select **create instance** from the menu.
- 3 Drag the instance to the first workspace and drop it by clicking.
- 4 Create a second connection post and place it on the second workspace.
- 5 Choose Tools > User Mode > Administrator.

You need to change modes because the default behavior of connection posts in developer mode is to go to the other end of the post. You must be in administrator mode to display the item table.

- 6 Select the connection post, select **table** from the menu, and enter a name for the `names` attribute.

This must be a G2 symbolic name, which cannot contain any spaces.

- 7 Select the second connection post, select **table**, and enter the same name for the `name` attribute.

`dxi-linking-connection-posts` also use a second naming attribute called `Dxi-link-name`. When a domain map is imported, the map importer sets the value of this attribute to the name of the superior object containing the matching `dxi-linking-connection-post`. The connection post displays this attribute on the domain map so you can easily identify the location of the other end of the connection.

If you manually build a `dxi-linking-connection-post`, you might want to define a value for this attribute.

to define a Dxi-link-name:

- 1 Choose Tools > User Mode > Administrator.
- 2 Select the connection post, select **table** from the menu, and enter a name for the Dxi-link-name attribute.

A G2 connection post does not automatically display the other end when you select it. However, you can define a G2 connection post as part of a class definition. This saves the time needed to create subworkspaces and connection posts each time an instance of the class is created.

To create and use a subworkspace connection post:

- 1 Click on the object-definition for the object that will have the subworkspace, then select **table** from the menu.
- 2 Select the text in the right hand column next to the attribute **Instance Configuration**. Type in *declare properties as follows: subworkspace-connection posts*.
- 3 Check to be sure that the object-definition contains at least one stub definition. If no stub definition exists, define one as described in [Adding Connection Stubs to Class Definitions](#).
- 4 Create an instance of the object. This is the superior object.
- 5 Click on the superior object and select **create subworkspace** from its menu. If you go to the subworkspace of this object, you will see that there are connection posts placed automatically, arranged in the same layout as the stubs. The connection posts are automatically given names that link them to the stubs on the superior object.

You can hook up connections to the stubs of the superior object, and hook up objects via connections to the subworkspace connection posts. Connectivity is maintained between all objects connected to the superior object and to objects connected to the subworkspace connection posts. This allows multiple levels of connectivity to be tested in your applications.

If you delete a stub, you must manually delete the corresponding connection post. If you add an additional stub to the superior object, a corresponding connection post is automatically placed on the subworkspace.

These features are described in the *G2 Reference Manual*.

Importing and Exporting a Domain Map

In most Integrity applications, the large number of external objects makes it impractical to build the domain map by hand. Integrity provides a map import facility to let you automate the building of the domain map. The domain map importer reads information about the external objects from a text file. You create the text file using whatever system you use to manage the objects.

Integrity also provides a domain map exporter. This lets you save your domain map to a text file that can later be imported back into your application or into another application.

The sections that follow describe an example using the *doc_demo.kb* sample application that demonstrates the use of the map export and import features.

Exporting a Sample Domain Map

This example shows how you can export a portion of the *opx_demo* domain map. Later you will delete the exported objects from the domain map, then use the domain map importer to restore the deleted objects.

First, you will export the MASTER-CONTROL-FACILITY and all of the object contained within NY.

To export a section of the *opx_demo* domain map:

- 1 Load *opx_demo.kb*.
- 2 Choose View > Toolbox - Integrity Export Import to display the export/import palette.
- 3 Create a new workspace.
- 4 Select the Dxi3 File Export Object from the File Export/Import palette, move the mouse over the new workspace, and click to place the object on the workspace.
- 5 Right click on the export object and choose Properties to configure these properties:
 - a Enter MASTER-CONTROL-FACILITY for the Source Workspace.
 - b Click "..." button to navigate to a destination directory to specify a destination file.
 - c Click the OK button to close the Properties dialog.
 - d Right-click on the export object and choose Start File Export to start the export process.

An attribute display on the export object tells you how many objects were exported. The file exported is an ASCII text file that defines the domain objects using the Domain map eXport/Import (DXI) file format.

The exported file uses the following format:

```
*****<object id> ( required)
action: {create|delete} [<type>] (optional)
superior: [<object id>] (optional)
delete-superior: [<object id>] (optional)
connected: [<object id>,...] (optional)
delete-connected: [<object id>,...] (optional)
attribute: <attribute name> = [<attribute value>] (Nxoptional)
relation: <relation name> [<object id>,...] (Nxoptional)
delete-relation: <relation name> [<object id>,...] (Nx optional
+ required empty line at end)
```

Importing a Sample Domain Map

To import a domain map, you create a text file that is similar to the one shown in the preceding example. As in the example, your file must contain a two line informational header, the ten-line definition header shown in the preceding table and at least two lines to define the values and relationships of each domain object to be imported. You create the file by outputting the text file in the DXI format from a database or other manager connected to your external objects.

Caution When you create a file for import, be sure that extra lines are not embedded in or at the end of the import file. This file should NOT contain tabs.

You can step through the process of importing a text file by following this example. Before you can follow these directions, you must export a DXI file using the preceding instructions to produce the file `master-control-facility.txt`. Before importing the objects, you also must delete the exported objects from the domain map. This will let you see how the objects are recreated using the Integrity import feature.

To delete the objects exported in the preceding example:

- 1 Using the Go To feature in the Standard toolbar, enter `master-control-facility` and press the Enter key.
A red arrow points to the item.
- 2 Right-click the `master-control-facility` and choose Show Details.

- 3 Select all devices on the subworkspace of the `master-control-facility`.
- 4 Right-click on one of the devices and choose Delete, then click OK to confirm the deletion.

Now that you have deleted all the objects exported to `master-control-facility.txt`, you can use the Integrity importer to return them to the application.

Caution Do not save `opx_demo.kb` after you have modified it for this example.

To import a sample DXI file:

- 1 Choose View > Toolbox - Integrity Export Import.
- 2 Click on the Dxi3 File Import object, move your mouse over the workspace you created in the import section, and click on the workspace to drop the item.
- 3 Right-click on the import object and choose Properties.
- 4 Select DXIDB-EXAMPLE-TRANSLATIONS for the Type To Class Workspace.
This defines the workspace where translation objects reside that will translate text to a symbol representing a class defined in the application.
- 5 Select OPFO-CONTAINMENT-OBJECT for the Default Class To Create.
- 6 Specify different values for the Column Height, Vertical Spacing, and Horizontal Spacing, as needed.
These are used as buffers between the placement of the newly created items.
- 7 Use the "..." button for the Local File Name to select the `master-control-facility.txt` file you created earlier in the import example.
- 8 Click OK to close the properties dialog.
- 9 To begin the import, right-click the import object and choose Start File Import.

You can display the domain map after the import to view the results. If you want to add connections on your domain map, you must add them manually after the objects are imported.

Using Translation Objects

When your application reads an object definition from a DXI import file, one of the first things it must do is determine the class of the object to create. The value written to the Type line in the DXI text file provides this information. You can supply two different values to type to provide information about the class of the object:

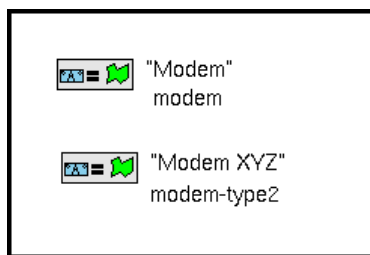
- The name of the class

- A string that is mapped to the class, using a translation object that maps the name of the string to a class

If it is easy to produce a DXI file that directly contains the name of the class of each object; this is an effective method to use to determine the class. Depending on the application, however, you may find it easier to translate a text string into a class name by using translation objects.

The translation object has two attributes, a class name and a text string. The importer matches the text string of the **Type** field to the text string in each translation object on the specified workspace. If the importer finds a match, it creates the class specified by the translation. If it finds no exact string match, the importer uses a “contained-in” approach to match translation strings to type string. If all efforts to determine a class fail, the importer creates an instance of the default class defined within the import object.

For example, two translation objects are shown in the following figure. The top label is the name of the **type** string from the external file and the bottom label is the class name that matches the **type**.



The table that follows shows how the importer maps several different **type** text strings, using these translation objects.

Type Text String	Class
Modem	modem
Modem XYZ	modem-type2
Modem DEF	modem
Modem XYZ4	modem-type2

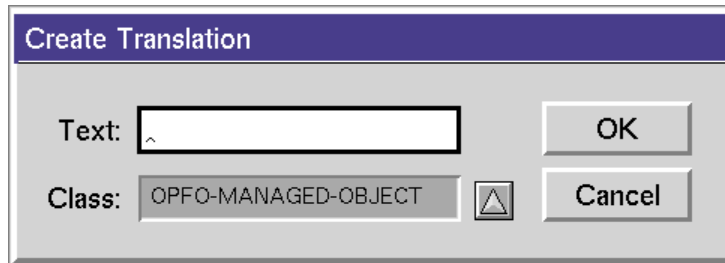
You create translation objects by using the palette blocks.

To create a translation object:

- 1 Choose View > Toolbox - Integrity Export Import.
- 2 Click on the Dxi3 Type To Class object, move your mouse to the workspace you created for the export example, and click on the workspace to drop the item.

- 3 Right-click on the block and choose Properties.

The translation object dialog is shown in the following figure:



- 4 Enter in the Type text string in the Name edit line.
- 5 Click the button displaying the triangle, then select the class you want to use for the Type text string you have defined and click OK.

Message Handling

Defines the Integrity message system including how to create message servers, browsers, and status bars, how to manage Integrity messages and their histories, and how to use the logging facility.

Introduction	177
Setting up the Message System	178
Working with Messages	205
Message Alarm Propagation	215
Logging Messages and Events	217
Error Handling	221
Creating User Defined Effects	223



Introduction

The previous chapter described how events come into Integrity. Once an event exists, you must make a decision as to how to respond to the event. The possible responses include:

- Display information about the event to an operator.
- Maintain a history of the event.
- Take an action.
- Discard an unimportant event.

You can select any combination of these to respond to an event.

You accomplish both the display of information and the maintenance of histories by using an Integrity structure called a **message**. Messages relate the events that occur to the domain objects they describe.

Each message is uniquely defined by the attributes:

- **Sender** - The object that sent the event.
- **Target** - The object the event is about.
- **Category** - The type of event.

By the time you decide to create a message, you have already defined the target, sender, and category of the event in the bridge or in a completion routine.

When you create a message, you give it a priority and assign it to a message server. A message server is an object that holds messages. A browser is an object used to view the messages in servers. A browser can subscribe to one or more message servers. More than one browser can subscribe to a single message server at the same time. When a browser has subscribed to a message server, the message server will send new messages to the browser as they are received. Browsers can also filter messages they receive from message servers. Filtering can be based on message priority, the sender of the message, the target of the message, patterns in the category of the message, or the value of any attribute of a message.

When you create a message, the message sends an alarm to the target object on the domain map. This alarm causes the alarm region of the object to change to the color associated with the priority of the message. When you use the default alarm propagation methods, the alarm propagates to the object that contains the target object. This propagation continues up until the top of the object containment hierarchy is reached. You can also define custom alarm propagation methods.

The rest of this chapter discusses the items you create to support the message system, the behavior of messages and how to create, acknowledge, delete and display messages.

Setting up the Message System

Before you can use the message system, you need to define certain objects within your application. These include:

- Servers that contain the messages.
- Browsers to view the messages.
- Status Bars to view the priority and acknowledgment status of the servers. These are optional objects.

- Escalation Objects to define routines called during defined phases of the message's life. These are optional objects.
- Custom Message Handling Routines that you provide to add additional behaviors to the basic message system. These are optional objects.

In addition to creating these objects and procedures, you might want to initialize certain parameters that affect the behavior of the message system. These are described in Setting Priority and Acknowledgment Colors on page 215.

Defining Message Servers

A **message server** is a container that holds messages. When you create a message, you assign it to a server. When you create a new application, the startup routine creates a server workspace and creates a server with the name xxx-message-server, where xxx is the prefix you selected for your application, on the server workspace. This default message server can be re-configured or deleted. By creating multiple servers, and assigning similar messages to a server, you can organize messages. For example, you might want to create one server that receives all Out of Service messages. You create, configure, and manage servers by using the palette blocks.

To create a new message server:

- 1 From the Integrity Components palette, select the message server block.
- 2 Select the module where you want to place the message server. Click OK.
- 3 Complete the configuration dialog shown below and click OK.

The screenshot shows a dialog box titled "Create Message Server". It has the following fields and controls:

- Name:** A text input field containing the text "NONE".
- Message class:** A dropdown menu currently displaying "SMH-SMALL-MESSAGE".
- Maximum history length per message:** A spin box with the value "300".
- Logging manager:** A dropdown menu currently displaying "UNSPECIFIED".
- Buttons:** "OK" and "Cancel" buttons are located on the right side of the dialog.
- Area below Logging manager:** A large, empty rectangular area with a vertical scrollbar on the right side, likely intended for a list of logging managers.

The following table describes the items on the configuration dialog:

Configuration Item	Description
Name	G2 symbolic name used to define the message server.
Message class	Class of the message created and stored in the message server. It must be a subclass of <code>smh-small-message</code> . To view a list of the message classes defined in the application, select the button displaying the triangle next to the Message Class edit line.
Maximum history length per message	Maximum number of history timestamps stored for a message in the message server.
Logging Manager	Logging manager used for logging the messages in the message server. Existing logging managers appear in the scroll area. Logging managers are described in Logging Messages on page 219.

To configure a message server:

- 1 Select the message server.
- 2 Double click on the message server you want to configure, then click Configure...

The configuration dialog is identical to the one shown above for creating a new message server.

Defining Browsers

A **browser** displays the messages in one or more message servers. You can also configure browsers to show the values of any attributes of the messages in columns of the scrolling message display. You can add filters to a browser to display only messages with a specified priority, acknowledgment status, target, sender, category that matches a specified text pattern, or that match a particular value for any attribute of the message.

Creating Browser Templates

A browser template is a master copy of the browser. The first time a user views a browser, a copy of the browser template is created and displayed on the user's window. The browser template does not display messages. A quick way to distinguish a browser template from a copy of the browser is that on the browser template the word "sample" appears in each column of the message scroll area. Any changes made to a copy of a browser are temporary and will be lost the next

time the system is started. If you want your changes to the browser permanent, make the changes to the browser template.

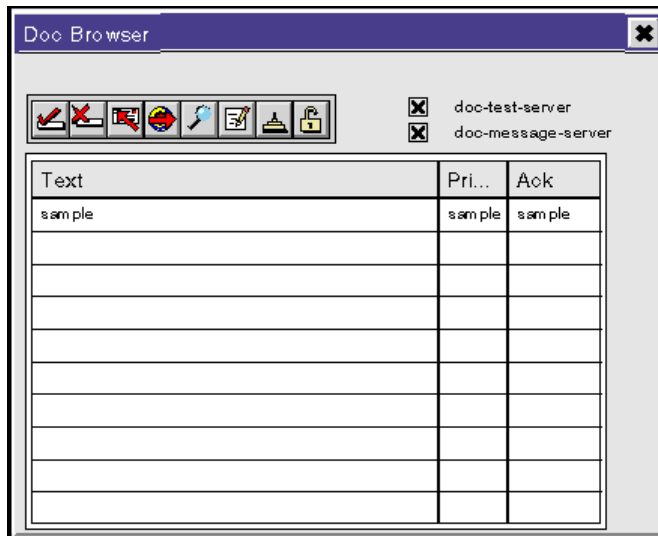
To create a browser template:

- 1 Click Browser Templates block from the Integrity Components palette.
- 2 Select the module, then click OK.
- 3 Complete the configuration dialog shown below and click OK.

The following table describes what to enter in the configuration dialog:

Configuration Item	Description
Name	G2 symbolic name to define the browser.
Display Procedure	Procedure called to display the browser. Leave the default unless you want to call a custom procedure. See <i>Writing Custom Procedures to Display and Hide a Browser</i> on page 193.
Hide Procedure	Procedure called to hide the browser. Leave the default unless you want to call a custom procedure. See <i>Writing Custom Procedures to Display and Hide a Browser</i> on page 193.

After you select OK, a browser template and a Subscriber/Filters palette appear. The browser template is shown in the following figure:



Configure this template according to the directions below.

Configuring a Browser

You can configure a browser by cloning and placing objects from a Subscribers/Filters palette onto the browser template to define the message servers and filters available to the browser. The Subscribers/Filters palette is shown in the following figure:



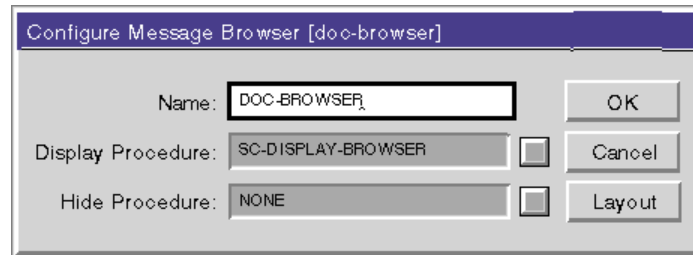
To configure a browser template:

- 1 Display the browser template.

The template and palette is automatically displayed after a browser is created. To display a browser template for an existing browser:

- a Locate the browser using the finder or the explorer view.
- b Select the name of the browser and click Configure...

The configuration dialog is shown below:



- c Click the Layout button on the configuration dialog.
- 2 Create and configure the subscribers. See *Creating and Configuring Subscribers* on page 183.
- 3 Create and configure the filters. See *Creating and Configuring Filters* on page 184.
- 4 Define the sorting characteristics of the browser. See *Defining the Sorting Characteristics of the Browser* on page 185.
- 5 Configure the columns on the message scroll area. See *Configuring the Columns of the Browser* on page 187.
- 6 Arrange the items on the browser template. See *Arranging the Items on the Browser Template* on page 193.
- 7 When you have completed the configuration, select the hide button in the upper right hand corner of the browser template.

If copies of the browser exist, you will be prompted to delete these copies.

Creating and Configuring Subscribers

You can configure the browser to receive messages from one or more message servers. You define the message servers displayed by a browser by selecting the palette item `smh-subscriber` from the Subscribers/Filters palette and by placing it on the browser template. The subscriber object provides a check box used at run time to subscribe or unsubscribe to the server it represents. This palette and the browser template must be visible to configure the choice of message servers.

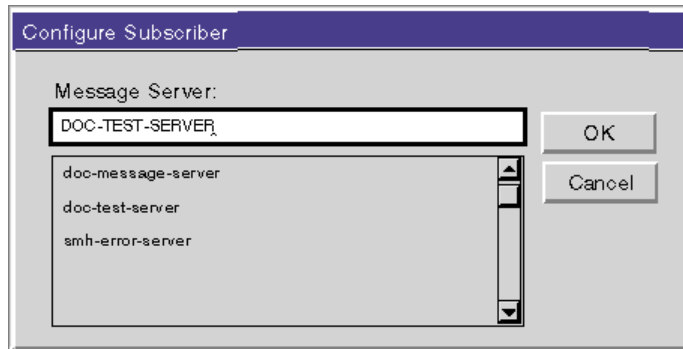
The Subscribers/Filters palette is shown in *Configuring a Browser* on page 182.

To create a subscriber for a browser template:

- 1 Display the browser template and the Subscribers/Filters palette as described in *Configuring a Browser* on page 182.
- 2 Click the SMH Subscriber check-box on the Subscribers/Filters palette to clone it.
- 3 Click on the Browser Template to drop the check box on the template.

- 4 Click on the check box to display the menu and choose configure...

The figure below shows the dialog you use to configure the subscriber button:



- 5 Select the message server from the scroll area whose messages you want to show in the browser.
- 6 If you want the browser to display all messages contained in the message server the first time it is displayed, click on the check box to display its menu and select `sc-toggle`.
- 7 If you have finished configuring the subscribers, filters, and scroll area of the browser, select the hide button in the upper right corner of the browser template.

If copies of the browser exist, you will be prompted to delete these copies.

Creating and Configuring Filters

You can configure the browser to filter the messages contained in the message servers it displays. You define the filters used by a browser by selecting the filter items from the Subscribers/Filters palette and placing them on the browser template. The Subscribers/Filters palette is shown in Configuring a Browser on page 182. You can apply five different types of filters:

- Generic Attribute Filter to filter messages using any attribute of the message.
- Priority Filter displays only messages with the specified priority.
- Category Filter displays only messages with the specified category. The characters "?" for one character and "*" for multiple characters can be used as wildcards.
- Target filter displays only the messages that target the specified domain object and all the domain objects in its subworkspace hierarchy.
- Sender filter displays only the messages that are sent by the specified domain object and all the domain-objects in its subworkspace hierarchy.

To create a filter for a browser template:

- 1 Display the browser template and the Subscribers/Filters palette as described in Configuring a Browser on page 182.
- 2 Click the check-box next to the filter on the Subscribers/Filters palette to clone it.
- 3 Click on the Browser Template to drop the check box on the template.
- 4 Click on the check box to display the menu and choose **configure...**
- 5 Configure the filter.

The value depends on the type of filter used. Filter values for each type of filter are:

- a Generic Attribute Filter - The name of the attribute of the message to match and the value for the attribute.
 - b Priority - A priority value selected from a list of priority values defined for the system, each of which has an associated color.
 - c Category - A text string matching the category of the filter. The text string can contain "*" for matching any 0 or more characters and "?" for matching any one character in order to perform wild card matches. The text string does not require quotes.
 - d Target - The Opfo-external-name of the target object matched by the filter.
 - e Sender - The Opfo-external-name of the sending object matched by the filter.
- 6 If you want the browser to activate the filter the first time it is displayed, click on the check box to display its menu and select **sc-toggle**.
 - 7 If you have finished configuring the subscribers, filters, and scroll area of the browser, select the hide button in the upper right corner of the browser template.

If copies of the browser exist, you will be prompted to delete these copies.

Defining the Sorting Characteristics of the Browser

You can define a browser to automatically sort incoming messages or to place new messages at the top or bottom of the browser. When you create a new browser template, automatic sorting is disabled by default.

Caution Automatic sorting adds a performance overhead to the message handling of the browser. If your system receives a high volume of messages, you should leave automatic sorting disabled.

To define the sorting characteristics of the browser:

- 1 Display the browser template as described in Configuring a Browser on page 182.
- 2 Click the mouse between the double lines that form the border of the scroll area containing the browser columns, then select **table**.

The figure below shows the attributes used to configure the sorting characteristics of the browser view:

a gqsv-view-configuration	
Item configuration	none
Add new items first or last	last
Scroll to display new items	false
Automatically resort new items	false
Automatically resort attribute changes	false
Key for column to sort initially	smh-creation-time
Initial sorting order	ascending

The attributes on this table are:

- **Add new items first or last** - Specify *first* to add all new messages to the top of the browser rows; *last* to add new messages to the bottom of the browser rows. This option only matters if **Automatically sort new items** is *false*. The default for a new browser template is *last*.
- **Scroll to display new items** - Specify *true* to enable scrolling when a new message is added to browser view; *false* to disable automatic scrolling. Setting this to true adds a performance overhead to the message handling of the browser. If your system receives a high volume of messages, you should leave this attribute as *false*. The default for a new template is *false*.
- **Automatically sort new items** - Specify *true* to sort messages as they are created and displayed; *false* to disable automatic sorting. The initial sort criteria is specified in **Key for column to sort initially**. The user can change the sort criteria by selecting the title of a column or, the time sort button on the browser view. When no columns are selected, the sort is done on time. To maximize performance, do not use automatic sorting. The default for a new template is *false*.

Note To enable sorting based on time, there is a 0 width column on the default browser template defined for the attribute **smh-creation-time**. This is not visible to the user but is used for sorting purposes. Do not delete this column.

- **Automatically resort attribute changes** - Specify *true* to resort the messages whenever an attribute used in the table selected as the sort criteria changes;

false not to resort. The default for a new template is *false*. To maximize performance leave this set to *false*.

- **Key for column to sort initially** - The initial sort criteria used when the browser is displayed for the first time. Enter the name of the column you want to use for the initial sort criteria. The default is *smh-creation-time*.
- **Initial sorting order** - Specify *ascending* to sort in ascending order; *descending* to sort in descending order. The user can reset this value using the Reverse sort order button on the browser view. The default is *ascending*.

In addition, the user can sort all the messages in the browser based on the values displayed in a column by clicking on a column header if that option is allowed in the column header configuration of the column. This can be done even when the browser is locked. Also, the user can go from any column value sort criteria to a sort based on *smh-creation-time* of messages by selecting the time-sort button on the browser.

When *automatically-sort-new-items* is set to *false*, the system adds all new messages at the top or the bottom depending on the value of *add-new-items-first-or-last*, regardless of the sort criteria selected.

Configuring the Columns of the Browser

Columns in the scroll area correspond to attributes of messages. You can display the values of any attribute of a message in named columns. At run time, users can click on a column heading to display the messages in a sorted order based on the values of that attribute.

You must configure changes to the columns on the message scroll area on the browser template. Displaying the template is described in *Configuring a Browser* on page 182.

To change the location of a column:

- ➔ Drag the title of the column to the new location and release the mouse button.

To delete a column:

- ➔ Click the mouse on the title row of the column you want to delete and Select delete column.

When you have finished configuring the subscribers, filters, and scroll area of the browser, select the hide button in the upper right corner of the browser template. If copies of the template exist, you will be prompted to delete these copies.

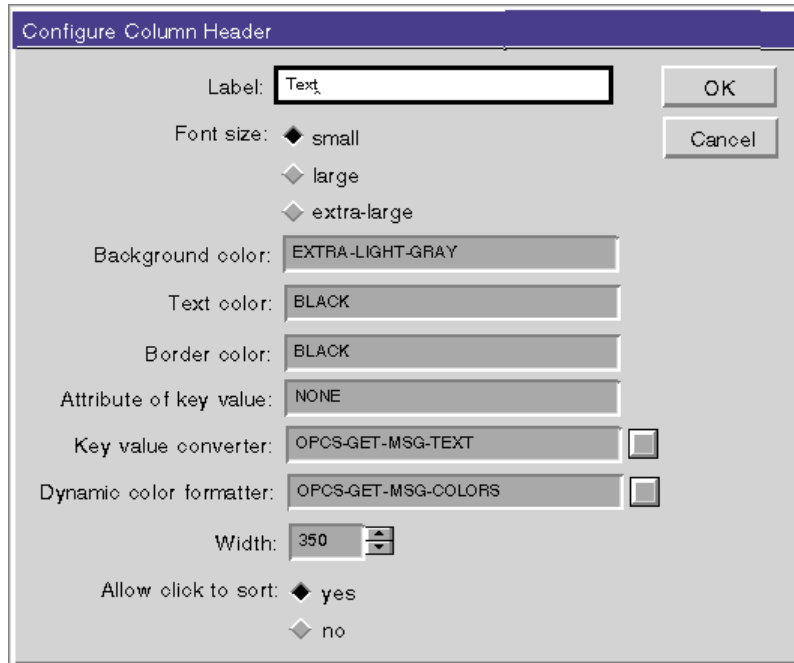
To add a new column:

- 1 Click the mouse on the title row of a column and select Add new column after or Add new column before.
- 2 Configure the new column as described below.

To configure a column on the browser template:

- ➔ Click on the column title and select configure... then complete the configuration dialog.

The dialog shown below appears:



The table below describes this dialog:

Dialog item	Description
Label	Column header label.
Font size	Size of the font. Can be <i>small</i> , <i>large</i> or <i>extra large</i> .
Background color	Default background color of the title of the column.
Text color	Default text color of the title of the column.
Border color	Default border color of the title of the column.
Attribute or key value	Name of a message attribute or a key which defines the contents of the column.

Dialog item	Description
Key value converter	Name of a procedure that takes in an item (message) and a symbol (attribute or key), and returns the value to display in the column cells.
Dynamic color formatter	Name of a procedure that takes in an item (message) and a symbol (attribute or key), and returns three symbols representing the background, text and the border color for the cell. The default <code>opcs-get-msg-colors</code> returns colors that signify the priority and acknowledgment status of the message. When a new column is added no default is provided so you must enter <code>opcs-get-msg-colors</code> if you want to use the default Integrity message colors.
Width	Width of the column. If the width of a column is set to 0, the column does not appear on the browser view.
Allow click to sort	true if you want to allow the user to sort the messages displayed on the browser based on the values in this column, otherwise false. This sorts existing messages. If automatic sorting is enabled, incoming messages are also sorting using the values in the column

When you add a new column, you need to define how the system determines the value to place in the column. You also must define the colors to use to display the background, text and border of the column.

The contents of the browser column are defined by the procedure entered as the Key Value Converter. The procedure defined as the key value converter is passed a message and the Attribute or Key Value entered for the column. You can write your own key value converter procedure, or you can use one of the two procedures Integrity provides. These are:

- `opcs-get-msg-text` - This procedure places the text of the message in the column. This procedure is the default called to get the value of the Text column on the default browser.
- `gqsv-get-attribute-value` - This procedure uses the Attribute or Key Value to find the value of the message attribute specified. If you use this procedure be sure that the Attribute or Key Value is the name of a valid message attribute for the class of message passed to `gqsv-get-attribute-value`. This procedure is

called to get the value for the Priority and Acknowledgement columns on the default browser.

To create a custom Key Value Converter procedure:

- 1 Display the browser template as described in Configuring a Browser on page 182.
- 2 Select the label of the column you want to configure, then select **configure ...**
- 3 Select the button next to the Key Value Converter edit box.

This button icon displays ...

- 4 Type in the name of your custom procedure and click OK.

This creates a template for your custom procedure and places it next to the browser template item.

To view and edit your custom procedure:

- 1 Locate the browser and highlight it.
- 2 Click **Go To** on the toolbar.
This takes you to the workspace where the browser template is defined.
- 3 Select the definition of the procedure, then select **edit** to define the contents of the procedure.

The template provided for a custom key value converter procedure named `my-key-value-converter` is shown below:

```
my-key-value-converter (Msg: class message, Key: symbol ) = (value)
begin
{The type of value returned by this procedure should match the cell-type
attribute of the column}
end
```

`doc_demo.kb` provides examples of custom browser configurations on the browser `doc-custom-browser`. On this browser, the two column `Target` and `Category` both use the Integrity procedure `gqsv-get-attribute-value` to provide the text for the column. In the `Target` column the attribute `smh-target` is passed to `gqsv-get-attribute-value`. In the `Category` column the attribute `smh-category` is passed to `gqsv-get-attribute-values`.

The column **Ack** uses a custom key value converter procedure named **doc-get-ack-status**. This procedure is shown below:

```
doc-get-ack-status(Msg : class message, Key : symbol) = (value)
status: text;
begin
if the smh-acknowledgement-status of msg is acknowledged
    then status = "X" else status = " ";
return status;
end
```

This procedure uses the value of the attribute **smh-acknowledgement-status** to place an "X" in the **Ack** column when the message is acknowledged and an empty string when the message is unacknowledged.

When you create a new column, you must also define the procedure to use to set the colors for the column. If you want to use the default color formatter, enter **opcs-get-msg-colors** as the value for the attribute **Dynamic Color Formatter** on the **Configure Column Header** dialog. This default procedure sets the background color of the cell to the background color assigned to the priority of the message by the initialization **opcom-priority-alarm-colors**. The text is set to the text color assigned to the priority of the message by the initialization **opcom-priority-alarm-text-colors**.

When the message is acknowledged, the background color of the cell is set to white and the border is set to the color associated with the priority of the message.

You can also create a custom color formatter.

To create a custom dynamic color formatter procedure:

- 1 Display the browser template as described in **Configuring a Browser** on page 182.
- 2 Select the label of the column you want to configure, then select **configure ...**
- 3 Select the button next to the **Dynamic Color Formatter** edit.

This button icon displays ...

- 4 Type in the name of your custom procedure and click **OK**.

This creates a template for your custom procedure and places it next to the browser template item.

To view and edit your custom procedure:

- 1 Locate the browser and highlight it.
- 2 Click **Go To** on the toolbar.

This takes you to the workspace where the browser template is defined.

- 3 Select the definition of the procedure, then select **edit** to define the contents of the procedure.

The template provided for a custom key value converter procedure named `my-dynamic-color-formatter` is shown below:

```
my-dynamic-color-formatter (Msg: class message, KeyOrAttribute: symbol ) =
(symbol,symbol,symbol)
begin
{This procedure should return the name of three G2 colors representing the
background color, the text color, and the border color of the column}

return the symbol white, the symbol, the symbol black, the symbol black;
end
```

The message and the key defined for the column is passed to the dynamic color formatter. The procedure must return the symbols that define the background, text, and border colors for the cell in this order.

The browser `doc-custom-browser` contains a column used to display the name of the city that contains the target object. The color of the column for the city is determined using the procedure `doc-color-by-city`.

The custom color formatting procedure from `doc_demo` is shown below:

```
doc-color-by-city(Msg : class smh-transient-message, KeyOrAttribute
:symbol) = (symbol, symbol, symbol)
color-back, color-text, color-border: symbol;
target-object: class doc-managed-object;
begin
target-object = call devu-domain-object-lookup (the smh-target of msg);
if the workspace of target-object is the same object as the subworkspace of
Houston
OR the workspace of target-object is the same object as the subworkspace of
Router-HOUS1
then color-back = the symbol light-yellowelse color-back = the symbol
light-cyan;
if the smh-message-category of msg = "Possible Problem"
then color-text = the symbol red
else color-text = the symbol black;
color-border = the symbol black;
return color-back, color-text, color-border;
end
```

In this example, the color of the background of the cell is determined by the city that is the container of the target object. The text is set to black unless the message category is "Possible Problem" in which case the text is red. The border color is set to black.

When you do not use the default procedure `opcs-get-msg-colors` to define the Dynamic Color Formatter for a new column, the colors are not affected by changes in priority or acknowledgment status.

Arranging the Items on the Browser Template

Once you have selected all the filters and subscribers and have configured all of the columns on the browser template, you can move the items around into the design that you prefer. You can also add other objects such as text or buttons to the browser.

Note To make your changes permanent, you need to make them on the browser template not on the view of the browser.

You can drag items on the browser template to move them to any desired location. You can also move a group of objects as described in *Working with G2 Objects* on page 144. By moving an entire group of objects you can move the tool bar on the browser template and the column scroll area.

Other objects can be added to the browser template just as to any other workspace. Customizing a workspace is described in *Working with G2 Objects* on page 144.

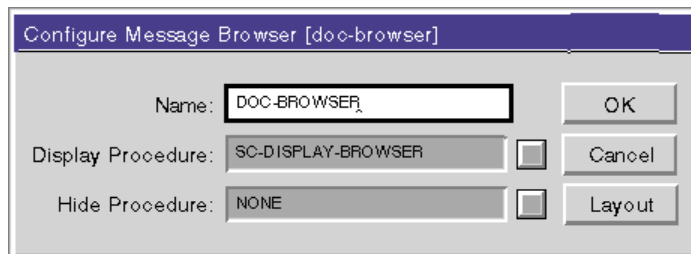
Writing Custom Procedures to Display and Hide a Browser

You can write your own procedures to customize what happens when a browser is displayed or hidden. You enter the name of your custom procedures in the browser configuration dialog.

To define a custom procedure to display a browser:

- 1 Locate the browser.
- 2 Double-click on the name of the browser and click `Configure...`

The browser configuration dialog is shown below:



- 3 Click the square button labeled `...` that is next to the edit box for the Display Procedure.

A dialog appears that asks for the name of the new procedure.

- 4 Enter the name you want to use for your custom procedure in the dialog then click OK.
- 5 If you have finished configuring the rest of the browser, click the OK button on the Configure Message Browser dialog.

The system creates the procedure you named and places it alongside the browser definition.

The dialog to Create a browser is the same as the one to Configure an existing browser, so you can also define custom procedures when you create the browser.

Use the same technique to create a custom procedure called when you hide the browser.

To view and edit your custom procedure:

- 1 Locate the browser and highlight it.
- 2 Click Go To on the toolbar.

This takes you to the workspace where the browser is defined. The browser workspace from *doc_demo.kb* is shown in the figure below:



In *doc_demo* the procedures, *doc-browser-display* and *doc-browser-hide* were created for the message browser *doc-option-browser* using the technique described above. When the system creates the procedures, it assigns them the arguments that your custom procedure must contain.

Note If your procedure definitions overlap, drag them apart to view them clearly.

- 3 Select the appropriate procedure object, then select **edit** to define the contents of the procedure.

The `doc_demo` custom procedures are shown below. Nothing has been added to these procedures. The system creates this code to provide the arguments and calls you must include in your custom procedures.

```

doc-browser-display (source: class item, target: class item,
                    wksp:class kb-workspace, win: class g2-window)
begin
    call sc-display-browser (source, target, wksp, win);
end

doc-browser-hide (wksp: class kb-workspace)
begin
end

```

Note Integrity hides the browser before it calls `doc-browser-hide`.

The arguments passed to these procedures are:

Argument	Description
source	Browser template you are configuring
target	Browser template you are configuring
wksp	Workspace of the view of the browser created when you select the Browser.
win	G2-window displaying the view of the browser.

You can see from the example that a new display procedure must call the Integrity procedure `sc-display-browser` to display the browser. You can add any other behavior in the procedure. Custom hide procedures do not need to call any Integrity procedures.

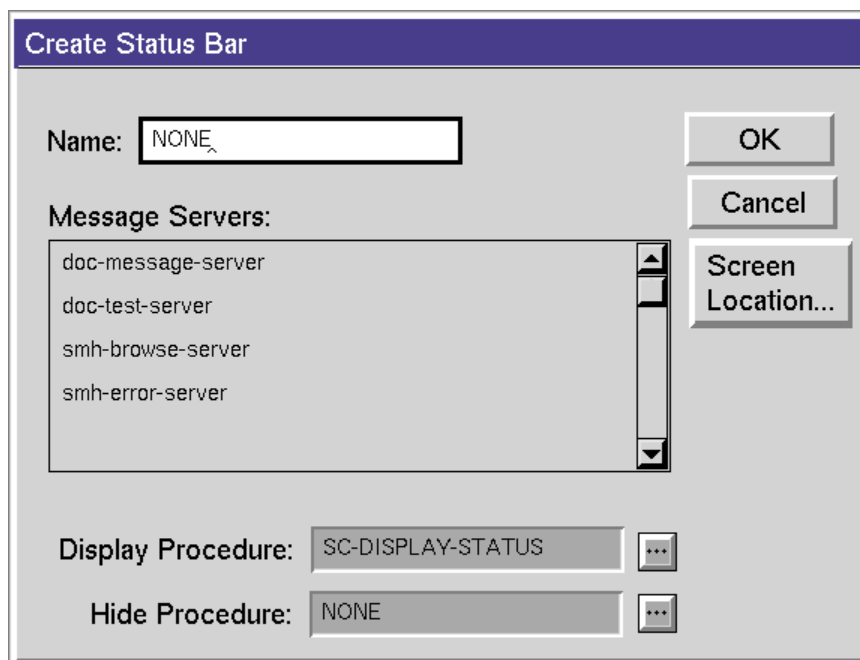
Defining Status Bars

A **status bar** is an indicator that displays the highest priority message posted in a specified message server and whether the message has been acknowledged or not. It provides an overview of the status of the message servers. Like browser templates, status bar templates are a master copy of the status bar. The indicators for each message server do not change colors on the status bar template, but do on a status bar copy. When a user selects a Status Bar, a copy of the status bar is created for a window.

At run time, the color of an indicator always corresponds to the highest priority message for that server, regardless of acknowledgement status. An sc-smh-status-subscriber's icon has two concentric icon-regions. The inner circle is only filled in when there are unacknowledged messages. This applies regardless of the priority of the unacknowledged messages. When all messages in a server are acknowledged, only the outer concentric circle is colored.

To create a new status bar template:

- 1 Click on the Status-bar block from the Integrity Components palette.
- 2 Select the module where you want to place the status bar, then click OK.
- 3 Complete the configuration dialog shown below and click OK.

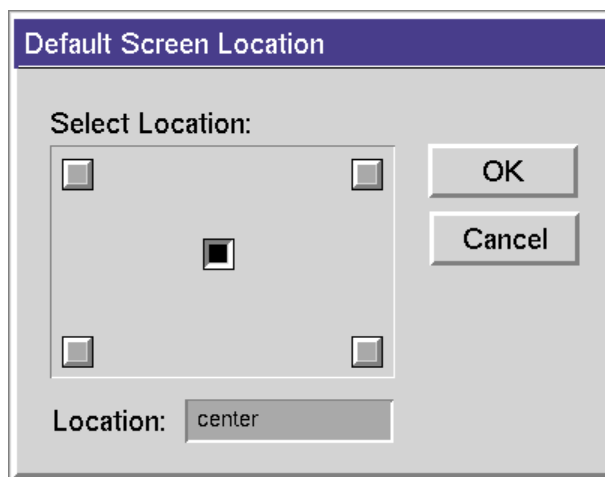


The following table describes what to enter in the configuration dialog:

Configuration Item	Description
Name	G2 symbolic name to define the message server.
Message Servers	Message servers whose message status is displayed on the status bar.

Configuration Item	Description
Display Procedure	Procedure called to display the status bar. Leave the default unless you want to call a custom procedure. See Writing Custom Procedures to Display and Hide a Status Bar on page 198.
Hide Procedure	Procedure called to hide the status bar. Leave the default unless you want to call a custom procedure. See Writing Custom Procedures to Display and Hide a Status Bar on page 198.
Screen Location	Click if you want to select the initial position of the status bar.

The figure below shows the dialog for selecting the initial position of the status bar:



To select the initial location of the status bar:

➔ Click on the check box in the position you want to select.

An `sc-smh-status-subscriber` is created on the status bar for each message server selected.

After you complete the dialog, the status bar template is displayed. You can rearrange the position of the message server indicators and the clock. You can also delete the clock.

When you have finished configuring the status bar, select the hide button in the upper right corner of the status bar template. If copies of the status bar exist, you will be prompted to delete these copies.

Writing Custom Procedures to Display and Hide a Status Bar

You can write your own procedures to customize what happens when a status bar is displayed or hidden. You enter the name of your custom procedures in the status bar configuration dialog.

To define a custom procedure to display a status bar:

- 1 Locate the status bar.
- 2 Select the name of the status bar and click Configure...
- 3 Click the square button labeled ... that is next to the edit box for the Display Procedure.

A dialog appears that asks for the name of the new procedure.

- 4 Enter the name you want to use for your custom procedure in the dialog, then click OK.
- 5 If you have finished configuring the rest of the status bar, click the OK button on the Configure Status Bar dialog.

The system creates the procedure you named and places it alongside the status bar definition.

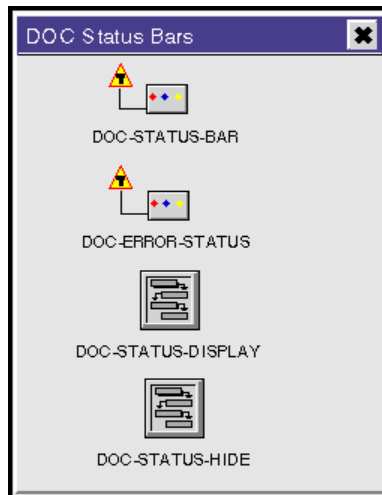
The dialog to Create a status bar is the same as the one to Configure an existing status bar so you can also define custom procedures when you create the status bar.

Use the same technique to create a custom procedure called when you hide the status bar.

To view and edit your custom procedure:

- 1 Locate the status bar.
- 2 Click Go To on the toolbar.

This takes you to the workspace where the status bar is defined. The status bar workspace from *doc_demo.kb* is shown in the figure below:



In *doc_demo* the procedures, *doc-status-display* and *doc-status-hide* were created for the status bar *doc-error-status*, using the technique described above.

Note If your procedure definitions overlap, drag them apart to view them clearly.

- 3 Select the definition of the procedure, then select **edit** to define the contents of the procedure.

The *doc_demo* custom procedures are shown below. Nothing has been added to these procedures. The system creates this code to provide the arguments and calls you must include in your custom procedures.

```
doc-status-display (source: class item, target: class item,
  wksp: class kb-workspace, win: class g2-window)
begin
  call sc-display-status (source, target, wksp, win);
end

doc-status-hide (wksp: class kb-workspace)
begin
end
```

The arguments passed to these procedures are:

Argument	Description
source	Status bar template you are configuring
target	Status bar template you are configuring
wksp	Workspace of the view of the status bar created when you select Status Bar from the Applications Objects menu.
win	G2-window of the view of the status bar.

You can see from the example that a new display procedure must call the Integrity procedure `sc-display-status` to display the status bar. You can add any other behavior in the procedure. Custom hide procedures do not need to call any Integrity procedures.

Defining Escalation Specifications

When a message is not acknowledged within a defined time period, Integrity can follow certain procedures that escalate the message status. You create an **escalation specification** object to define the procedures Integrity should follow. The procedures that the escalation specification defines are called **escalation procedures**.

You associate an escalation specification with messages containing a specified target, category, and/or priority. You can define five phases for the escalation specification. Each phase has a duration after which the next phase is invoked. When a new phase is invoked, it calls the escalation procedure for that phase and passes it the message as the argument.

The demonstration application contains an example of an escalation specification.

To see an example of an escalation specification in the demo application:

- 1 Load `doc_demo.kb`.
- 2 List the Escalation Specifications by using the Finder.
- 3 Click on Doc-Printer-Escalation and select the Configure... button.

The configuration of this escalation specification is displayed. Three phases are defined. Each phase calls the same procedure, `Doc-RaisePriority`. This

procedure increases the priority of the message every minute that elapses without the message being acknowledged by the user.

- 4** To see the escalation object work:
 - a** Choose **View > Domain Map**.
 - b** Navigate through the domain map until you select the object for the Houston site.
 - c** Select **View External Objects** from the Houston domain map.
 - d** Use the Finder to locate the Doc-Browser.
 - e** Select **Ext-printer-B1** and select **Noisy** from the menu.

This sends a **Noisy** message which targets **Printer-B1**. The escalation specification defined for the printer replaces the old message with a new one with a higher priority every minute that the message remains unacknowledged. You can see the effects of this escalation as the color of both the message in the browser and the color of the printer on the domain map changes every minute to match the priority of the message.

- 5** To view the procedure **Doc-RaisePriority**:
 - a** From the Explorer view, double click the “+” next to **Escalation Specifications**.
 - b** Select **doc-printer-escalation**, click **Go To**, then select **Escalation Procedures**.
 - c** Select the procedure definition icon and select **table**.

To define an escalation specification:

- 1** Click on the **Escalation Specification** block from the **Integrity** components palette.
- 2** Select the tab for the module where you want to place the escalation specification.
- 3** Select the item **Escalation Specifications**.

- Click Create to display the dialog shown below:

Phase	Delay Time	Escalation Procedure
1	0 minutes	
2	0 minutes	
3	0 minutes	
4	0 minutes	
5	0 minutes	

- Configure the dialog as described in the following headings.

Specifying the Target of an Escalation Specification

The target defines the domain objects associated with the escalation specification. The escalation specification applies to every message that targets the domain object you specify. You can specify the target by:

- The class of the target.
- The external name of the target. The specification for the external name can contain the wildcard "*" to specify a match for any characters or the wildcard "?" to match a single unspecified character.

Caution Using wildcards to define external names can cause a performance loss to the system. We strongly recommend that you do not use wildcards when the target type is an object class. This can significantly slow system performance.

The target specifications of the escalation specifications are initialized on start-up and on re-configuration.

Specifying the Category of an Escalation Specification

When you define an escalation specification, you can specify the category of the targets that you want to be linked to the escalation specification.

Note Because each domain object can only be linked to a single escalation specification, you cannot define more than one escalation specification for the same target even if the categories of the target are different. The category definition simply allows you to limit the application of the escalation specification of a particular target.

Specifying the Priority of an Escalation Specification

You assign each escalation specification a priority. Before a message is linked to an escalation specification, the priority of the message is checked with the priority of the escalation specification. The priority can:

- Match any priority, using Any Priority.
- Match a specific defined priority, using Priority =.
- Match any priorities less than or equal to a defined priority, using Priority <=.

You select the type of priority by checking using the radio buttons on the configuration dialog.

Specifying the Procedures Called in Escalation Specifications

An escalation specification can have from one to five phases, each of which is associated with a procedure. You can specify three types of information in the escalation procedure associated with each phase of an escalation specification:

- The name of a G2 procedure. This procedure is called when the escalation specification completes the phase associated with the procedure.
- The name of an OPAC procedure. This is a graphical procedure you create with the OPAC module of Integrity. The OPAC procedure is started when the escalation specification completes the phase associated with the OPAC procedure.
- The name of another escalation specification. After the message enters the escalation phase, a relationship is formed between the new message and the new escalation specification.

Timing the Invocation of Escalation Phases

Each escalation specification contains five phases. Each of these phases has a delay time attribute which determines the time at which the phase is invoked for a message related to the escalation specification. When a message is created, it has an attribute called **Creation-time**, which is set to the time the message was created. This time is set using G2 time, which represent the date and time in seconds measured from a preset starting point.

The phase delay times are added sequentially to the message creation time to create an array of times at which each phase is invoked. The time of the first phase is saved in the message attribute **Revisit-time**. The table below shows an example of how the delay times are used to create an array that defines the times at which each phase begins.

In this example, assume the message creation time was 12:02.

Phase	Delay Time	Revisit-Time
1	5 min.	12:07
2	7 min.	12:14
3	7 min.	12:21
4	7 min	12:28
5	5 min	12:32

Because there is overhead in testing to see if the revisit times have been reached for all the messages related to escalation specifications, the messages are tested only at an interval defined by the parameter **smh-gep-update-rate**. You set this parameter, using an initialization, as described in *Editing the Value of an Initialization* on page 80.

If you assume that last escalation specification update occurred at 12:00 and **smh-gep-timed-update** is set to update escalations every five minutes, the phases defined for the message shown in the table above are invoked as follows:

Test Time	Message Status
12:00	not yet created
12:05	not yet created
12:10	enter phase 1
12:15	enter phase 2
12:20	still in phase 2
12:25	enter phase 3
12:30	enter phase 4
12:35	enter phase 5

The message status is only updated if the update time is greater than or equal to the revisit-time of the message.

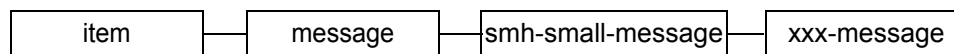
Note You can see from these tables that it is important to define an `smh-gep-update-rate` that is less than the delay times defined in the escalation specifications. If the test interval is called only every 30 minutes and escalation phases are defined every 5 minutes, the escalations will not be invoked as defined. On the other hand, a very low `smh-gep-update-rate` will cause a serious performance loss to the system.

Working with Messages

This section describes how to create, view, acknowledge and delete messages. It also describes message histories and how messages propagate through the domain map.

What is a Message?

In an Integrity application, a message is an object that stores information about an event. A message is stored in a message server and can be displayed on a message browser. The main message class in Integrity is an `smh-small-message`. When you create a new application, a new message class named `xxx-message`, where `xxx` is the name of the prefix you selected for your application, is automatically created. The hierarchy of an `xxx-message` is shown in the figure below:



A message is created by calling the procedure `smh-create-message`. This procedure defines the message attributes and assigns the message to a message server. The message server you select determines the class of the message created by `smh-create-message`. For information about specifying the message class for a message server, see *Defining Message Servers* on page 179.

The `smh-small-message` class has the attributes described in the table below:

Attribute	Description
<code>text</code>	Text of the message. Set by the argument <i>txt</i> in the procedure <code>smh-create-message</code> .
<code>smh-sender</code>	Domain object that is the sender of the message. Set by the argument <i>sender</i> in the procedure <code>smh-create-message</code> .
<code>smh-target</code>	Domain object that is the target of the message. Set by the argument <i>target</i> in the procedure <code>smh-create-message</code> .
<code>smh-message-category</code>	Category of the message. Set by the argument <i>category</i> in the procedure <code>smh-create-message</code> .
<code>smh-repetitions</code>	Number of repetitions of the message. When a message with the same target, sender and category as an existing message is created, this value is incremented in the existing message unless the message is created using the <code>-r</code> option.
<code>smh-creation-time</code>	Time the message is created shown in G2-time. Set by the argument <i>time-sent</i> in the procedure <code>smh-create-message</code> .
<code>smh-formatted-creation-time</code>	Time the message is created in the format MM/DD/YY HH:MM:SS.
<code>smh-revisit-time</code>	Time the current escalation routine targeting the message is invoked. See Timing the Invocation of Escalation Phases on page 203. This value is set by the values specified in from the Escalation Specification configuration dialog.
<code>smh-revisit-method</code>	Name of the current escalation routine. Set by the values specified in the Escalation Specification configuration dialog.

Attribute	Description
smh-deletion-time	Calculated from the <code>smh-creation-time</code> and the value of the argument <i>lifetime</i> specified in the procedure <code>smh-create-message</code> .
smh-priority	Priority of the message. Set by the argument <i>priority</i> in the procedure <code>smh-create-message</code> .
smh-acknowledgement-status	By default a message is created with the status of <code>unacknowledged</code> . If the <i>option -nack</i> is used in <code>smh-create-message</code> , the message is created with a status of <code>acknowledged</code> . The status of an <code>unacknowledged</code> message is updated when the message is acknowledged either by the user or a call to the procedure <code>smh-acknowledge-message</code> .
smh-acknowledger	Name of the window that sent the message acknowledgment. If it is the local window, the value is set to the current G2-mode.
smh-additional-text	A second text field added to the message. Set by the argument <i>additional-text</i> in the procedure <code>smh-create-message</code> .
smh-user-comment	Comment added to the message by the user. This is added by selecting a message in the browser then clicking the add comment button. See <i>Using the Browser to View and Interact with Messages</i> on page 210.

Creating a Message

You create a message by using either an Integrity procedure or, if you use the OPAC graphical language module, an OPAC graphical procedure. The call to create a message usually occurs in a completion routine or a reasoning routine after you have parsed an incoming event to determine the target, sender and category and have decided that a message should be created.

In the example shown below from `doc_demo`, the method `doc-out-of-service` is called whenever an event of the category “out-of-service” occurs. The target,

sender, category, and text of the event are passed to the method and a message is created and placed in the doc-message-server.

```
doc-out-of-service (target: class doc-object, sender: class doc-object,
  category: text, info: text)
msg: class doc-message;
begin
msg = call smh-create-message (doc-message-server, sender, target,
category, info, "", -1, 2, 300, false, sender, "-r");
end
```

The Integrity procedure to create a message is:

smh-create-message

```
(server: class smh-message-server, sender: class object,
target: class opfo-domain-object, category: text, txt: text, additional-text: text,
time-sent: float, priority: integer, lifetime: integer, show-display: truth-value,
win: class object, options: text)
-> message: class smh-small-message
```

Argument	Description
<i>server</i>	Message server to contain the message.
<i>sender</i>	Sender of the message.
<i>target</i>	Target of the message.
<i>category</i>	Message category.
<i>txt</i>	Value assigned as the text of the message
<i>additional-text</i>	Value assigned to the attribute additional-text of the message.
<i>time-sent</i>	Time the message is created. This is G2-time format. Pass in any negative number to set the time to the current time.
<i>priority</i>	Priority of the message, which is an integer.
<i>lifetime</i>	Number of seconds the message is maintained before being deleted. If a value of -1 is entered, the message will not be automatically deleted.
<i>show-display</i>	Either true or false . Currently not used.

Argument	Description
<i>win</i>	Any G2 window or object. Currently not used.
<i>options</i>	<p>-nohist - Do not maintain a history for the message. -nack - Set the message as an acknowledged message.</p> <p>Several options define how to handle a new message which is a duplicate of an existing message. Only one of these can be used at a time:</p> <p>-a - <i>txt</i> is appended to the main text of the existing message and <i>txt2</i> is appended to the value of additional-text of the existing message.</p> <p>-i - information regarding the Repetitions of the message is appended to the main text of the message and the repetitions counter attribute of the message is incremented. This is the default option.</p> <p>-r - the old message is deleted and replaced with the new message. Escalation procedures from the old message are copied to the new message. The text of the new message replaces the text of the old message.</p>

The call to `smh-create-message` returns the `smh-small-message` that is created.

Note When you call `smh-create-message`, you can use only one of the options `-a`, `-r` or `-i`.

For information on creating a message, using OPAC, see the *OPAC User's Guide*.

Maintaining Message Histories

Histories are an important component of the messaging system because they allow you to reason about and filter incoming events based on the time and frequency of identical events. This can prevent alarm flooding resulting from repeated events generated from a single fault.

When you create a message, the system searches to see if a history object already exists for the type of message. A history object which matches the *target*, *sender* and *category* arguments of the new message is considered an existing history. When there is an existing history, the *time-sent* of the new message is added to the end of the history list unless `-nohist` is included in the *options* argument of the call to create the message.

Note When you create a message with `-nohist` passed as one of the *options* arguments, no history objects are created or updated.

For information on how to search message histories see Querying Message Histories on page 228.

Using the Browser to View and Interact with Messages

You can display the browser by selecting the *browser name* from the Browser finder list, or from the Explorer tree, then selecting **View Object**. When you select a message in the browser, the buttons at the top of the browser become active.

To select browser messages:




- 1 If you want to select a single message, click on the message..
- 2 To select a group of messages, click on the first message in the group then hold down Shift and then click on the last message in the group.







To deselect browser messages:

- 1 Click on another message.
- 2 If only one message is shown on the browser, to deselect it (so that its status color is visible), hold down the Control-key and click on the message.

Tip To display the name of a browser button name, place the cursor over the browser button then hold down the mouse button.

Each of the Message browser toolbar options is described below:

Option	Description
	Acknowledge the selected messages.
	Delete the selected messages. Note: Only acknowledged messages can be deleted.
	Go To Sender of the message. A flashing arrow points to the item on the domain map that is the sender of the message.

Option	Description
	Go to Target of the message. A flashing arrow points to the item on the domain map that is the target of the message.
	View message details of the selected message. This button displays the values of the attributes of the message.
	Add Comment to the selected message. This button lets the user add a comment to the message.
	Time Sort the messages. To sort a group of messages by time, click on this button.
	Reverse the sort order of the messages. To sort a group of messages, click on the title of the column you want to use as the sort key.
	Lock or Unlock the view of the browser. This locks out all operations on messages and disables updating the browser with new messages.

The message subscribers define the message servers whose messages appear in the browser. Filters limit the browser to display only messages with the attributes defined by the filter.

To activate and deactivate filters and subscribers on the browser:

➔ Click on the filter or subscriber check box and select **sc-toggle** from the menu.

In Operator Mode, you click on a check box to select it.

The message server subscribers on the browser display the status summary of the messages in the corresponding message server. Each message server subscriber's icon has two concentric icon-regions displayed on the side of the check boxes. The inner circle icon displays the alarm status of the highest priority unacknowledged message in the specified message server. The outer concentric icon displays the alarm status of the highest priority acknowledged message in the specified message server. It provides at a glance an overview of the status of the messages in message servers even when the message server is not selected.

Messages can be sorted by the cell values contained in any column on the browser or by the creation time of the message. The sort order can be changed by selecting the Sort button on the button bar.

To sort the messages displayed on a browser by the value of a column:

- ➔ Click on the title bar of the column you want to use for the sort key. The order can be switched from ascending to descending by selecting the Reverse Sort Order button from the toolbar.

To sort the messages displayed on a browser by time:

- ➔ Click on the button with the clock icon that is in the tool bar. The order can be switched from ascending to descending by selecting the Reverse Sort Order button from the toolbar.

Note When you select a column or the Time Sort button to sort messages, the messages in the browser are sorted. If the automatic sort feature is enabled, all incoming messages are also sorted according to the criteria selected. For information on setting the sorting characteristics of the browser, see *Defining the Sorting Characteristics of the Browser* on page 185.

Acknowledging Messages

Every message you create has an attribute called **Acknowledgment-status**. This attribute keeps track of whether a message is acknowledged or unacknowledged. The **acknowledgment-status** of the message determines the color of the message and the **acknowledgement-region** color of the target object and the objects above the target in the containment hierarchy.

To acknowledge a message from a browser:

- 1 Display a browser that shows the message you want to acknowledge.
- 2 Click on the message.
- 3 Select the acknowledge button on the browser.

The button icon shows a check mark.

When a message is acknowledged on one browser, all views of the message on other browsers are also acknowledged.

To acknowledge a message programmatically:

- ➔ Use the procedure:
`smh-acknowledgment-proc` (*Message*: class smh-small-message, *win*: class object)

Deleting Messages

Every message created in an Integrity application should eventually be deleted. If messages are not deleted, the message information base will continue to grow until its size eventually becomes a bottleneck to the system. Messages are deleted several ways:

- The user can delete a message from a browser.
- Specify the *lifetime* argument of the `smh-create-message` procedure when you create the message. After the specified time, the message is automatically deleted.
- Delete a message programmatically using an Integrity procedure. These are described in the section below.

When a message is deleted from the browser, the system can display a dialog verifying that the message should be deleted. This feature can be enabled or disabled by setting the value of the initialization item `opcsrui-confirm-message-deletion` to true or false.

To delete a message from the browser:

- 1 Display a browser which shows the message you want to delete.
- 2 Click on the message.
- 3 If the message has not been acknowledged, click on the acknowledge button to acknowledge it.
- 4 Select the delete button on the browser. The button icon shows an X.

When a message is deleted on one browser, all views of the message on other browsers are also deleted.

To delete a specific message programmatically:

→ `smh-delete-message`

(*server*: class smh-message-server, *sender*: class object,
target: class opfo-domain-object, *category*: text, *win*: class object,
options: text)

Argument	Description
<i>server</i>	Message server to contain the message.
<i>sender</i>	Sender of the message.
<i>target</i>	Target of the message.
<i>category</i>	Message category.

Argument	Description
<i>win</i>	Any G2 window or object.
<i>options</i>	-all - Deletes the message in all message servers irrespective of the message server argument specified.

This procedure is used to delete a specified message with the designated sender, target and category either in one message server, or in all message servers.

To delete all messages within a specified message server:

→ smh-server-delete-all-messages
 (*server*: class: smh-message-server)

To delete an entire message server and all its messages:

→ smh-delete-server
 (*server*: class: smh-message-server)

Reading and Writing Messages to a File

You can write the contents of a message server out to an external message information base text file, and read the contents of an message base information base text file back into a message server.

To write the contents of a message server to a message information base text file:

→ smh-write-server-mib-to-file
 (*server*: class smh-message-server, *filename*: text)

Where *server* is the handle to a message server defined in the application and *filename* is a legal file name.

To read a message information base text file into a message server:

→ smh-read-server-mib-from-file
 (*server*: class smh-message-server, *filename*: text)

Where *server* is the handle to a message server defined in the application and *filename* is a legal file name.

smh-read-server-mib-from-file requires that the message information base be stored in the file in the same format in which smh-write-server-mib-to-file writes the message information base. Both of these procedures should be used in conjunction with each other.

Message Alarm Propagation

When you create a message, the message sends an alarm to the domain object targeted by the message. From the target domain object, the alarm propagates up through the containment hierarchy. When an alarm reaches a domain object, it compares the `_opfo-highest-message-priority` to the priority of the message that generated the alarm. If the `_opfo-highest-message-priority` is lower than the message priority, it is reset to the new high value. Next, the alarm propagation routine checks the acknowledgment status of the message generating the alarm. If the status is unacknowledged, the alarm sets the `_opfo-acknowledgement-status` of each domain object in the containment hierarchy to unacknowledged.

The values of `_opfo-highest-message-priority` and `_opfo-acknowledgement-status` define the colors used to display the `alarm-region` and the `acknowledge-region` of the domain object. You map specific colors to priority values using Initializations as described below in Setting Priority and Acknowledgment Colors.

Setting Priority and Acknowledgment Colors

When you create a message, you assign the message a priority. A priority is an integer value. The highest priority message has a priority value of 1. Each priority has an associated background color, called the priority color, and text color. When a browser displays a message, it uses the priority to set the background and text color of the message.

The priority color also affects the color of the alarm regions of domain objects. The highest priority message targeting a domain object defines the color used for the `alarm-region` icon region of the object.

The priority colors are defined in the array `opfom-priority-alarm-colors`. The text colors are defined in the array `opfom-priority-alarm-text-colors`. If a message has a priority that does not have a color assigned to it, the color defined by the `opfom-default-priority-color` object is used as the message background color on the browser and the domain objects.

Colors are also used to show the acknowledgment status of objects on the domain map. If an object is targeted by an unacknowledged message, the color of the `acknowledgement-region` of its icon changes to the color defined by the `opfom-unacknowledged-color` object. If no unacknowledged messages target the object, its `acknowledgement-region` is displayed in the color specified by the `opfom-acknowledged-color` object.

To edit an object that defines a priority color, you edit the Initializations that targets the object. For details on how to edit initializations, see Editing the Value of an Initialization on page 80.

The table below summarizes the color assignments used to set the priority colors and the initialization targets you can edit to change them. For detail about these initializations, see the *Integrity Reference Manual*.

Initialization Target	Color Defined
opfom-priority-alarm-colors	Background colors used for messages and domain object alarm-region .
opfom-priority-alarm-text-colors	Text colors used for messages and domain object alarm-region .
opfom-default-priority-color	Color used for the background and domain object alarm-region of a priority not defined in opfom-priority-alarm-color .
opfom-unacknowledged-color	Color used for the acknowledgement-region of a domain object targeted by an unacknowledged message.
opfom-acknowledged-color	Color used for the acknowledgement-region of a domain object not targeted by an unacknowledged message.

Setting Default Message Priorities

Domain map objects have an attribute named `_opfom-highest-message-priority` which keeps track of the highest priority of the messages targeting the object. The value of this attribute determines the color used to display the **alarm-region** of the object's icon.

When the domain map is initialized at system startup, a default value of 9999 is assigned to this attribute, using an initialization with the target **opfom-default-initial-priority**. Because this priority is not mapped to a color, the **opfom-default-priority-color** is used, which is transparent.

The `_opfom-highest-message-priority` of an object changes when a higher priority message targets the object. Keep in mind that the highest priority has a value of 1, so the lower the numeric value, the higher the priority. The priority of the message determines the alarm color of the target object and the objects above the target in the workspace containment hierarchy. When the last message targeting an object is deleted, the value of the object's `_opfom-highest-message-priority` attribute is reset to the value of the integer parameter **opfom-default-priority-on-**

`del-of-last-message` or `opfom-default-initial-priority` depending on whether the parameter, `opfom-revert-priority-to-initial` (a boolean) is set to true or false.

In *Integrity* the default values for `opfom-default-priority-on-del-of-last-message`, `opfom-default-initial-priority`, and `opfom-revert-priority-to-initial` are set to 6, 9999 and true respectively. The values of all these parameters can be changed by editing their initializations.

The initial priority of an object changes when a higher level priority message targets the object. When the last message targeting an object is deleted, the value of the object's `_opfo-highest-message-priority` attribute is reset to the value of the parameter `opfom-default-priority-on-delete-of-last-message`. A system initialization defines this default to equal 6. Another parameter, `opfom-revert-priority-to-initial` is a boolean can be set to true to override the `opfom-default-priority-on-delete-of-last-message` with the value of the `opfom-default-initial-priority`.

To edit an object that defines a priority color, you edit the Initializations that targets the object. For details on how to edit initializations, see *Editing the Value of an Initialization* on page 80.

The table below summarizes the initializations defined that affect the priorities set after the last message targeting an object is deleted. For details of these initializations see the *Integrity Reference Manual*.

Initialization Target	Priority value set
<code>opfom-default-initial-priority</code>	Initial priority.
<code>opfom-default-priority-on-delete-of-last-message</code>	Priority after all messages against target are deleted.
<code>opfom-revert-priority-to-initial</code>	Revert to initial priority or default priority after all messages against a target are deleted. This results in the domain objects color being set to the value of <code>opfom-default-priority-color</code> .

Logging Messages and Events

Integrity includes a logging facility that lets you create and manage log files. You can use a log manager to log all the actions performed on the messages in a message server or to log events which never became messages.

You can configure a logging manager to enable or disable logging and to automatically delete any empty log files.

The logging facility is used to log error messages as described below in Logging System Errors on page 223.

Creating a Logging Manager

Before you can log any events you need to create a log manager.

To create a logging manager:

- 1 Click on the logging manager block from the Integrity components palette.
- 2 Select the module where you want to place the logging manager, then click OK.
- 3 Complete the configuration dialog shown below:

The screenshot shows the 'Create Logging Manager' dialog box with the following settings:

- Name: NONE
- Logging enabled: True
- Directory: (empty)
- File name template: log_*.txt
- File name generator: GLF-DEFAULT-FILE-NAME-GENERATO ...
- File header writer: GLF-DEFAULT-LOG-FILE-HEADER-WR ...
- Time interval to open new log file: 88400 seconds
- Maximum file size in bytes: 100000
- Log file scheduler: GLF-DEFAULT-LOG-FILE-SCHEDULER
- Automatically delete empty log files: True
- Current log file: (empty)

Configuration Item	Description
Name	Name of the logging manager.
Logging Enabled	Whether the logging manager is enabled or disabled.
Directory	Directory where log files are written.

Configuration Item	Description
File Name Template	Log file names are generated based on this template. In the default template, * is replaced by the digits of the year, month, day, hour and minute at which the log file is created.
File Name Generator	Name of the procedure to generates the file names. You can specify a custom procedure.
File Header Writer	Name of the procedure to generate the file header. You can specify a custom procedure.
Time Interval to Open New File	Length of time interval in seconds when a new log file is opened.
Maximum File Size in Bytes	Maximum file size of the log file. When the size is reached, a new log file is opened.
Log File Scheduler	Name of the procedure used to schedule log file creation. The default can be replaced with a custom procedure.
Automatically Delete Empty Log Files	Select whether empty log files are automatically deleted.

Logging Messages

You can configure the Integrity message system to automatically log all messages sent to a specified message server. The creation of the messages themselves and all actions performed on the messages are logged. The actions logged include:

- Creation
- Deletions
- Acknowledgments
- Addition of user comments
- Changes to the text of a message

To log all messages in a particular message server:

- 1 Create a Logging Manager as described in [Creating a Logging Manager](#) on page 218.
Be sure that the Logging Manager is enabled.
- 2 Navigate to Message Servers, double click on the “+” to view the message servers defined and select the message server you want to log.
- 3 Select the Configure... button.
- 4 Choose a the Logging Manager from the scroll area on the message server configuration dialog.

Logging Events Programmatically

You can use the logging facility to log any kind of information to a log file. For example, events can come into the system for which you do not create messages but which you want to send to a log file.

To write text to a log file:

→ `glf-write-to-log-file`
(*log-name*: class glf-logging-manager, *info*: text)

log-name is the name of the Logging Manager. The Logging Manager writes the *info* passed to its current log file.

Defining Closing Times for a Log File

You can use several different methods to specify when a log file should be closed and a new file opened. These include:

- Defining a time interval in the logging manager dialog item Time Interval to Open New File. This method is implemented in the default Log File Scheduler Procedure.
- Setting a list of daily closing times using the API procedure `glf-set-fixed-log-closing-times`. See the *Integrity Reference Manual* for a description of this procedure.
- Defining a maximum size for the log file set in the logging dialog item Maximum File Size in Bytes
- Writing a custom scheduling procedure and naming it in the logging dialog item Log File Scheduler.

Customizing the Logging Manager

Three attributes of the logging manager configuration dialog define procedures used to customize the behavior of the logging manager. All of the following custom procedure take two arguments: (*log*: class `glf-logging-manager`, *client*: class object).

- File Name Generator is the name of the procedure to generate the names of the log files. The default procedure uses the File Name Template and the File Directory to name the log files. Any custom procedure defined should return a text which will be the name of the log file.
- File Header Writer writes out a header text for the log files. The default writes out time stamp information. Any custom procedure defined should write a header text to the log file of Log.
- Log File Scheduler defines a procedure that times the opening and closing of log files. The default closes the file after a time interval defined by Time Interval to Open New File. Any custom procedure defined should return an integer time-interval (in seconds) after which a new file will be opened.

Error Handling

The error handling routines in Integrity use the Integrity message system. When an error occurs, a default system procedure creates an Integrity message for each error and sends the message to the message server `smh-error-server`. You configure the default browser created as part of a new application to display the error messages.

You define the behavior of the default error handler using initialization objects. The table below lists each initialization used as part of the error handling system and describes how it is used:

Initialization Item	Error handling definition
<code>smh-system-error-server</code>	Message Server that receives error messages.
<code>devu-error-handler-proc</code>	Name of the error-handling procedure.
<code>devu-error-lifetime</code>	Lifetime of an error message.
<code>devu-high-priority</code>	Value assigned to a high-priority error.
<code>devu-medium-priority</code>	Value assigned to a medium-priority error.

Initialization Item	Error handling definition
devu-low-priority	Value assigned to a low-priority error.
devu-system-category	Value assigned to the message category of a system error.

You can change any of the default values for these items by defining an initialization item in your application module. Setting initializations is described in *Editing the Value of an Initialization* on page 80. For a detailed description of these initializations, see the *Integrity Reference Manual*.

Creating a New Error Handling Procedure

If you decide to define a new error handling procedure by changing the initialization value of `devu-error-handler-proc`, you must make the arguments for your new error handler match the arguments passed to the default procedure, `smh-send-error-message`. These arguments are described below:

`smh-send-error-message`

(*target*: class item, *sender*: class: item, *error-type*: text, *priority*: integer, *error-name*: symbol, *error-text*: text, *error-lifetime*: integer)

Argument	Description
<i>target</i>	Item causing the error.
<i>sender</i>	Sender of the error.
<i>error-category</i>	Category of the error.
<i>priority</i>	Priority of the error.
<i>error-name</i>	Name of the error.
<i>error-text</i>	Text describing the error.
<i>error-lifetime</i>	Time interval in seconds before the error is deleted.

Logging System Errors

The default message server used to receive error messages is named `smh-error-server`. This message server is defined to log all messages, using a Log Manager named `smh-error-log`.

To disable or enable error logging:

- 1 Select Tools > Inspect from the Integrity main menu.
- 2 Type *go to smh-error-log*.
- 3 Click on End.
- 4 Select `smh-error-log` to display its menu.
- 5 Select `turn on logging` to enable logging. Select `turn off logging` to disable logging.

To disable or enable logging programmatically:

- `glf-enable-logging`
(*obj*: class `glf-logging-manager`, *win*: class object)
- `glf-disable-logging`
(*obj*: class `glf-logging-manager`, *win*: class object)

These procedures are described in the reference section.

Creating User Defined Effects

Integrity defines a set of parameters that contains the names of procedures called when messages are created, acknowledged, deleted, or modified. These parameters are defined in the table below:

Action on message	Calls Procedure Contained in Parameter
Create a new message	<code>smh-user-message-creation-proc</code>
Delete a message	<code>smh-user-message-deletion-proc</code>
Acknowledge a message	<code>smh-user-message-acknowledgement-proc</code>
Add a comment to a message	<code>smh-user-message-comment-proc</code>
Change the text of a message	<code>smh-user-message-text-change-proc</code>

When you start your application, the system initializes the default values of these parameters, using Initializations. You can modify the behavior of the message system by editing the Initializations to change the names of the procedures called by these parameters.

To add a custom procedure when a message is created, modified, or deleted:

- 1 Create the new procedure.

Procedures called using `smh--user-message-text-change-proc` must accept the arguments:

(msg: class smh-transient-message, old-text: text, win: object)

All other procedures called by these initializations should accept the arguments:

(msg: class smh-transient-message, win: object)

- 2 Edit the Initialization that targets the parameter you want to change.

Editing initialization objects is described in *Editing the Value of an Initialization* on page 80. For a detailed description of the initializations, see the *Integrity Reference Manual*.

The `doc_demo` sample application provides an example of the use of a custom procedure. The message class used in `doc_demo`, `doc-message` defines the new attribute `city-of-target`. A procedure named `doc-set-city-value` is assigned to the system initialization `smh-user-message-creation-proc`. The system calls this procedure every time it creates a new message. The procedure uses information from the domain map to set the value of `city-of-target`. You can view this procedure by selecting the Application Workspaces > Miscellaneous.

The default procedures defined in Integrity add logging behavior to the message system, as described in *Logging Messages* on page 219. If you call a custom procedure, and you want to retain the logging functionality, you must call the original default logging routines in your custom procedures. The default values for these procedures are defined in the alphabetic listing of initializations in the *Integrity Reference Manual*.

Reasoning About Events

Describes how you use message histories and the domain map to reason about incoming events in Integrity.

Introduction	225
Examples of Reasoning Routines	226
Creating Reasoning Routines	227
Searching for Related Messages	227
Querying Message Histories	228
Filtering Messages and Events	229
Implementing Alarm Thresholding	232
Correlating Events	234
Diagnosing Faults	235
Automating Recovery and Preventing Faults	235



Introduction

Integrity provides a rich environment for reasoning about events generated by a group of external objects. Events coming into an Integrity application are related to the objects in the domain map to which the event refers. The domain map objects are related to history objects, which keep track of the message histories of each object. The histories let you reason about the occurrence and timing of related messages. The domain map itself contains information about connectivity

and containment relationships among the objects. These relationships make it possible to reason using the occurrence of messages to objects related to the current object.

You reason about incoming events in order to:

- Filter out events containing unnecessary or redundant information.
- Perform alarm thresholding to add information to events based on the number and time of events received.
- Correlate events to determine a likely source of a fault.
- Diagnose symptoms of a problem.
- Monitor the progress of a process.
- Perform automated recovery.
- Prevent faults from occurring.

This chapter discusses how an Integrity application can use the domain map and message structure to reason about incoming events.

Examples of Reasoning Routines

Many of the examples used in this chapter are shown in the *doc_demo.kb* sample application. It can be helpful to load this application and try some of the examples to see some of the different types of reasoning routines described.

To view reasoning routines in doc_demo:

- 1 Load *doc_demo*.
- 2 Choose View > Domain Map from the Integrity main menu.
- 3 Click on the Gensym logo, then select go to subworkspace from the menu.
- 4 Click on Houston, then select go to subworkspace from the menu.
- 5 Click on View External Objects.

This shows a simulated set of external objects, which are the objects modeled on the Houston workspace of the domain map.

- 6 From the DOC Application Objects workspace, select Message browsers > Doc-Browser from the Integrity main menu.
- 7 Press CTRL+S with the mouse cursor over each of the windows to shrink them so you can view several windows at one time.
- 8 Using the workspace finder, locate and view the Docdemo-top-level workspace, select Application Objects, then select Reasoning Routines.

This workspace shows the definitions of the Reasoning Routines used in the examples.

Creating Reasoning Routines

When you use the startup KB for your package to create a new application, the system creates a workspace for you to place reasoning routines. Reasoning routines are subclasses are methods or procedures. The user interface lets you create and navigate to methods that are reasoning routines.

To create a new reasoning routine:

- 1 Open the DOC Reasoning Routines workspace.
- 2 From the Core G2 Objects > Definition Objects palette, select Method declaration, method and/or procedures you need for your reasoning routine.
- 3 Drag and drop the blocks to the workspace.
- 4 Right-click on a block, then select properties. Edit the fields to configure the block.
- 5 If you are creating a method, enter the name of the class to which the method applies. You can select the button displaying the symbol of the triangle to view a list of the classes defined in the application.
- 6 Close the Properties box to save the configuration.

This creates an object for the method or procedure you define and places it on the Completion Routines workspace. When you create a method, the system automatically defines a method declaration as well as the class definition for the method specified.

To define the reasoning method or procedure:

- 1 Select the name of the method or procedure from the Finder.
- 2 Select the Configure button.
- 3 Type in the method or procedure in the edit window.

Searching for Related Messages

Reasoning routines often make use of information about the previous messages involving a specified target, sender, category or a combination of these attributes.

Several procedures are provided that let you search the message information base for existing messages that match certain specifications. Each of these procedures places all messages that match the specified criteria on a list passed as an argument to the procedure. These procedures include:

- `smh-get-messages-about` searches for all messages with a specified target.
- `smh-get-messages-sent-by` searches for all messages sent by a specified sender.
- `smh-get-messages-in-server` searches for all the messages in the specified server.
- `smh_message-query` searches for all messages with a specified target, sender and category.

For information on these procedures see the API reference in the *Integrity Utilities Guide*.

Querying Message Histories

The procedures described above let you directly search the message information base. Another way you can obtain information about previous messages to use in reasoning routines is to query the message histories.

The Integrity messaging system maintains a history for each message with a unique target, category, and sender. You can query these message histories to return a list that contains the times a message of a specific description was created.

To query a message history call the procedure:

→ `smh-get-message-history`
(target: item, sender: item-or-value, category: text,
message-category-starting-position: integer, timestamp-now: float,
match-time-interval-seconds: float, time-stamps-list: class float-list)

Argument	Description
<i>target</i>	Target of the message.
<i>sender</i>	The sender of the message. The symbol <code>no-sender</code> can be sent to match all messages with the specified target without regard to category.
<i>category</i>	Message category. This text can contain the wildcard symbols "*" and "?". The symbol "*" is matched by any number of arbitrary characters. The symbol "?" is matched by exactly one arbitrary character.

Argument	Description
<i>message-category-starting-position</i>	Starting position in the category from which wildcard symbols used in the <i>category</i> argument are matched.
<i>timestamp-now</i>	Time from which you want to search backwards for messages. Entering -1 selects the current time.
<i>match-time-interval-seconds</i>	Number of seconds back from the <i>timestamp-now</i> time you want to search for messages.
<i>time-stamps-list</i>	Empty list provided to hold the results of the history query.

This procedure searches for a history related to the specified *target*, *sender* and *category*. It returns a list of all the times between *timestamp-now* and (*timestamp-now*) - (*match-time-interval-seconds*) that messages were received for that particular history.

The sections below provide examples of how you can use message histories in different types of reasoning routines.

Filtering Messages and Events

In a large network of external objects, unnecessary events often occur for a wide variety of reasons. These events can congest the system and make it difficult for operators to sort out and respond to the real events that need attention. Filtering is done on several different levels in an Integrity application:

- The filtering of duplicate messages automatically done by the Integrity message system.
- Filtering routines called in response to a particular target, sender or category of an event. These routines can filter events based on past events and on relationships among the domain objects.
- Filtering done in the external bridge to prevent events from entering the system. This filtering is done without regard to any previous events.

Filtering Duplicate Messages

When an Integrity message is created, the system checks the existing messages to see if a message with the same target, sender, and category already exists. If it does, a duplicate message is not created. When you create a message, using `smh-create-message`, the *options* argument determines how Integrity handles a new

message with the same target, sender and category as an existing message. These options are summarized in the table below:

Options Argument	Description
-i	Add information about the number of messages to the text of the message. This is the system default.
-a	Append the text of the new message to the old text.
-r	Replace the old message.

When you use the -i option to create a duplicate message, instead of creating a new message, the message system adds the new information to the text of the existing message and adds the time of the new message to the end of the list maintained by the message history. If -nohist is added to the *options* argument, the time is not added to the history list.

Since duplicate messages are not created or displayed, the message system automatically filters duplicate messages.

Filtering Based on Past Events

The filtering done automatically by the message system only filters duplicate messages. You might also want to filter events before they become messages because of their relationships to past messages. These relationships can involve messages targeting the same object or messages targeting related domain objects.

Filtering Events that Occur in Pairs

Some things that happen to an external object can result in not one but a pair of events. For example, the failure of a piece of equipment can result in an event of the category Out-of-Service which is always followed by an event of the category Connection-Restored. If you know that these event relationships exist, you can write a filtering routine that responds to the Connection-Restored by immediately deleting the Out-of-Service event instead of by displaying the Connection-Restored message. This filters out an unnecessary message and initiates the response appropriate to the message.

The `doc_demo` application shows an example of this type of filtering.

To view an example of filtering paired events:

- 1 Load `doc_demo` and display the External Object Simulation, Doc-Browser, and Houston workspace as described in the beginning of this chapter.
- 2 Click on any object in the right hand column on the External Object Simulation workspace and Select **Out of service** from the menu.

The Out of Service message appears on the browser and displays an alarm on the target object on the Houston section of the domain map.

- 3 Select the same object you just selected but this time send a **Connection Restored** event.

The Out of Service message and alarms are deleted.

In this example, a reasoning method is called whenever any event occurs with the category **Connection Restored**. Instead of creating a new message, this routine deletes the existing Out of Service message targeting the object.

Filtering Events Based on Domain Relationships

You can do filtering based on past messages received by objects related to the target of an event. For example, you might have a situation with a external object, `BigDevice` which, after a certain type of failure, generates an Out-of-service event. As a side effect of this failure, a secondary device, `SmallDevice`, might begin to generate its own Failure events at regular intervals whenever `BigDevice` is out of service. To eliminate these messages, you would create a reasoning routine called by the completion routine whenever an event is received by `SmallDevice`. Before creating a message for a Failure event coming from `SmallDevice`, you test to see if the event could be a result of an out of service from `BigDevice`. For example,

```
smh-get-message-history (BigDevice, BigDevice, "Out-of-Service", 1, -1, 300,
bad-list)
smh-get-message-history (BigDevice, BigDevice, "In-Service", 1, -1, 300,
good-list)
if the last item of good-list > the last item of bad-list then {failure message is
not the result of BigDevice, post new message for SmallDevice}
```

In this example, you check to see if an Out-of-Service message has been posted against `BigDevice`. If so, you test to see if an In-Service message has been posted to clear the out of service condition. As long as the `BigDevice` remains out of service, the Failure event from `SmallDevice` is ignored. You can filter the event by discarding it, logging it and then discarding it, or by creating a message and then immediately deleting the message. You might create and then immediately discard a message to maintain history of the message event.

Filtering Events in the Bridge

Some events enter an Integrity application through a GSI bridge. You can add event filtering directly into the external and/or the internal bridge. Generally, the closer to the event the filtering occurs, the more efficient is the throughput of the system. Filtering in the bridge usually involves writing code to recognize certain events that are not necessary to pass to the Integrity application. These events are never passed into the internal bridge.

It might be more convenient to make an initial event filter in the G2 side of the bridge rather than in the external bridge. Procedures written in G2 are easier to develop and maintain; however, there is a performance premium to pay in moving the event further along into the system. Event filtering at this level usually consists of a very basic procedure of discarding the event based on one or more of its target, sender, and category attributes, or one of the values of one of the parameters passed with the event.

In the `doc_demo` sample application, a simple filter is included in the `doc-simulated-external-bridge` procedure. This can be viewed by selecting Supporting Procedures from the External Object Simulation Workspace. You would write a real external bridge procedure by using the GSI library routines. The simulation simply shows that it is the function of the external bridge to begin parsing and decoding the incoming events. This example implements a simple filter for any events that are received from a serial card. Whenever the `sender-name` matches the name of a serial card, the event is logged and then discarded.

Implementing Alarm Thresholding

The occurrence of many duplicate messages against an object in a certain time period can signify an event that needs particular attention. A reasoning routine can be designed to recognize this scenario and to create a special message to inform the operator of the situation. This is called an **alarm threshold**. An alarm threshold defines a number of messages and a time period. When the messages sent with the same target, sender and category exceed this threshold, you create a new message or increase the priority of the message.

For example, in the `doc_demo` application, a special warning is sent whenever three “Noisy” messages are sent against a target within a 5 minute period. Whenever an event with the Noisy category is received, `doc-reasoning-routine` calls the method `doc-noise-handler`, which is shown next:

```
doc-noise-handler (target: class doc-managed-object, sender: class doc-
object,category: text, info: text)
n: float;

begin

    call smh-create-message (doc-message-server, sender, target, category,
info, "",the current time,4, 999, false, sender, "-a");
    call doc-clean-list (doc-hist-list1);
    call smh-get-message-history (target, the symbol any-sender, "Noisy" , 1,
the current time, 300, doc-hist-list1);
    if the number of elements in doc-hist-list1 > 3 then call smh-create-
message (doc-message-server, sender, target, "Serious-noise", "There is
a serious noise problem with [the opfo-external-name of target]", "", -1, 1,
99999, false, target, "-a");

end
```

`doc-noise-handler` creates a message to represent the event, calls a procedure that empties a list to hold the results of the history search, and calls `smh-get-history` to find how many messages have targeted the object of the new event in the past five minutes. If there are more than three such messages, `doc-noise-handler` sends a new message that warns the operator that a serious problem with noise exists for this object.

To view an alarm threshold in `doc_demo`:

- 1 Click on any object in the right hand column of the External Object Simulation workspace.
- 2 Select Noisy from the list of events.

The Noisy message appears on the browser and displays an alarm on the target object on the Houston section of the domain map.

- 3 Select the same object and send three more Noisy events.

After the fourth event, a new message with a higher priority is displayed. This message informs the operator that there is a serious noise problem.

Correlating Events

The message histories and the domain object relationships provide valuable information about the state of the external objects. When you combine information about the state of several related devices, you can sometimes infer that there can be a problem with a higher-level device. This is called **alarm correlation**.

For example, in `doc_demo` three tape drive devices are defined on the Houston workspace of the domain map. These drives are all connected to a common object, `computer-H1`. In `doc_demo` a special reasoning method handles Out-of-Service events on the `doc-tape-drive` class.

When an Out-of-Service event occurs targeting a tape drive, `doc-tape-drive::doc-out-of-service` checks the message histories of all the tape drives for Out of Service events. If all the tape drives have had an Out of Service event posted in the last five minutes, `doc-tape-drive::doc-out-of-service` creates a message in the message server `doc-test-server` to notify the operator that a problem might exist with `computer-H1`. The application defines `doc-test-server` to hold messages generated by reasoning routines instead of external events.

To view an example of alarm correlation in `doc_demo`:

- 1 Using the finder, locate and view the Doc-Browser.

Be sure that the check box in front of the message server `doc-test-server` is checked.

- 2 Select `Ext-Tape-Drive1` on the External Object Simulation workspace and Select `Out of Service` from the menu.

The Out of Service message appears on the browser and displays an alarm on the target object on the Houston section of the domain map.

- 3 Repeat steps 1 and 2 for the objects `Ext-Tape-Drive2` and `Ext-Tape-Drive3`.

After each tape drive has posted an Out of Service event, a new message is displayed which warns that `computer-H1` might be out of order.

The method that implements this correlation routine, `doc-tape-drive::doc-out-of-service`, is on the Reasoning Routines workspace in `doc_demo`.

Diagnosing Faults

Alarm correlation, described above, is one method of diagnosing the source of faults based on relationships among domain objects. Another way you can diagnose faults is to use the information provided in an event to initiate a query to uncover other relevant information needed to diagnose the source of a fault.

For example, assume you are managing a file server. One of the functions of this server is to save daily log files that only need to be maintained for a certain period of time. If an Out of Disk Space event is received from this server, it can be displayed as a message to the operator. However, to make the message more useful, you can create a reasoning routine that not only accepts the incoming event but also finds the source of the problem.

In this example, the reasoning routine would initiate a request to the actual file server to receive a list of the files on the server. The procedure receiving the requested information would analyze the new information to determine the source of the problem and tell the operator which type of files are causing the problem.

To initiate requests from external objects, you must define a bridge process, which handles the communication from the Integrity application to the external objects. Chapter 6, Handling Events, describes bridges that allow external objects to send unsolicited events to an Integrity application. You set up the bridge to send information from Integrity to the external objects in a similar fashion. An internal part resides within the Integrity application that has links to calls in the external bridge. This external bridge is linked to the managed equipment and passes messages to them. The bridges must be individualized to the protocols defined by the types of equipment you manage.

Automating Recovery and Preventing Faults

The example in Diagnosing Faults on page 235 describes how you might query an external device to determine the cause of an error. In that example, the diagnosis was presented to the operator for action. In some cases you can go a step further and automate the action needed to recover from a fault. Because an Integrity application can communicate directly with the external equipment it manages, once the cause of a fault is discovered, using the reasoning routines in the application, you can define a recovery procedure to automatically recover.

In the example of the disk drive overloaded with out-of-date log files, when an alarm is received, a reasoning routine checks to see if the log files are a possible source of the out-of-disk space problem. In the diagnostics example, the system adds information to the alarm by discovering the cause of the problem. Instead of simply informing the operator of the problem, the system can immediately take the necessary actions, which is to delete the out-of-date files.

In complicated situations, you can build the knowledge of skillful operators into the system to create an expert system, which queries both the system and the operator to decide or suggest the best possible course of action. This insures that there is a uniform procedure to respond to certain types of alarms and makes the knowledge of the most skilled operators available to those that are less experienced.

In situations where reliability of a component is critical, an Integrity application can monitor the component to prevent faults from occurring. For example, Integrity can periodically query system components to see if they are reaching an overloaded state and you can design procedures to relieve the overload before a failure occurs.

The G2 language is designed for automating and monitoring complex real-time systems of many different kinds. Since an Integrity application has full access to G2, you can seamlessly integrate the functionality of fault prevention and fault response.

@ 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

Numerics

- 180 menu choice
 - Layout menu
- 90 Clockwise menu choice
 - Layout menu
- 90 Counterclockwise menu choice
 - Layout menu

A

- About Integrity menu choice
- Access Tables menu choice
- Acknowledge Messages Upon Selection
 - attribute
- acknowledgment region
 - setting colors of
- Address field
- adjusting
 - micro position of objects
 - order of objects
- Administrator mode
 - configuring user preferences for
 - description of
 - Tools menu
- alarm propagation
- alarm thresholding
- Align or Distribute menu choice
 - Layout menu
- application
 - steps for building
- applications
 - interacting with objects in
 - navigating
- attributes
 - adding to a subclass
 - displaying

B

- Back menu choice
 - Go menu
- Background Color attribute

- background images, loading
- Beep Enabled attribute
- borders, adjusting workspace
- bridge
 - definition of
 - external
 - filtering in
 - parsing in
- Bring to Front menu choice
 - Layout menu
- browser
 - arranging items on template
 - configuring
 - overview
 - subscribers
 - configuring columns
 - creating
 - custom display and hide procedures for
 - defining
 - defining filters
 - defining sorting characteristics
 - displaying from a custom menu
 - using with messages
- Bundles

C

- cascade menu item
 - creating
- category
 - definition of
- Charts menu choice
- choice menu item
 - creating
- class
 - components of
 - creating a hierarchy
 - definition of
 - importing definition
 - message
 - object hierarchy
- client
 - connecting

- directly to server
 - to a specific server
 - disconnecting
- Clone menu choice
- Edit menu
- Close menu choice
 - exiting client, using
- File menu
- color
 - setting priority of
- colors
 - configuring
 - for workspaces
 - editing for objects
- Colors menu choice
- Edit menu
- completion routine
 - creating
 - definition of
- configuring
 - browser
 - browser columns
 - menu bar template
 - message server
- connection
 - adding stubs for
 - creating a configuration object for
 - creating icons for
 - creating using a stub
 - definition of
- connection post
 - creating
 - definition of
- connection stubs
 - creating
- consistent modularization
- containment object
 - definition of
- containment relationship
 - definition of
- creating
 - browser
 - cascade menu item
 - choice menu item
 - completion routines
 - connection icons
 - connection posts
 - connection stubs
 - domain map subworkspace
 - domain object
 - initialization

- JMail interface objects
- log manager
- logging manager
- menu bar template
- message
- message server
- reasoning routines
- show workspace menu item
- top-level domain object
- translation objects
- cross-sections
 - creating
- customer support services

D

- Debug Specific Fault Models menu choice
- Default User Mode attribute
- Default Web Location attribute
- defining
 - browser
 - escalation specification
 - initializations
 - status bar
- Delete Background Image menu choice
 - deleting background images, using
 - Workspace menu
- Delete menu choice
 - deleting objects, using
 - deleting workspaces, using
 - Edit menu
- deleting
 - message
 - modules
 - objects
 - workspaces
- descriptor preferences
- Desktop Layout
- details
 - displaying for objects
 - showing
 - for container objects
 - superior object of
- Developer mode
 - configuring user preferences for
- Diagnose menu choice
 - summary
- Diagnosis Managers menu choice
 - summary
- Diagnostic Console menu choice

- summary
- disconnecting
 - from the client
 - using menu
- Documentation menu choice
- domain map
 - components of
 - definition of
 - exporting
 - importing
 - placing domain object on
 - tutorial
- domain object
 - connecting
 - connecting across workspaces
 - creating
 - creating subworkspace for
 - importing
 - using wizard
 - naming
 - placing on domain map
- Down menu choice

E

- Edit menu
- editing
 - initializations
- email
 - configuring
 - address
 - format
 - to send
 - delivering messages by
 - starting JMail Bridge
 - startup parameters for sending
- Enable Status Bar Message Browser attribute
- Enable Tuning menu choice
- error handler
 - customizing
 - description of
 - logging facility
- errors
 - creating new handler
 - default handler
 - logging
- escalation specification
 - defining
 - duration of phases of
 - priority of

- procedures called in
- target of
- timing of
- event
 - definition of
 - interpreting
 - parsing
- Event and Alarm Metrics menu choice
- events
 - correlating
 - filtering
 - handling
 - logging
 - relating to domain objects
- Events queue
- existing application
 - adding Integrity functionality to
- Exit menu choice
 - exiting the server, using
- exporting a domain map
- Extended Menus attribute

F

- F4 key
- Fault Models menu choice
 - summary
- faults
 - diagnosing
 - preventing
- file
 - closing a log file
- File menu
- files
 - g2.ok*
 - StartServer.bat*
 - twng.exe*
- filtering
 - based on domain relationships
 - duplicate messages
 - events
 - in the bridge
 - paired events
 - using message histories
 - using past events
- filters
 - using with browser
- Finder Options
- Finder options preferences
- Flip Horizontally menu choice
 - Layout menu

- Flip Vertically menu choice
- Layout menu
- Foreground Color attribute
- Forward menu choice
- Go menu

G

- G2
 - components of
- G2 Help Topics menu choice
- G2 JMail Bridge menu choice, Start menu
- G2 toolbox
 - g2.ok* file
- Get menu choice
 - Workspace menu
- GIF files, loading as background images
- Go menu
- Go To menu choice
 - manage dialog
 - project hierarchy
 - Search dialog
- Go to Superior menu choice
 - View menu

H

- Help menu
- Hide menu choice
 - View menu
- histories
 - message
- history
 - message
 - querying
- Home menu choice
 - Go menu
- Home Process Map attribute
- HTTP menu choice

I

- icons
 - creating
 - for connections
- Import menu choice
- importing
 - domain map
 - from text file
 - using wizard

- MIBs
- Indicate Items attribute
 - configuring
- inheritance
 - definition of
- initialization
 - creating
 - editing
- initializations
 - defining
- Initialize Application menu choice
 - Project menu
- Integrity
 - adding functionality to existing application
 - Core Services
 - exiting
 - other modules for
 - starting server
 - in secure G2 environment
- Integrity Help Topics menu choice
- Integrity toolbar
 - View menu
- Integrity toolbox
 - using
- interface
 - setting up
- Interface Pools menu choice
 - Project menu
- Interfaces menu choice
 - Project menu
 - SMTP

J

- Java Mail (JMail)
 - configuring
 - in configuration file
 - in user preferences
- JMS menu choice
- JPEG files
 - loading as background images
 - saving workspaces to

K

- knowledge base
 - definition of

L

- layering
- Layout menu
- Layout toolbar
 - View menu
- Left menu choice
- list and array editing
 - enabling
- Load Background Image menu choice
 - loading background images, using Workspace menu
- Load Options
- local text resource
 - adding columns in spreadsheet of
- logging
 - closing log file
 - customizing
 - definition
 - description of
 - events
 - information
 - messages
- logging manager
 - creating
- Logic menu choice
 - summary

M

- Manage dialog
 - displaying object properties and details using
- Manage menu choice
- managed object
 - definition of
- managing
 - objects
 - using Manage dialog
 - using Project menu
- menu
 - cascade item
 - cloning
 - compiling
 - connection items
 - constructing
 - displaying browser on
 - local text resource for
 - selecting a workspace for
 - text resource for
 - using

- view components of
- menu bar template
 - configuring
 - creating
- menu selection icon
 - adding
- menus
 - Edit
 - File
 - Go
 - Help
 - Layout
 - Model
 - Project
 - Tools
 - Workspace
- merging
- modules
- message
 - acknowledging
 - alarms
 - classes
 - color of
 - creating
 - definition
 - definition of
 - deleting
 - histories
 - history
 - interacting with using browser
 - logging
 - querying history
- Message Board menu choice
 - View menu
- Message Browser
- Message Browser menu choice
 - View menu
- message browsers
 - configuring
 - for modeler mode
 - for operator mode
 - showing by default in operator mode
- Message Browsers menu choice
- message information base
 - description of
- message server
 - configuring
 - creating
 - defining
 - read and write contents to file
- message system

- customizing
 - setting up
- messages
 - acknowledging
 - delivering by email
 - search for related
- Messages queue
- method
 - definition of
- MIBS, importing
- Mobile Email
 - address
 - Notification
- Model menu
- Modeler Browser attribute
- Modeler mode
 - configuring
 - user preferences for
 - description of
- models
 - working with
- modules
 - available for Integrity
 - definition of
 - deleting
 - merging
 - renaming
 - saving
 - working with
- My User Preferences menu choice
 - configuring user preferences, using
 - Project menu

N

- Navigator
 - menu choice
- Navigator Button
- Navigator menu choice
 - View menu
- Navigator preferences
- network
 - viewing information about
- Networks & Devices menu choice, Project menu
- New Instance menu choice, project hierarchy
- New menu choice
 - creating
 - top-level workspaces, using
 - File menu

- Workspace menu
- Normal menu choice
- Nudge menu choice
 - Layout menu

O

- Object Models menu choice
- objects
 - adjusting the order of
 - aligning
 - copying
 - definition of
 - deleting
 - displaying properties for
 - distributing
 - editing colors
 - flipping
 - interacting with
 - in Modeler mode
 - managing
 - nudging
 - resizing
 - rotating
 - selecting
 - all
 - individual
 - transferring
 - working with
- Open menu choice
 - File menu
- Operations Expert
 - features and benefits
- Operator Actions menu choice
- Operator Browser attribute
- Operator mode
 - configuring user preferences for
 - description of
 - user mode
- opfo-external-name
 - assigning
- Order menu choice
 - Layout menu

P

- palette/messages preferences
- palettes
 - using items from
- popup menus

- interacting with objects, using
- popup menus, displaying
- PPD files, processing
- Preferences
- Print menu choice
 - File menu
- priority
 - color of
 - setting colors of
 - setting default
- Project
 - menu
 - managing objects, using
 - using
 - using submenus
- Project menu
- properties dialogs
 - shortcuts for displaying
- properties dialogs, displaying
- properties files
 - backup copies
- Properties menu choice
 - Edit menu
 - for items on workspaces

Q

- Queues menu choice

R

- reasoning routine
 - creating
 - definition of
 - examples of
- Refresh menu choice
 - Go menu
- remote procedure call
- renaming
 - modules
- Reports menu choice
- resizing objects
- Restore Last Pane Settings attribute
- Right menu choice
- Rotate or Flip menu choice
 - Layout menu

S

- Save as JPEG menu choice

- File menu
- Save As menu choice
 - File menu
- Save menu choice
 - File menu
- Save Options
- saving
 - modules
- scaling workspaces
- search
 - for related messages
- Search menu choice
 - Tools menu
- secure G2, running in
- selecting browser messages
- Select All menu choice
 - Edit menu
- Send to Back menu choice
 - Layout menu
- sender
 - definition of
- server
 - connecting to
 - default
 - specific
 - disconnecting from
 - shutting down
 - using menus
 - starting
 - on specific port
- Server Information menu choice
- Set Default User Mode attribute
- Setting Preferences
- setup wizard
- Show Detail menu choice
 - summary of common tasks
- View menu
 - workspaces
- Show Logbook attribute
- Show Users menu choice
- show workspace menu item
 - creating
- Shrink Wrap menu choice
 - Layout menu
- Shut Down G2 menu choice
- shutting down server
 - using menus
- SMTP menu choice
- SNMP setup
- sorting
 - configuration on browser

- SQL menu choice
- Standard toolbar
 - View menu
- startServer.bat* file
- status bar
 - custom display and hide procedures for defining
- Status Bar menu choice
 - View menu
- Stop menu choice
 - Go menu
- stubs
 - adding and deleting
 - creating
- subclass
 - adding attributes to
 - creating icons for
 - defining
 - displaying attributes for
 - viewing attributes of
- subscribers
 - configuring
- subworkspace
 - creating
- System Performance menu choice
- System Settings menu
- System-Administrator mode
 - configuring user preferences for
 - description of

T

- Tabbed Mdi Mode attribute
- target
 - definition of
- Telnet Command attribute
- Templates menu choice
- Text Parsing menu choice
- Tip of the Day
- toolbars
 - Integrity
 - Layout
 - Standard
 - using
 - Web
- toolbox
 - G2
 - Integrity
- Toolbox - G2 menu choice
 - using

- Toolbox - Integrity Export Import menu choice
- Toolbox - Integrity menu choice
- Toolbox - Message Parsing Engine menu choice
- Toolbox - ODIE Subscriber menu choice
- Toolbox - OPAC menu choice
- Toolbox - SNMP Traps menu choice
- Tools
 - menu
- top-level domain object
 - creating
- Transfer menu choice
 - Edit menu
- translation objects
 - creating
 - using
- trapd.conf.ppd* file
- twng.exe* file

U

- Uninitialize Application menu choice
 - Project menu
- Up menu choice
- User Interface Theme attribute
- User Mode menu choice
 - switching user modes, using
 - Tools menu
- user mode options
- user modes
 - configuring default
 - specifying user preferences for different
 - switching
- User Name attribute
 - Modeler mode
- user preferences
 - configuring
 - in Modeler mode
 - creating and configuring
 - specifying for different types of users
- User Preferences menu choice
 - configuring user preferences, using
 - Project menu
- Users menu choice

V

- virtual desktop preferences

W

- Web toolbar
 - View menu
- Window menu
- wizard, setup
- workspace
 - definition of
- Workspace Margin attribute
- Workspace menu
 - Delete Background Image
 - description of
 - Get
 - Load Background Image
 - New
- workspaces
 - adjusting borders for
 - deleting
 - editing
 - colors of
 - margins of
 - name of
 - properties
 - hiding
 - interacting with
 - loading background images
 - printing
 - saving as JPEG
 - scaling
 - showing superior object of detail
 - shrink wrapping

X

- XMB files, loading as background images

Z

- Zoom In menu choice
 - View menu
- Zoom menu choice
 - View menu
- Zoom Out menu choice
 - View menu
- Zoom to Fit menu choice
 - View menu

